Ne-Zheng Sun · Alexander Sun

Model Calibration and Parameter Estimation

For Environmental and Water Resource Systems

Springer

Model Calibration and Parameter Estimation

Ne-Zheng Sun • Alexander Sun

Model Calibration and Parameter Estimation

For Environmental and Water Resource Systems

Springer

Ne-Zheng Sun
Department of Civil and Environmental Engineering
University of California at Los Angeles
Los Angeles
California
USA

Alexander Sun
Bureau of Economic Geology, Jackson School of Geosciences
University of Texas at Austin
Austin
Texas
USA

ISBN 978-1-4939-2322-9
ISBN 978-1-4939-2323-6 (eBook)
DOI 10.1007/978-1-4939-2323-6

Library of Congress Control Number: 2015931547

Mathematics Subject Classification (2010): 97Mxx, 93A30, 93B11, 93B30, 90Cxx, 65Kxx, 86A05, 15A29, 35R30, 86A22, 65M32, 65N21, 81T80, 62P12, 65Cxx

Springer New York Heidelberg Dordrecht London

Printed on acid-free paper

Springer New York is part of Springer Science+Business Media (www.springer.com)

To
Rachel, Albert, Adam, and Jacob

Preface

A mathematical model constructed for a real system must be calibrated by data and its uncertainty must be assessed before used for prediction, decision making, and management purposes. After more than a half century of study, however, the construction of a reliable model for complex systems is still a challenging task.

In mathematical modeling, the “prediction problem” or the “forward problem” (inputs $\rightarrow$ outputs) uses as model inputs a fixed model structure, known model parameters, given system controls, and other necessary information to find the system states (as model outputs). Unless all properties of the modeled system can be measured directly, model inputs tend to always contain unknowns or uncertainties that have to be determined indirectly. Model calibration (outputs $\rightarrow$ inputs) uses the measured system states and other available information to identify or estimate the unknown model inputs. Thus, in a certain sense it is the “inverse problem” of model prediction. The history of studying model calibration is probably as long as the history of forward modeling, but the progress of study had been slow due to the very nature of inversion: identifying the causes from results is always more difficult than predicting the results on the basis of known causes. Various optimization-based data-fitting methods developed for solving the classical inverse problem in mathematics have been proven to be successful for model calibration, only if a system’s structure is simple and well-defined, and both the number and dimensions of unknown parameters are low. When these assumptions do not hold, the use of data-fitting may produce an unacceptable model.

With the advances in computing, instrumentation, and information technologies, more and more sophisticated numerical models have been developed for simulating complicated physical, chemical, and biological processes observed in environment, energy, water resources, and other scientific and engineering fields. The advent of highly sophisticated software packages has made solution to the “forward problem” much easier, but, at the same time, calibrating the resulting model becomes more difficult due to the increase of model complexity. Modelers gradually realize that: (1) the requirement of model uniqueness has to be given up; (2) the classical concept of model inversion must be extended to include model structure identification, model reduction, and model error quantification; and (3) five fundamental problems of modeling technology (i.e., the model selection problem, model calibration problem, model reliability problem, model application problem, and data collection problem) must be considered systematically rather than separately or sequentially during the model construction stage. The model calibration problem has thus become more challenging but, at the same time, the process of solving the problem becomes more interesting and rewarding than the simple data-fitting exercise. Empowered by these methodological understandings and newly developed tools in mathematics and statistics, the research on model construction has made significant progress in recent years. As a result, existing methods have been improved and new and promising approaches have emerged.

This book provides a comprehensive introduction on all aspects of constructing useful models: from the deterministic framework to the statistical frameworks; from the classical inverse problem of parameter estimation to the extended inverse problem of system structure identification; from physical-based models to data-driven models; from model reduction to model uncertainty quantification; from data sufficiency assessment to optimal experimental design; and from basic concepts, theory, and methods to the state-of-the-art approaches developed for model construction. A central problem to be considered in this book is how to find surrogate models for predetermined model applications.

Chapter 1 is a general description of the modeling technology. Models that are often seen in environmental and water resources fields are introduced here and are used to exemplify different methods throughout the book. Based on different criteria of model calibration, three kinds of inverse problem are defined: the classical inverse problem (CIP) for parameter estimation, the extended inverse problem (EIP) for system identification, and the goal-oriented inverse problem (GIP) for model application. After reading this chapter, readers will be able to get a holistic picture of mathematical modeling, know the difficulties and problems of model construction, and learn how this book is organized.

Part I of this three-part book (Chapter 2–5) is contributed to the solution of CIP. Basic concepts and methods on linear model inversion and single-state nonlinear model inversion are given in Chapter 2; singular value decomposition and various nonlinear optimization algorithms are introduced briefly. In Chapter 3, the multi-state model inversion is cast into a multi-objective optimization problem and solved by the evolutionary algorithms. Regularization is also introduced in this chapter from the point of view of multi-criterion inversion. The inverse problem is reformulated and resolved in the statistical framework in Chapter 4. Monte Carlo based sampling methods, including the Markov Chain Monte Carlo method, are introduced for finding the posterior distribution. Various methods of model differentiation are given in Chapter 5. Model differentiation is a necessary tool for almost all topics covered in this book.

Part II of this book (Chapter 6–8) is dedicated to the solution of EIP. In Chapter 6, various methods for parameterizing deterministic functions or random fields are introduced. Principal component analysis and other linear and nonlinear dimension reduction methods, as well as their applications to inverse solution, are covered. Model structure identification and hyperparameter estimation are the main topics of Chapter 7, in which various adaptive parameterization approaches, the level set method, multiscale inversion, and geostatistical inversion are introduced. Methods for constructing data-driven models are given in Chapter 8, including linear regression and various machine learning methods such as artificial neural networks, support vector machine, and Gaussian process regression.

Part III of the book (Chapter 9−12) is contributed to the topic of model reliability. Chapter 9 introduces various data assimilation methods for inverse solution that allow us to update a model continuously to improve its reliability whenever new data become available. Methods used for uncertainty quantification, including Monte Carlo simulation, global sensitivity analysis, stochastic response surface, are introduced systematically in Chapter 10. The effects of model parameter uncertainty and model structure uncertainty on model outputs are assessed. To construct a more reliable model, more data are needed. Design of informative and cost-effective data collection strategies is the subject of Chapter 11, in which, optimal experimental design is formulated into a multi-objective optimization problem. The criteria of optimal design for linear model inversion are derived. For nonlinear model inversion, Bayesian and robust design methods, especially, the interval-identifiability-based robust design, are introduced. In Chapter 12, after the goal-oriented forward problem is described, the GIP is formulated and solved in both the deterministic and statistical frameworks. When the existing data are insufficient, a cost-effective experimental design method is given. Finally, the goal-oriented pilot-point method is described.

Preliminary mathematics required for reading this book is reviewed in details in three Appendices. To help readers better understand the text, review questions are given at the end of each chapter. All major methods introduced in this book are illustrated with numerical examples created by the authors, including the information on available toolboxes. Alex Sun authored Chapters 6, 8, 9, and 10 and edited the whole book. Other chapters are authored by Ne-Zheng Sun. This book can be used as a textbook for graduate and upper-level undergraduate students majoring in environmental engineering, hydrology, or geosciences. It also serves as an essential reference book for petroleum engineers, mining engineers, chemists, mechanical engineers, biologists, medical engineers, applied mathematicians, and others who perform mathematical modeling. Much of the research conducted by the authors over the years has been made possible by the support from U.S. National Science Foundation (NSF), National Aeronautics and Space Administration (NASA), Department of Energy (DOE), Environmental Protection Agency (EPA), and Nuclear Regulatory Commission (NRC). We are grateful to all of our current and past collaborators for insightful discussions. Ne-Zheng Sun would like to give special thanks to Drs. Jacob Bear and William Yeh for their guidance, support, collaboration, and long-term friendship. Alex Sun would like to thank Drs. Yoram Rubin and Dongxiao Zhang, and his colleagues at the University of Texas at Austin for their advice and collaboration.

We are grateful to the editors Achi Dosanjh, Donna Chernyk, and Danielle Walker at Springer for their support and guidance in every step of the process. Finally, we would like to thank Fang and Zhenzhen for their endless love, sacrifice, and patience during this multiyear-long book project.

Santa Monica, CA — Ne-Zheng Sun
Austin, TX — Alex Sun

Contents

Symbols and Notation

A	A set
$\mathcal{A}$	Linear operator
$\mathcal{A}^*$	Adjoint operator
$A_{w,m}(u)$	Smolyak formula for approximating u in m-dimensional domain at w-level
$AE(\cdot,\cdot)$	Model application error
$\mathbf{b}$	Boundary condition parameters
B	Bias of a point estimator
$C(\mathbf{x},t)$	Mass concentration field
$C_0(\mathbf{x})$	Initial mass concentration
C_{obs}	Observations of concentration field
$C(\mathbf{r})$	Autocovariance function
$\mathbf{C}$	Covariance matrix
$\mathbb{C}$	A list of subregions of the admission region
$\mathbf{C}_D$	Observation error covariance
$\mathbf{C}_P$	Parameter covariance
Cov	Covariance
$\mathbf{d}$	Data vector
$\mathbf{d}_k$	Displacement or search direction vector resulting from k-th iteration in nonlinear solvers
D	Dispersion coefficient
	Observation data
	A set of design variables
D_E^*	Optimal design under probability average
D_i	Elementary effect of i-th input parameter
D_d	Molecular diffusion coefficient
$\mathcal{D}(u)$	Observation design that serves as mapping from U_{ad} to F_{ad}
$\mathcal{D}^{-1}$	Inverse mapping from $\mathbb{F}$ to $\mathbb{U}$
$\mathbf{D}$	Dispersion tensor
$\mathcal{DM}$	Combination of forward and sampling mapping

$\mathbf{e}_i$	Unit vector along the *i*-th coordinate axis
$\mathbf{e}_D$	Observation error vector
E	Sink/source term
	Performance function in adjoint state method
$E(\cdot)$	Expectation operator
Erfc	Complementary error function
$f(\cdot)$	Generic function
$\hat{f}(\cdot)$	Reduced-order approximation of $f(\cdot)$
$f_i(\boldsymbol{\theta})$	Measured value of $f(\cdot)$ at location $\mathbf{x}_i$
	The *i*-th objective function of an MOO problem
$\mathbf{F}$	Nonlinear operator
	Continuously differentiable vector function
F_{ad}	Value region of observation design
$\mathbb{F}$	Observation space
$\mathcal{F}$	Mapping in data-driven model
g	Gravity acceleration
$\mathbb{G}$	Goal space
$\mathbf{g}(\boldsymbol{\theta})$	Gradient vector
$g(\cdot)$	Measurement equation
$G(\theta)$	Data fitting term, $\left\Vert u_D(\theta) - u_D^{obs} \right\Vert_{\mathbb{F}}$
$\mathbf{G}$	Coefficient matrix in a linear system of equations
$\mathbf{G}_k$	Measurement matrix in Kalman filter
$\tilde{\mathbf{G}}_k$	Measurement matrix in Kalman filter for augmented state
$\mathbf{G}^{\dagger}$	Moore–Penrose pseudoinverse of $\mathbf{G}$
$h(\mathbf{x}, t)$	Water depth in Saint Venant equation
	Hydraulic head in porous flow equation
H	Hypothesis
H_i	Univariate Hermite polynomial of order *i*
$H(\cdot)$	Heaviside function
	Coefficient matrix
$\mathbf{H}$	Hessian matrix
$\mathcal{H}(p)$	Differential entropy of probability distribution p
$\mathcal{H}_0$	Hartley measure
$\mathcal{H}_r$	Relative entropy (differential entropy of a continuous random variable)
$\mathcal{H}_s$	Shannon entropy measure
I	Prior information
$I_k(\mathbf{x})$	Indicator spatial random function
$\mathbf{J}_D$	Sensitivity or Jacobian matrix
$k(T)$	Thermal conductivity
$k(\cdot,\cdot)$	Kernel function

$K(\mathbf{x})$	Hydraulic conductivity
$k(t)$	Transfer function
K_s	Coefficient in Monod model of biomass growth
$k(\cdot,\cdot)$	Kullback–Leibler relative entropy
$\mathbf{K}$	Kernel matrix
$\mathbf{K}_k$	Kalman gain matrix
L	Lipschitz constant
	Lagrange function
L_j	Lagrange interpolation formula
L_i	Laguerre polynomial of order *i*
$\mathbf{L}$	Laplacian matrix
$\mathcal{L}$	Operator
$\boldsymbol{\mathcal{L}}$	Multistate model operator
$L(\boldsymbol{\theta})$	Likelihood function
L_2	Euclidean norm, $L_2 \triangleq \lVert\cdot\rVert_2$
M^t	Real system model
M^*	Identified model
$\mathcal{M}$	Forward mapping (or forward model)
$\mathcal{M}^{-1}$	Inverse mapping
$\mathbf{M}_k$	Model matrix in Kalman filter
$\tilde{\mathbf{M}}_k$	Model matrix in Kalman filter for augmented state vector
$\mathbb{M}$	System space
$\mathfrak{M}$	A set of candidate models for model selection
n	Manning's roughness coefficient
$\mathbf{n}$	Unit normal vector
$\mathcal{N}(\cdot)$	Gaussian distribution
$p(t)$	Precipitation
$p(\boldsymbol{\theta})$	Probability distribution function of $\boldsymbol{\theta}$
$p_0(\boldsymbol{\theta})$	Prior probability distribution function of $\boldsymbol{\theta}$
$p_*(\boldsymbol{\theta})$	Posterior probability distribution of $\boldsymbol{\theta}$
$\mathbf{p}$	Parameters characterizing system internal properties
P	Probability
$\mathbf{P}_k$	State covariance matrix
$P(t)$	Precipitation
P_{ad}	Admissible region of parameters
P_0	Subset of P_{ad}
P_i	Legendre polynomial of order *i*
$\mathcal{P}$	Set of generators of a Voronoi diagram
$\mathbb{P}$	Parameter space
$\mathbb{P}_k$	A subregion of $\mathbb{P}$
$PE(\cdot,\cdot)$	Parameter deviation

$q(t)$	Spring discharge rate
$q\left(\theta \middle| \theta_i\right)$	Proposal distribution
$\mathbf{q}$	Control variables
Q	Streamflow rate
	Pumping rate
Q_R	Net recharge rate per unit area
$\mathbf{Q}$	Precision matrix
$\mathbf{Q}_\upsilon$	Covariance of parameter error
$\mathbf{Q}_\eta$	Covariance of model error
r	Metropolis ratio
R	Reaction term
	Time domain
$R(\theta)$	Regularization term in bi-criterion solution, $\left\|\theta - \theta_{pri}\right\|_{\mathbb{P}}$
$\mathbf{R}$	Measurement error matrix
$\mathbb{R}$	Real space
$RE(\cdot,\cdot)$	Fitting residual error
$\dot{s}_{k,j}$	Partial derivatives $\partial s_k / \partial \theta_j$ in AD forward mode
$\bar{s}_{k,i}$	Partial derivatives $\partial u_i / \partial s_k$ in AD backward mode
s_i	The i-th singular value of a matrix
S	Nutrient concentration in Monod model of biomass growth
	Storage coefficient
S	Model structure
$S(\theta)$	Distance (i.e., norm of residual) between model output, $u_D(\theta)$, and observations, u_D^{obs}
$S_\alpha(\boldsymbol{\theta})$	Objective function of a weighted-criterion, single-objective optimization problem
S_o	Channel bed slope
S_f	Channel energy slope
S_y	Specific yield of unconfined aquifer
$\underline{S}$	Lower bound of objective function in a branch-and-bound subregion
S_σ	Variance-normalized sensitivity coefficient
S_i	First-order Sobol' index of *i*-th parameter
S_i^T	Total-order Sobol' index of *i*-th parameter
S_{ij}	Elements of local sensitivity matrix
$\mathbf{S}$	Model structure
t	Time
T	Upper bound of a time interval
	Transmissivity
	Temperature
$\mathcal{T}$	Template function in pattern-based geostatistics
T_h	Transmissivity
u	System state (scalar)

$\mathbf{u}$	System states (vector)
$\mathbf{u}_0$	Initial condition of system states
$\boldsymbol{u}_D$	Model output corresponding to observation data
$\mathbf{u}_D^{obs}$	Observations of system states
$\tilde{u}$	Approximation of true system state
U_k	Exogenous variables
U_{ad}	Value region of forward solution
$\mathbb{U}$	State space
$\mathbf{U}$	Unitary matrix from SVD
$\mathbf{U}_r$	Unitary matrix containing *r* leftmost columns of $\mathbf{U}$
$U(\cdot)$	Utility function
$v(x,t)$	Stream flow velocity
	Advection velocity
$\mathbf{v}(\mathbf{x},t)$	Darcy flux field
$\mathbf{V}(\mathbf{x},t)$	Average velocity field
V	Total variance
V_i	Univariate partial variance
$V_{i,j}$	Bivariate partial variance
$V_{\mathcal{P}}(\mathbf{x})$	Voronoi cell of generator x
V_t	White noise
$\mathbf{V}$	Unitary matrix from SVD
$\mathbf{V}_r$	Unitary matrix containing r leftmost columns of $\mathbf{V}$
Var	Variance
w_i	Weights
$\mathbf{w}$	Weight vector
$\mathbf{W}$	Weight matrix of weighted least squares
X_t	Random process
$\hat{X}$	Predicted value
$\mathbf{X}$	Multivariate random variable
	Input data matrix
$\mathbf{X}_s$	Sample set matrix
$\mathbf{z}$	Observation data vector
α	Order of convergence
	Regularization parameter
	Acceptance probability of Metropolis-Hastings algorithm
α_i	Weights assigned to objective functions when converting from multiple objectives to a single objective
α_L	Longitudinal dispersivity
α_T	Transverse dispersivity
β	Shape parameter of weighting functions used by inverse distance weighting
	Shape parameter of Gaussian distribution

δ	Variation
δ_{ij}	Kronecker delta
Δ_k	AIC difference between the *k*-th model and the best performer of all candidate models
ε	Shape parameter of Gaussian RBF
	Parameter accuracy requirement
$\boldsymbol{\varepsilon}$	Observation error
ϕ_i	Basis function
$\phi(\cdot)$	Objective function of optimality criteria
Φ	Mapping from real space to feature space
$\mathbf{\Phi}$	Design matrix
Γ	Set of all boundaries
η	Norm of observation error
	Lagrange multiplier
$\boldsymbol{\eta}_k$	Process error
$\tilde{\boldsymbol{\eta}}_k$	Augmented process error vector
κ	Matrix condition number
λ	Step size along search direction
λ_i	Eigenvalue
	Lagrange multiplier
λ_i^0	Weights of ordinary kriging
γ	Scale parameter in Markov random field
	Prediction accuracy requirement
$\gamma(\mathbf{r})$	Semivariogram
$\gamma(\cdot)$	Autocovariance function
μ	Growth rate in Monod model
	Mean of univariate Gaussian distribution
	Drift parameter in regression kriging
$\boldsymbol{\mu}$	Mean of multivariate Gaussian distribution
$\mu_{\max}$	Maximum growth rate in Monod model
ν_G	Voronoi diagram centers
ξ	Germ
Ω	Spatial domain of a model
	Integration volume
Ω_j	Sample space of a random variable/process
	Parameter zone in zonation method
Ω_T	Library of pattern templates
θ	A system parameter
$\boldsymbol{\theta}$	System parameters
$\hat{\boldsymbol{\theta}}$	Point estimator of $\boldsymbol{\theta}$
	Parameterized form of a distributed parameter θ
$\tilde{\boldsymbol{\theta}}_D$	Worst-case parameter for design D
$\boldsymbol{\theta}_0$	Staring value of parameters for iterative solvers

$\boldsymbol{\theta}_{CG}$	Conjugate Gaussian estimator of $\boldsymbol{\theta}$
$\boldsymbol{\theta}_{cgm}$	Constrained global optimum
$\boldsymbol{\theta}_{GLS}$	Generalized least squares estimator of $\boldsymbol{\theta}$
$\boldsymbol{\theta}_{GTR}$	General Tikhonov regularization solution
$\boldsymbol{\theta}_{LS}$	Least squares solution
$\boldsymbol{\theta}_{OLS}$	Ordinary least squares solution
$\boldsymbol{\theta}_{MAP}$	Maximum a posteriori estimator of $\boldsymbol{\theta}$
$\boldsymbol{\theta}_{MLE}$	Maximum likelihood estimator of $\boldsymbol{\theta}$
$\boldsymbol{\theta}_{p}$	Prior guess of parameter $\boldsymbol{\theta}$
$\boldsymbol{\theta}_{S}$	Parameters associated with model structure S
$\boldsymbol{\theta}_{WLS}$	Weighted least squares solution
θ^{t}	True system parameter
$\boldsymbol{\theta}^{U}, \boldsymbol{\theta}^{L}$	Upper and lower bound of $\boldsymbol{\theta}$ in constrained optimization
$\boldsymbol{\theta}_{qs}$	Quasi-solution obtained by minimizing $S(\boldsymbol{\theta})$
$\boldsymbol{\theta}^{\dagger}$	SVD pseudo inverse solution
$\boldsymbol{\theta}_{ts}^{\dagger}$	TSVD pseudo inverse solution
$\boldsymbol{\theta}^{*}$	Minimizer of $S(\boldsymbol{\theta})$
$\rho(\cdot)$	Autocovariance function
σ	Standard deviation
σ_X^2	Variance of random process X_t
$\boldsymbol{\Sigma}$	Diagonal matrix from SVD decomposition containing singular values as elements Covariance matrix of multivariate Gaussian
$\boldsymbol{\Sigma}_r$	Singular value matrix retaining only *r* non-zero singular values
$\boldsymbol{\Sigma}_D$	Observation error covariance matrix
$\boldsymbol{\Sigma}_P$	Covariance matrix of parameter
τ	Dummy integration variable
	Tortuosity
ω_i	Weights used for converting multi-objective functions into single objective
ξ_i	Orthogonal random variable
	Slack variables used in SVM
ψ	Matrix potential in vadose zone
	Adjoint state
	Sigmoid function
$\boldsymbol{\psi}$	Hyperparameter set
$\boldsymbol{\Psi}$	Multivariate polynomial basis function
υ_k	Error vector in parameter evolution equation

Symbols, Operators

$\hat{}$	Estimated quantity
$\varnothing$	Empty set
$\otimes$	Tensor product
$\cap$	Set intersection operator
$\cup$	Set union operator
$\lVert\cdot\rVert_1$	L-1 norm
$\lVert\cdot\rVert_2$	L-2 norm
$\sum$	Summation
$\prod$	Product
∇	Gradient
$\nabla\cdot$	Divergence
Δ	Variation
$\{\ldots\}$	Set
$(\cdot,\cdot)$	Inner-product operator
$\bigcup$	Union

Acronyms

AD	Automatic differentiation
AIC	Akaike information criterion
ANN	Artificial neural network
ANOVA	Analysis of variance
AR	Autoregressive model
ARMA	Autoregressive moving average model
ARMAX	Autoregressive moving average with exogenous inputs
BIC	Bayesian information criterion
BnB	Branch and bounding algorithm
BFGS	Broyden–Fletcher–Goldfarb–Shanno algorithm for solving unconstrained nonlinear optimization problems
CVE	Cross validation error
CIP	Classical inverse problem
CDF	Cumulative distribution function
DIRECT	Dividing rectangles algorithm
DOF	Degrees of freedom
EKF	Extended Kalman filter
EIP	Extended inverse problem
EnKF	Ensemble Kalman filter

EWR	Environmental water resources
FA	Factor analysis
FNN	Feedforward neural network
FPE	Final prediction error
GA	Genetic algorithm
GFP	Goal-oriented forward problem
GIP	Goal-oriented inverse problem
GLS	Generalized least squares
GMRF	Gaussian Markov random field
GP	Gaussian process
GPR	Gaussian process regression
GPCE	Generalized polynomial chaos expansion
GRF	Gaussian random field
GSA	Global sensitivity analysis
IK	Indicator kriging
IDW	Inverse distance weighting
INI	Interval identifiability
KF	Kalman filter
KL	Karhunen-Loève expansion
KPCA	Kernel principal component analysis
LHS	Latin hypercube sampling
LPA	Local polynomial approximation
L-BFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm
MAP	Maximum a posteriori
MLE	Maximum likelihood estimator
MLP	Multilayer perceptron network
MOCOM	Multiobjective complex evolution
MOEA	Multiobjective evolutionary algorithm
MOO	Multi-objective optimization
MRE	Minimum relative entropy
MSE	Mean square error
MRF	Markov random field
NSGA	Nondominated sorting genetic algorithm
OK	Ordinary kriging
OED	Optimal experimental design
OLS	Ordinary least squares
PC	Principal component(s)
PCA	Principal component analysis
PCE	Polynomial chaos expansion
PDE	Partial differential equation
PDF	Probability distribution function
POD	Proper orthogonal decomposition
PnT	Pump-and-treat system
RBF	Radial basis function
RBFN	Radial basis function network

RF	Random function
RMSE	Root mean square error
RVM	Relevance vector machine
SA	Simulated annealing
SK	Simple kriging
SCM	Stochastic collocation method
SDA	Sequential data assimilation
SOO	Single-objective optimization
SRM	Structural risk minimization
SVD	Singular value decomposition
SVM	Support vector machine
SVR	Support vector regression
TSVD	Truncated singular value decomposition
UQ	Uncertainty quantification
VC	Vapnik-Chervonenkis dimension
WCP	Worst-case parameter
WLS	Weighted least squares

Chapter 1
Introduction

The German physicist Werner Heisenberg once said, "What we observe is not nature herself, but nature exposed to our method of questioning." Mathematical models have long served as such a tool of questioning. A mathematical model of a physical system quantitatively describes the relationship among various variables that characterize the states, internal structure, and external conditions of the system.

In Sect. 1.1, several commonly seen models in environmental and water resources (EWR) studies are shown. A general form of mathematical models and their classifications is then provided. Different mathematical models are constructed across different science and engineering disciplines. Models used in the EWR fields are often nonlinear, dynamic, stochastic, and governed by partial differential equations (PDEs). The traditional process of constructing a EWR model involves data collection, conceptualization, model calibration/parameter estimation, and finally, the evaluation of model reliability. During this process, the following problems must be solved: (1) the *forward problem* for simulation and prediction; (2) the *inverse problem* for model calibration and parameter estimation; (3) the *design problem* for effective data collection; and (4) the *reliability problem* for model application.

In EWR modeling, the solution of forward problem has become routine. We assume that readers of this book are already familiar with the solution of forward problems in their own fields. Thus, no attempt is made to survey numerous forward solution techniques existing in the literature; instead, a brief introduction to basic analytical and numerical methods is provided in Sect. 1.2.

Data types available for model construction include prior information, direct measurements of parameters, observations of state variables, as well as the accuracy requirement of model applications. Depending on the availability and use of different types of data, different approaches and criteria for model calibration and parameter identification exist. The *classical inverse problem* (CIP) reverses the forward solution to seek model inputs from model outputs, under the assumption that the model structure error is absent. In Sect. 1.3, two common criteria, "fitting observed data" and "using prior information," are used to formulate the inverse problem in both deterministic and statistical frameworks. The *extended inverse problem* (EIP),

N-Z. Sun, A. Sun, *Model Calibration and Parameter Estimation,*
DOI 10.1007/978-1-4939-2323-6_1

requiring the identification of both model structure and model parameters in an adaptive sense, is also introduced in that section.

The model reliability problem to be introduced in Sect. 1.4 has recently become one of the most concerned issues in EWR modeling. But this problem is very challenging because of the inherent difficulties related to determining the model scale, model complexity, and data sufficiency. The *goal-oriented inverse problem* (GIP) incorporates "reliability assurance" as the third criterion in its formulation, in which separate models are constructed for the same system according to the goals of model applications and associated accuracy requirements.

1.1 Mathematical Modeling

1.1.1 Modeling an Open System

A EWR system, such as a watershed or groundwater basin, is an *open system* that continuously exchanges mass and/or energy with its surroundings. The states of a EWR system are determined by both its internal structure and external conditions. The following types of variables are often used to characterize a EWR system:

- *State variables* that characterize the states of a system
- *System parameters* that characterize the structure and properties of a system
- *Boundary parameters* that describe the outer conditions of a system in both spatial and temporal domains
- *Control variables* that represent external forces acting on a system.

Mathematical models are omnipresent in EWR fields, including surface hydrology, hydrogeology, environmental engineering, petroleum engineering, agriculture, ecology, and cross-disciplinary fields coupling two or more of these individual subjects. These EWR models are derived from the laws of mass, energy, and momentum conservation, empirical formulas, constitutive relationships, or statistical learning theory. They are also determined by appropriate subsidiary conditions and assumptions used during model simplification. Some examples are given below.

1.1.2 Examples of EWR Models

Example 1.1 *The Monod model of biomass growth*
Monod model is used for describing the growth of microorganisms (Bungay 1998)

$$\mu = \frac{\mu_{\max} S}{K_S + S}, \tag{1.1.1}$$

where μ is the specific growth rate, S is the nutrient concentration, $\mu_{\max}$ is the maximum specific growth rate, and K_S is the Monod coefficient. In this model, μ is the state variable, S is the control variable, and $\mu_{\max}$ and K_S are model parameters to be determined by experiments. In (1.1.1), the state variable is given explicitly by an algebraic equation.

Example 1.2 *A convolution integral model for spring discharge prediction*
The relationship between precipitation rate *p(t)* and spring discharge *q(t)* in a basin can be represented by the following model (Beven 2001)

$$q(t) = \int_0^t K(t-\tau)p(\tau)d\tau, \tag{1.1.2}$$

where the transfer function $K(\cdot)$ reflects catchment characteristics and can be obtained from historical observations of $p(t)$ and $q(t)$. In this model, $K(\cdot)$ is the model parameter, $p(t)$ is the control variable, and the state variable $q(t)$ is given by a convolution integral.

Example 1.3 *Lumped parameter rainfall–runoff model*
A lumped parameter rainfall–runoff model of a watershed can be expressed in the following general form (Sorooshian 2008; Beven 2001)

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, \boldsymbol{\theta}, \mathbf{p}), \tag{1.1.3}$$

where the watershed is divided into M subbasins, $\mathbf{u} = (u_1, \ldots u_M)^T$ is the state vector, with its ith component representing the average soil moisture content of the ith subbasin, $\mathbf{f}(\cdot)$ is a nonlinear operator representing the system transition over an instant of time, $\boldsymbol{\theta}$ is a set of system parameters, and $\mathbf{p}(t)$ is a forcing term whose ith component represents the average precipitation rate over the ith subbasin. Equation (1.1.3) is a set of ordinary differential equations (ODE), and an appropriate initial condition $\mathbf{u}(t)|_{t=0} = \mathbf{u}_0$ is needed for its solution. It can be considered as a discretized form of a distributed parameter rainfall–runoff model.

Example 1.4 *Channel flow model*
Water flow in a shallow river can be modeled by the Saint Venant equations, which assume 1-D flow, small bed slopes, and hydrostatic pressure distribution in the vertical direction (Chow 1959)

$$\frac{\partial h}{\partial t} + \frac{\partial (hv)}{\partial x} = q \tag{1.1.4}$$

and

$$\frac{\partial v}{\partial t} + g\frac{\partial h}{\partial x} + v\frac{\partial v}{\partial x} = F(v, h). \tag{1.1.5}$$

In (1.1.4), x is the longitudinal distance along the river, $h(x,t)$ is the water depth, $v(x,t)$ is the flow velocity, and $q(x,t)$ denotes the lateral inflow rate per unit length of the river. In (1.1.5), g is the acceleration of gravity; $F(v,h) = S_0 - S_f$, where S_0 is the riverbed slope, S_f is the energy slope defined as $v^2 / c^2 h$, and c is a roughness coefficient. The 1-D Saint Venant equations are nonlinear first-order PDEs that require appropriate auxiliary conditions to solve for $h(x,t)$ and $v(x,t)$.

The appropriate auxiliary conditions for (1.1.4) and (1.1.5) include initial conditions $h(x,0) = f_0(x)$ and $v(x,0) = g_0(x)$ at $t = 0$ and boundary conditions $h(0,t) = f_1(t)$ and $v(0,t) = g_1(t)$ at the upstream end $x = 0$, where f_0, f_1, g_0, and g_1 are known functions. This model consists of two coupled nonlinear PDEs with the state variables $h(x,t)$ and $v(x,t)$, control variable q, parameters c and S_0, and initial and boundary conditions.

Example 1.5 *Water quality control of a river segment*
Water quality in a river segment is affected by the quality of inflow water from its upstream boundary. Assume that the purpose of constructing a model is to manage the water quality along the river segment. If the river is relatively narrow and shallow, only the cross-sectional average concentration along the river segment needs to be known. In this case, we can use the following 1-D advection–dispersion–reaction model (Schnoor 1996):

$$\frac{\partial C}{\partial t} - \frac{\partial}{\partial x}\left(D \frac{\partial C}{\partial x} \right) + v \frac{\partial C}{\partial x} + RC = 0, \tag{1.1.6}$$

with initial and boundary conditions

$$C(x,t)\,|_{t=0} = C_0(x),\; C(x,t)\,|_{x=0} = C_{in}(t),\; \frac{\partial C}{\partial x}\,|_{x=L} = 0. \tag{1.1.7}$$

The governing equation (1.1.6) is a second-order parabolic PDE derived from the mass conservation and Fick's law. In this model, the cross-sectional average concentration distribution, $C(x,t)$, is the state variable, and the concentration of inflow water, $C_{in}(t)$, is the control variable. Other parameters and variables include the flow velocity v; the dispersion coefficient D, which represents not only the molecular diffusion, but also the effect of the small-scale turbulent flow; the linear reaction rate constant R; the length of the river segment L; and the initial concentration distribution C_0.

If the reaction is nonlinear, the reaction term will depend on the concentration. For uniform and steady-state flow, the velocity $v(x,t)$ in (1.1.6) can be measured. When the river flow is nonsteady and nonuniform, however, $v(x,t)$ must be calculated using the river flow model presented in Example 1.4.

Example 1.6 *Modeling groundwater flow in an unconfined aquifer*
Transient flow in an unconfined aquifer is governed by the following second-order parabolic PDE by invoking the Dupuit–Forchheimer approximation, which states

that water-level variation in the horizontal direction is negligible and head distribution in the vertical direction is hydrostatic (Bear 1979):

$$S_y \frac{\partial h}{\partial t} = \nabla\cdot[K(h-b)\nabla h] + Q_R - \sum_{i=1}^{N_w} W_i \delta(\mathbf{x}-\mathbf{x}_i),\ i = 1, 2, \cdots, N_w. \quad (1.1.8)$$

In (1.1.8), $h(x,y,t)$ is the water table elevation; W_i is the pumping rate from the ith well located at $\mathbf{x}_i$; N_w is the total number of wells; $\delta(\mathbf{x}-\mathbf{x}_i)$ is the Dirac delta function; Q_R is the net recharge rate per unit area; S_y is the specific yield; K is the hydraulic conductivity; and b is the bottom elevation of the aquifer. To solve the state variable $h(x,y,t)$ from (1.1.8), we must have the following information: control variables W_i and Q_R; system parameters S_y, K, and b; the geometry of flow region Ω; the initial condition $h(x,y,0) = f_0$ over Ω; and the boundary conditions $h|_{\Gamma_1} = f_1$ and $-K(h-b)\nabla h\cdot\mathbf{n}|_{\Gamma_2} = f_2$ along the entire boundary of Ω, where f_0, f_1, and f_2 are known functions, Γ_1 is the given water-level boundary condition, and Γ_2 is the given flux boundary section with $\mathbf{n}$ being its unit normal vector. In practice, the above-mentioned conditions and parameter values may not be known completely and exactly.

Example 1.7 *Cleanup of a contaminated aquifer*
Pump-and-treat is a remediation technology commonly used in the remediation of contaminated aquifers. To determine pumping locations and rates, we can couple a groundwater flow model, such as the one given in *Example 1.6,* with a 2-D advection–dispersion model (Bear 1979):

$$\frac{\partial \theta C}{\partial t} = \nabla\cdot[\theta \mathbf{D}\nabla C] - \nabla\cdot(\theta \mathbf{V} C) - S(C) \quad (1.1.9)$$

subject to initial and boundary conditions

$$C|_{t=0} = g_0,\ C|_{\Gamma_1} = g_1,\ -(\mathbf{D}\nabla C - \mathbf{V}C)\cdot\mathbf{n}|_{\Gamma_2} = g_2. \quad (1.1.10)$$

In this model, the state variable, $C(x,y,t)$, is the contaminant concentration in the aquifer; θ is the effective porosity; $\mathbf{V}(x,y,t)$ is the average linear velocity determined using the hydraulic head $h(x,y,t)$ solved from the flow model; $\mathbf{D}$ is the hydrodynamic dispersion coefficient (a tensor); and $S(C)$ is a sink/source term accounting for chemical reaction, decay, and adsorption processes. In the auxiliary conditions given in (1.1.10), g_0, g_1, and g_2 are known functions, and other notations are the same as those given in *Example 1.6.*

Hydrodynamic dispersion accounts for the effects of mechanical dispersion and molecular diffusion in porous media. Mechanical dispersion is caused by the difference between the average velocity in the macroscopic level and the distributed velocity in the microscopic level. For an isotropic porous medium, the dispersion coefficient $\mathbf{D}$ depends on the flow velocity $\mathbf{V}$, longitudinal dispersivity α_L, and transverse dispersivity α_T, according to

$$\begin{aligned} D_{11} &= \left(\alpha_L V_1^2 + \alpha_T V_2^2\right) / |V| + \tau D_d, \\ D_{12} &= D_{21} = (\alpha_L - \alpha_T) V_1 V_2 / |V|, \\ D_{22} &= \left(\alpha_T V_1^2 + \alpha_L V_2^2\right) / |V| + \tau D_d, \end{aligned} \tag{1.1.11}$$

where D_{11}, D_{12}, D_{21}, and D_{22} are elements of the tensor $\mathbf{D}$, V_1 and V_2 are components of $\mathbf{V}$ and $|V|$ is its magnitude, D_d is the molecular diffusion coefficient, and $0 < \tau < 1$ is a coefficient related to tortuosity of flow paths in the porous medium. Detailed derivation of the advection–dispersion equation (1.1.9) can be found in Bear (1979) and Sun (1996).

Major assumptions underlying (1.1.9) include the following: (i) Upscaling of the flow field from the microscopic level to the macroscopic level is valid; (ii) Fick's law is applicable for hydrodynamic dispersion; (iii) the Dupuit–Forchheimer assumption is valid; and (iv) a 2-D mass transport model is acceptable; if not, a 3-D coupled flow and contaminant transport model should be constructed.

When the control variables (i.e., the extraction locations and rates) are changed in the flow model, the flow field $\mathbf{V}$ and the dispersion tensor $\mathbf{D}$ in the mass transport model are changed accordingly, which cause the change in concentration distribution. The effect of pump-and-treatment can thus be predicted.

Example 1.8 *Large-scale emission management*
In large-scale air pollution modeling, the concentration distribution of a chemical compound is governed by a mass balance equation that takes account of the effects of advection, dispersion, deposition, emission, and chemical reactions (Zannetti 1990). If N_c chemical components are involved, the model consists of the following N_c-coupled PDEs:

$$\frac{\partial C_i}{\partial t} - \nabla \cdot (\mathbf{K} \nabla C_i) + \nabla \cdot (\mathbf{V} C_i) + k_i C_i - E_i + Q_i(C_1, C_2, \cdots, C_{N_c}) = 0, \quad (i = 1, 2, \cdots, N_c) \tag{1.1.12}$$

where $C_i(\mathbf{x}, t)$ is the concentration distribution of the ith chemical compound, $\mathbf{K}$ is a dispersion tensor that accounts for the effect of small-scale airflow, $\mathbf{V}$ is the average velocity of airflow at the large scale, k_i is the deposition rate of the *i*th compound, $E_i(\mathbf{x}, t)$ is the emission source strength, and Q_i is the function of chemical reactions between the ith compound and other compounds.

An important assumption used in deriving (1.1.12) is the so-called *K*-theory that is similar to Fick's law. It states that the large-scale dispersion flux, caused by the difference between the small-scale airflows and the average airflow at the large scale, is proportional to the concentration gradient, with the dispersion tensor $\mathbf{K}$ being the proportion coefficient. In the case of steady-state airflow, we can use a diagnostic model, for which the velocity $\mathbf{V}$ in the advection term is simply obtained by interpolation and extrapolation of meteorological measurements. When the purpose of model construction is to predict the development of a contaminant plume under transient airflow condition, a prognostic model is needed where the velocity $\mathbf{V}$ is obtained by solving a dynamic meteorological model.

The air pollution model consists of N_c -coupled, second-order parabolic PDEs and their auxiliary conditions. In this model, the N_c concentration distributions are state variables, the emission source strengths are control variables for managing the air quality, and others are system parameters that can be calculated by empirical formulae or estimated by data. The components of airflow velocity and dispersion tensor may contain significant uncertainty. ■

From the examples presented thus far, we can draw the following conclusions:

- The governing equations of a physics-based model are derived from fundamental physical and chemical laws and, thus, can be used in any case; however, the values of system parameters and auxiliary conditions are case dependent and need to be determined on a case-by-case basis.
- Different governing equations may be derived for the same problem when different assumptions are involved. Making assumptions and simplifications is essential in EWR modeling because we cannot construct a model that characterizes a EWR system exactly in all aspects and at all scales. In fact, the process of choosing appropriate assumptions and determining the appropriate level of model complexity is also case dependent and is oftentimes an art in EWR modeling.

As a solid example of the above conclusions, we see that models of mass transport in rivers (Example 1.5), porous media (Example 1.7), and air (Example 1.8) all share the same mathematical form, namely the advection–dispersion–reaction equation

$$\frac{\partial C}{\partial t} = \nabla\cdot[\mathbf{D}\nabla C] - \nabla\cdot(\mathbf{V}C) - R(C) + E, \tag{1.1.13}$$

which is subject to appropriate auxiliary conditions. However, the dispersion coefficient $\mathbf{D}$ has different physical explanations and magnitudes in different problems and the same for the reaction term $R(C)$ and sink/source term E. Equation (1.1.13) may become nonlinear when $\mathbf{D}$ or $\mathbf{V}$ is dependent on C (in the case of viscous flow) or when $R(C)$ is a nonlinear function.

1.1.3 General Form and Classification

Although different systems and models appear in different EWR fields, all of them can be expressed in the following general form:

$$\boldsymbol{\mathcal{L}}(\mathbf{u}, \mathbf{q}, \mathbf{p}, \mathbf{b}) = \mathbf{0}. \tag{1.1.14}$$

Equation (1.1.14) is called an operator equation (see Appendix A). It may consist of one or a set of algebraic equations, integral equations, ODEs, or PDEs that represent the quantitative relationships between state variables and other variables or parameters of the system under consideration. In (1.1.14), $\mathbf{u}$, $\mathbf{q}$, $\mathbf{p}$, and $\mathbf{b}$ are state variables, control variables, model parameters characterizing the internal properties of the system, and boundary parameters in both spatial and temporal domains, respectively. They can be scalars, vectors, functions, and vector functions.

A classification of mathematical models is provided below:

- *Single-state model* versus *multistate model,* depending on whether the number of state variables is one or more than one.
- *Linear model* versus *nonlinear model,* depending on whether or not all model equations are linear.
- *Deterministic model* versus *probabilistic* (or *stochastic*) *model,* depending on whether random variables appear in model equations.
- *Lumped parameter model* versus *distributed parameter model*, depending on whether or not spatially varying parameters are involved in model equations. When the spatial variability of a system can be ignored or averaged, the system can be regarded as a point and simulated by a lumped parameter model. A distributed parameter model is represented by PDEs, while a lumped parameter model is represented by ODEs.
- *Steady-state model* versus *transient model*, depending on whether or not the time variable is involved in model equations.
- *Physics-based model* versus *data-driven model*, depending on whether or not physically based parameters appear in model equations.

All model types listed in the above exist in EWR modeling, as we have already seen in Sect. 1.1.2. The state of a EWR system may be characterized by more than one correlated state variables, its structure is usually inhomogeneous, the relationships between its variables can be nonlinear, its state variables and parameters may depend on both locations and time, and its internal structure and external conditions are usually uncertain. As a result, (1.1.14) can be a highly complex mathematical model for a EWR system. This book focuses on the construction and calibration of such highly complex models.

1.1.4 Model Construction Process

A typical process of constructing an EWR model consists of the following steps (Fig. 1.1):

- *Data collection:* Collect all existing data that can provide information for model construction and calibration, such as measurements of state variables, control variables, and system parameters, and the knowledge of experts; site-specific field campaigns may be designed and conducted to acquire additional information.

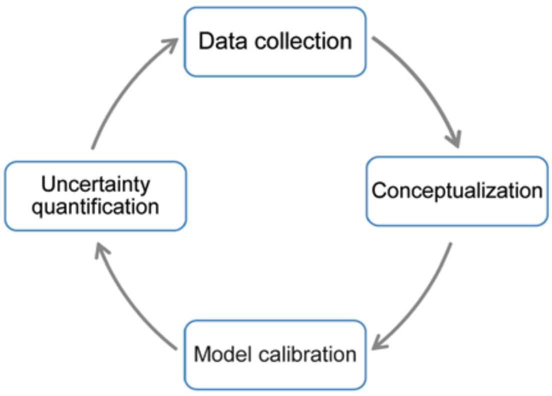


Fig. 1.1 Major steps of model construction

- *Conceptual model development:* Select an appropriate model structure and determine the model equations based on the underlying physical/chemical processes, analysis of the collected data, and the assumptions made for model simplifications.
- *Model calibration:* Adjust the structure of the conceptual model and identify its parameters so that model outputs can fit the collected data.
- *Uncertainty quantification:* Estimate the reliability of the constructed model when it is used for prediction and decision support.

Among the four steps, the data collection step is the basis of model construction. Without sufficient data, no one can construct a useful model. The conceptualization step is probably the most challenging step for a modeler. A model should be an appropriate simplification of a real system. If the model is oversimplified, important characteristics of the system may be lost, while if it is overcomplicated the model cannot be well calibrated because of data insufficiency. Therefore, the development of a conceptual model is regarded as an evolving process through which the conceptual model is adjusted to reflect newly gained knowledge. In the model calibration step, the model structure is corrected and model parameters are modified through fitting the model outputs to observed data. Model calibration is the key to successful modeling, but can be very challenging. A large fitting residual, of course, is unacceptable, but a small fitting residual does not necessarily mean that the model is acceptable. In the latter case, we can only say that the model cannot be rejected, and that is why we need the additional uncertainty analysis step to assess the reliability of the calibrated model before it is used for prediction and decision making. Assessing the reliability of a nonlinear and complicated model is another challenging problem in model construction. When the calibrated model is deemed to be unreliable, new data must be collected, and the above steps must be repeated. As a result, the model construction process forms a *closed loop* (see Fig. 1.1), as opposed to the *open-loop* process in which all steps are done only once. The construction of a useful EWR model can thus be both time-consuming and expensive.

As described in the remaining sections of this chapter, three common problems are solved repeatedly during a model construction process: the forward problem for simulation and prediction, the inverse problem for model calibration and parameter estimation, and the uncertainty quantification problem for reliability assessment.

1.2 Forward Solution

1.2.1 The Forward Problem

In the forward problem, the model equation (1.1.14) is solved to find the unknown system states $\mathbf{u}$ for given $\mathbf{q}, \mathbf{p}$ and $\mathbf{b}$. The forward solution forms the basis of model study. The general form of a forward solution can be represented by

$$\mathbf{u} = \mathcal{M}(\mathbf{q}, \mathbf{p}, \mathbf{b}), \tag{1.2.1}$$

where $\mathcal{M}$ is a mapping from $(\mathbf{q}, \mathbf{p}, \mathbf{b})$ to state variables $\mathbf{u}$. Mapping is an extended concept of function when its variables are functions (see Appendix A). Analogous to computer programming, $\mathcal{M}(\cdot)$ can be seen as a subroutine with $\mathbf{q}, \mathbf{p}$, and $\mathbf{b}$ as its inputs, and $\mathbf{u}$ as its outputs. The forward solution provides an explicit representation of the "excitation–response" relationship for the system being modeled.

In some simple models, such as the models in *Examples 1.1 and 1.3*, the forward solution is given explicitly. For most EWR models, however, state $\mathbf{u}$ appears in the model equation implicitly, and one or more equations must be solved to obtain the forward solution.

1.2.2 Solution Methods

The solution of the forward problem has been extensively studied in mathematics and engineering. Analytical and numerical methods for solving linear and nonlinear algebraic equations, integral equations, ODEs, and PDEs can be found in many mathematical textbooks for scientists and engineers (e.g., Polyanin and Manzhirov 2007; Hoffman 2001; Lapidus and Pinder 1982; Beven 2012; Celia and Gray 1992; Sun 1996; Helmig 1997). In most cases considered in this book, an EWR model is a distributed parameter model expressed by one or more PDEs (i.e., $\mathcal{M}$ is physically based). Typical PDEs appeared in EWR modeling include the elliptic PDEs used to describe steady-state flow processes, the parabolic PDEs used to describe transient flow and transport processes, the second-order hyperbolic PDEs used to describe wave or oscillation processes, and the first-order hyperbolic PDEs used to describe pure advection processes. Analytical and numerical methods for solving PDEs can be found in many books on EWR modeling (Bear 1979; Sun 1996; Zlatev 1995;

Lynch 2005; Batu 2006; Szymkiewicz 2010; Kolditz 2002; Maidment 1993; Abbott and Refsgaard 1996; Chen 2007; Bear and Cheng 2010; Singh 2012).

This book also considers data-driven models, in which the model mapping $\mathcal{M}$ is developed based solely on the information content of observed data. Data-driven models mainly involve algebraic equations, and usually, no specialized forward solution method is needed (Bishop 2006).

Over the years, many specialized software packages have been developed in different EWR fields for solving the relevant forward problems (http://www.scientificsoftwaregroup.com). For better understanding of the contents of this book, we provide a brief review below on the commonly used methods for obtaining forward solutions of physics-based models.

1.2.2.1 Analytical Methods

An analytical solution is a solution of the forward problem expressed by known functions or through some operations on them (series or integrals). Superposition of fundamental solutions, separation of variables, Laplace, Fourier, and other transforms are the commonly used techniques to obtain analytical solutions. Analytical solutions can be found only for idealized models where the model parameters are constant and model geometry is simple. Thus, they are most useful for theoretical studies, simple parameter identification problems, and testing of newly developed numerical methods and codes.

Example 1.9 *Analytical solution for the canonical 1-D mass transport model*
When D and v in (1.1.6) are constant, and $C_0 = 0$, $C_{in} = const$, $L = \infty$ in (1.1.7), the model in Example 1.5 reduces to the 1-D canonical mass transport problem for which the following analytical solution can be obtained by Laplace transform:

$$C(x,t) = \frac{C_{in}}{2}\exp\left(\frac{vx}{2D}\right)\left\{\exp\left[-\frac{x}{2D}\sqrt{v^2+4RD}\right]\operatorname{erfc}\left[\frac{x-\sqrt{v^2+4RDt}}{2\sqrt{Dt}}\right]\right.$$
$$\left.+\exp\left[\frac{x}{2D}\sqrt{v^2+4RD}\right]\operatorname{erfc}\left[\frac{x+\sqrt{v^2+4RDt}}{2\sqrt{Dt}}\right]\right\}. \tag{1.2.2}$$

This solution provides an explicit expression of the relationship between the system state, control variable, and system parameters. For example, it shows that the concentration distribution is proportional to the concentration of inflow water. Detailed derivation of this solution and other analytical solutions for mass transport in open channels and aquifers can be found, for example, in Chow et al. (1988) and Sun (1996).

1.2.2.2 Discretization Methods

Most real-world models do not have analytical solutions because of their irregular geometries, complex boundary conditions, and inhomogeneous parameters. As a result, numerical methods are widely used to find approximate solutions. The often used numerical methods for solving PDEs include the finite difference method (FDM), the finite element method (FEM), the finite volume method (FVM), and their variations. These methods are all based on discretizing the spatial domain into a finite number of elements (or cells) and the temporal interval into a finite number of steps. After discretization, all infinite-dimensional functions (both known and unknown) are approximated by finite-dimensional vectors. For a dynamic model, a PDE is first reduced to a set of ODEs after the spatial discretization, and the set of ODEs is further reduced to a set of algebraic equations after the temporal discretization.

Example 1.10 *Numerical solution of the mass transport model*
To solve the advection–dispersion–reaction equation (1.1.13), a discretization method consists of the following common steps:

1. Spatial discretization. The spatial domain is partitioned into M elements with N nodes (when FEM is used). The concentration distribution $C(x, y, z, t)$ is replaced appropriately by a set of nodal concentration values

$$\mathbf{C}(\mathbf{t}) = [C_1(t), C_2(t), \cdots, C_N(t)]^T. \tag{1.2.3}$$

   The PDE is then reduced to a set of ODEs

$$\mathbf{B}\frac{d\mathbf{C}}{dt} + \mathbf{AC} = \mathbf{f}, \tag{1.2.4}$$

   where the coefficient matrices $\mathbf{A}$, $\mathbf{B}$, and the right-hand-side vector $\mathbf{f}$ depend on parameters $\mathbf{D}, \mathbf{V}$, R, and E, boundary conditions, and the method used for spatial discretization.
2. Time discretization. The time interval $[0, T]$ is partitioned into K time steps $0 = t_0 < t_1 < \cdots < t_n < t_{n+1} < \cdots < t_K = T$ and let $\mathbf{C}_n$ be the nodal value vector at time t_n. Using finite difference approximation to replace the time derivative in (1.2.4), we have

$$\left(\frac{\mathbf{B}}{\Delta t_n} + \alpha\mathbf{A}\right)\mathbf{C}_{n+1} = \left(\frac{\mathbf{B}}{\Delta t_n} - (1-\alpha)\mathbf{A}\right)\mathbf{C}_n + \mathbf{f}, \tag{1.2.5}$$

where $\Delta t_n = t_{n+1} - t_n$ and $0 \le \alpha \le 1$, with $\alpha = 0$ for explicit scheme and $\alpha = 1$ for implicit scheme.

3. Solving a system of algebraic equations. Let matrix $\mathbf{H} = \mathbf{B} / \Delta t_n + \alpha\mathbf{A}$ and vector $\mathbf{b} = (\mathbf{B} / \Delta t_n - (1-\alpha)\mathbf{A})\mathbf{C}_n + \mathbf{f}$. At the beginning, $\mathbf{C}_0$ is obtained from the initial condition. At $t = t_n$, $\mathbf{C}_n$ is known and the solution of algebraic equations (1.2.5) gives

$$\mathbf{C}_{n+1} = \mathbf{H}^{-1}\mathbf{b}. \tag{1.2.6}$$

Applying (1.2.6) repeatedly for $n = 1, 2, \cdots, K$, we can obtain the discrete forward solutions $\mathbf{C}(t_1), \mathbf{C}(t_2), \cdots, \mathbf{C}(t_K)$ for all time steps.

When the governing equation (1.1.13) is nonlinear (e.g., R depends on the unknown concentration), (1.2.6) becomes a set of nonlinear algebraic equations. In this case, an iterative solver is needed at each time step to obtain solutions (Lapidus and Pinder 1982).

1.2.2.3 Particle Tracking Methods

FDM, FEM, FVM, and their variations belong to *the Eulerian methods* as they use a spatially fixed coordinate system to find the solution. For advection-dominated mass transport problems, Eulerian methods are known to produce inaccurate results due to *numerical dispersion* and *overshooting* (Celia and Gray 1992). Particle tracking is a kind of Lagrangian method that directly simulates the movement of a large amount of mass particles in the flow field, instead of solving the advection–dispersion PDE (Delay et al. 2005; Zhang et al. 2000; Dagan et al. 1992; Pollock 1988).

Finite cell method (Sun 1999, 2002) and smooth particle hydrodynamics (Liu and Liu 2010; Tartakovsky et al. 2007) can be used to incorporate various physical, chemical, and biological processes into particle tracking. After the concentration distribution $\mathbf{C}_n$ at time step t_n is known ($\mathbf{C}_0$ is given as the initial concentration distribution), the concentration distribution $\mathbf{C}_{n+1}$ at time step t_{n+1} is calculated explicitly by

$$\mathbf{C}_{n+1} = \mathcal{F}(\mathbf{C}_n) \tag{1.2.7}$$

where $\mathcal{F}(\cdot)$ is an operator that directly simulates advection, dispersion, attachment, mechanical interaction, chemical reaction, growth, and decay processes using a set of particles. It depends also on model parameters, sink/sources, and boundary conditions. The solution of complicated PDEs is replaced by calculating the operator $\mathcal{F}(\cdot)$ in each time step.

1.2.3 Well-Posedness of the Forward Problem

A mathematical problem is said to be well-posed in the sense of Hadamard if it satisfies the following three conditions (Hadamard 1902):

1. *Existence:* The problem has a solution.
2. *Uniqueness:* The problem does not have a different solution.
3. *Stability:* The solution is continuously dependent on data used to determine the solution.

It has been shown in mathematics that the forward problem of various EWR models is well-posed, provided that appropriate auxiliary conditions are prescribed. Different types of auxiliary conditions are required for different types of ODEs and PDEs.

Note that the numerical solution of a forward problem is not unique because of the numerical error. Different numerical solutions may be produced for the same problem when different numerical methods or schemes of discretization are used. This fact, however, does not contradict with the uniqueness of the forward problem because all of these solutions converge to the unique accurate solution when the grid resolution and time steps are made sufficiently small. When a software package is used to solve a forward problem, we should make sure that the numerical solution is convergent and stable and the discretization error is within an acceptable range.

1.3 Model Calibration and Parameter Estimation

1.3.1 Data Availability

Successful model calibration and parameter identification depend on the quantity and quality of available data at a study site. For EWR modeling, the following types of data should be collected:

- *Prior information.* Examples of prior information include previously developed models for the site (or for similar sites), land use and land cover characteristics, soil properties, geological structures, and expert knowledge about the site. Prior information generally contains significant uncertainty and needs to be transferred to quantitative expressions, such as guessed values and ranges of the parameters under estimation.
- *Measurements of the unknown parameters*. Physical parameters can be measured either in the laboratory or in the field. A distributed parameter, of course, can only be measured at limited locations and times. Note that the measured value of a parameter may not be appropriate to be used directly in a model when the parameter is scale dependent.
- *Observations of state variables.* This type of data can be obtained from historical records and/or designed field experiments. Measuring state variables (such as pressure, temperature, concentration, and streamflow rate) is relatively easier than measuring parameters (such as hydraulic conductivity and dispersivity). Observation error is often assumed to be normally distributed with zero mean.
- *Model application data.* This type of data typically include the objectives of model construction (or the goals of model application) and their accuracy/reliability requirements and can be elicited from a manager or a decision maker.

1.3.2 Methods and Criteria

Depending on what types of data are available and how they are used, different approaches are developed for model calibration and parameter estimation. Let us consider several cases:

- Estimating a parameter with only measurements of the parameter. When the unknown parameter is a deterministic function of spatial and/or temporal variables, various interpolation and approximation methods can be used. Incorporating prior information may help to reduce the errors of interpolation, extrapolation, and approximation. When the unknown parameter is regarded as a realization of a random field, statistical interpolation or approximation methods are needed.
- Estimating a model parameter with observations of state variables. In the absence of model error, this problem is a *CIP*, which aims to infer model inputs based on model outputs. Incorporating prior information and measurements of the parameter can help stabilize the inverse solution.
- Modifying a model structure with prior information and observation data. Model structure error in EWR models can never be avoided because a model is always a simplified representation of the underlying physical system. In this case, we need to solve a so-called EIP to reduce the model error through modifying the model structure, assuming that the information provided by available data allows us to do so. Identifying a model structure is actually a system identification problem.
- Incorporating the model application data into the process of model calibration and parameter estimation to determine the model complexity and data sufficiency. This problem is called the GIP, in which the model to be constructed is determined not only by observation data and prior information, but also by model application data.

All of these methods for model calibration and parameter estimation are based on the following three criteria:

(C-1) The model outputs should fit the observed data as close as possible.

(C-2) The prior information should be used as much as possible.

(C-3) The model should be reliable for the given goals of model application.

The CIP and EIP are formulated using both (C-1) and (C-2), and the GIP is formulated by all the three criteria. In the remaining sections of this chapter, we will introduce more concepts and ideas on various inverse problems and give detailed explanations of the three criteria.

1.3.3 The Inverse Problem

A model establishes a relationship between state variables $\mathbf{u}$ and other variables $(\mathbf{q}, \mathbf{p}, \mathbf{b})$. If there is no model error, a model can be used in two directions: the forward problem solves $\mathbf{u}$ from the model when $(\mathbf{q}, \mathbf{p}, \mathbf{b})$ are given, while the inverse problem solves $(\mathbf{q}, \mathbf{p}, \mathbf{b})$ from the model when observations of $\mathbf{u}$ are available. The "excitation–response" relationship of the forward problem is reversed in the inverse problem. In EWR modeling, we may see the following types of inverse problems:

- *System identification:* identifying the true or a representative structure of a system and associated structure parameters
- *Parameter identification:* identifying continuous or discrete model parameters that may or may not have a physical meaning
- *Sink/source identification:* identifying distributed or point sources (or sinks) that may vary with time and/or location
- *Auxiliary condition identification:* determining the initial condition, boundary geometry, and boundary conditions
- *Stochastic variable identification:* estimating statistics of a random variable or a random function

1.3.3.1 Ill-Posedness of Inverse Solution

In a model construction process, various inverse problems are solved either iteratively or simultaneously. Let $\boldsymbol{\theta}$ be a part of model input parameters (variables) that need to be identified and $\boldsymbol{\varsigma}$ be all other model input parameters (variables) that are known. Without showing $\boldsymbol{\varsigma}$ explicitly, model equation (1.1.14) is rewritten as

$$\mathcal{L}(\mathbf{u}, \boldsymbol{\theta}) = \mathbf{0}. \tag{1.3.1}$$

Solving $\mathbf{u}$ from (1.3.1), we obtain the forward solution

$$\mathbf{u} = \mathcal{M}(\boldsymbol{\theta}). \tag{1.3.2}$$

A question is thus raised: can we solve $\boldsymbol{\theta}$ from the same equation (1.3.1) to obtain the inverse solution? To address this question, we first note that $\mathbf{u}$ and $\boldsymbol{\theta}$ have different positions in the model equation. As a result, the same model may have different types of equations when used for forward solution and inverse solution. This point is further elucidated using the following example.

Example 1.11 *Identifying the dispersion coefficient $D(x)$ in Example 1.5*
In model equation (1.1.6), when $C(x,t)$ is known and $D(x)$ is the unknown function, the same equation becomes

$$\alpha \frac{dD}{dx} + \beta D = \gamma, \tag{1.3.3}$$

where

$$\alpha = \frac{\partial C}{\partial x}, \ \beta = \frac{\partial^2 C}{\partial x^2}, \ \gamma = \frac{\partial C}{\partial t} + v\frac{\partial C}{\partial x} + RC. \tag{1.3.4}$$

Because $C(x,t)$ is known, the coefficients α, β, and γ can be determined. Now, we see that model equation (1.1.6) is a second-order parabolic PDE for solving $C(x,t)$ in the forward problem, but it turns into a first-order ODE for solving $D(x)$ in the inverse problem. Solving (1.3.3) for $D(x)$, we get

$$D(x) = \frac{\gamma}{\beta} + c\exp[-(\beta / \alpha)x], \tag{1.3.5}$$

where c is an arbitrary constant. Although the inverse solution (1.3.5) looks simple, it is actually useless because of its ill-posed nature. First, the solution is not unique because it contains an arbitrary constant c. To determine the constant, we need to know the value of $D(x)$ at a point in $[0, L]$, but this condition is not given in the model. Second, the solution is unstable because a small change in $C(x,t)$ may cause a large change in the derivatives of $C(x,t)$ which, in turn, may cause the coefficients α, β, and γ to become large or even unbounded. ■

Using a numerical method, we can reduce the model equation to a set of algebraic equations, but the ill-posed nature of the inverse problem cannot be eliminated, as it is shown by the following example.

Example 1.12 *Solving the inverse problem from a numerical forward solution*
Let us return to Example 1.10. After discretization, the unknown parameters become an *M*-dimensional vector $\boldsymbol{\theta}$, and its components appear in the coefficient matrix $\mathbf{H}$. By treating $\boldsymbol{\theta}$ as the unknowns and $\mathbf{C}_{n+1}$ as the known data, we can rewrite the forward solution equation (1.2.6) as

$$\mathbf{G}\boldsymbol{\theta} = \mathbf{d}, \tag{1.3.6}$$

where $\mathbf{G}$ is an $N \times M$ matrix and $\mathbf{d}$ is the right-hand data vector depending on $\mathbf{u}$. When $\mathbf{G}$ is not a square matrix, the system of equations (1.3.6) may be either overdetermined ($N > M$) or underdetermined $(N < M)$. In the former case, the inverse solution does not exist because no $\boldsymbol{\theta}$ can satisfy all equations, while in the latter case, the inverse solution is nonunique. When $\mathbf{G}$ is a square matrix $(N = M)$, (1.3.6) is ill conditioned because $\mathbf{G}$ is nearly singular in this case (Sun 1994). An ill-conditioned linear system is unstable. ■

Solving $\boldsymbol{\theta}$ from the model equation (1.3.1) is called the *direct method* of inversion. The previous discussion illuminates the inherent difficulty of direct inversion. Fortunately, there is another way called the *indirect method* of inversion that solves the unknown $\boldsymbol{\theta}$ through inverting the forward model (1.3.2). The following trial-and-error method provides the primitive form of indirect inversion.

1.3.3.2 The Trial-and-Error Method

The *trial-and-error method*, which is based on both criteria (C-1) and (C-2), has been widely used in practice for model calibration. To use this method, we only need (i) some observation data of the state variables, (ii) a computer code that solves the forward problem, and (iii) prior information of the identified parameters.

The trial-and-error method consists of the following steps:

1. Set an initial guess $\boldsymbol{\theta}_0$ based on prior information.
2. Use $\boldsymbol{\theta}_0$ as the inputs to the forward model to generate the model-calculated system state.
3. Compare the model-calculated system state with the observed data. If the difference between the two is smaller than a user-defined tolerance level, then stop; otherwise, proceed to the next step.
4. Use a new guess $\boldsymbol{\theta}_1$ to replace $\boldsymbol{\theta}_0$ and go back to step 2.

In the above process, steps 1 and 4 are completed by a domain expert who knows how to guess a reasonable $\boldsymbol{\theta}_0$ from the prior information and how to modify it to make the model outputs closer to the observed values. The advantages of the trial-and-error method are as follows: (1) it only requires a numerical code or a software package for the forward solution; (2) it can be used not only for parameter identification, but also for model structure calibration; and (3) the knowledge and expertise of an expert can be readily incorporated into the solution process.

Disadvantages of the trial-and-error method are as follows: (1) it is time consuming and cannot handle relatively complex models with a large number of unknown parameters; (2) the best solution cannot be obtained; and (3) different modelers may obtain different results. Nevertheless, the trial-and-error method has provided the rudimentary idea for more sophisticated inversion theories.

1.3.3.3 Classical Inverse Problem

The CIP assumes no model structure error and has been extensively studied in mathematics. The assumption of no model structure error implies that the model has the same "excitation–response" relationship as the physical system being modeled. Thus, when the model variables and parameters (i.e., model inputs) are correctly assigned, the model calculated states (i.e., model outputs) would coincide with the observed system states if there is no observation error. When observation error exists, however, the accurate inverse solution can never be obtained.

Theoretical studies on CIP have been conducted in both deterministic and statistical frameworks. In the deterministic framework, the classical theory of inverse solution is concerned about how to define approximate solutions, how to find the best approximate solution, and how to make the inverse solution unique and stable. Like the trial-and-error method, the formulation of CIP is based on criteria (C-1) and (C-2). The use of prior information can help make the inverse solution unique and stable.

In the statistic framework, the estimated parameter is regarded as a random variable (field), and the true parameter is merely a realization of the random variable (field). The initial probability distribution of the estimated parameter is determined by prior information and is called the prior probability distribution, which may contain large uncertainty. The inverse solution is a process that transfers information from data to the estimated parameter. The relationship between them is established by the model, and the information transfer is completed through the Bayes' theorem. After information transfer, the probability distribution of the estimated parameter (called the posterior distribution) is expected to be more concentrated, and thus, the uncertainty is reduced. In the statistical framework, the inverse solution always exists and the uniqueness is not required.

In Part I of this three-part book (Chaps. 2–5), we discuss solution of CIP in both deterministic and stochastic frameworks. Basic concepts of CIP and methods on linear model inversion and single-state nonlinear model inversion are given in Chap. 2; singular value decomposition and various nonlinear optimization algorithms are introduced briefly as the solution methods of CIP. In Chap. 3, CIP and multistate model inversion are cast into a multiobjective optimization problem and solved by the evolutionary algorithms. Regularization is also introduced in this chapter from the point of view of multiobjective optimization. It can be seen as a general method for stabilizing the inverse solution. The inverse problem is reformulated and resolved in the statistical framework in Chap. 4. Monte Carlo-based sampling methods for finding the posterior distribution are introduced. Various methods of model differentiation are given in Chap. 5. Model differentiation is a necessary tool for almost all topics related to model construction covered in this book, such as nonlinear optimization, local and global sensitivity analysis, model reduction, model structure identification, nonparametric model calibration, reliability analysis, optimal experimental design, etc. Both the forward and reverse modes of automatic differentiation are introduced.

1.3.4 Model Structure Identification

A mathematical model of a complex system is a simplified representation of the system with certain assumptions. As a result, the model structure error cannot be avoided and may be significant. This is especially true for EWR models because the structure of a EWR system is usually inhomogeneous and exhibits multiscale variability. Unlike observation error, model structure error is a type of systematic error, and its probability distribution is generally non-Gaussian. When solving CIP with an erroneous model structure, we may find that the fitting residual cannot be reduced to a satisfactory level, no matter how the model parameters are changed. At this point, we should consider reducing the model structure error. Because prior information and observation data contain the system structure information, calibration of model structure is possible.

We will use $(\mathbf{S}, \boldsymbol{\theta})$ to represent a model, where $\mathbf{S}$ represents the model structure and $\boldsymbol{\theta}$ denotes the model parameters associated with the structure. The problem of identifying both the model structure and its associated parameters is called the EIP (Sun and Sun 2002). Strictly speaking, EIP is not an inverse problem in the original sense (i.e., swapping the positions of inputs and outputs of a model). Solving EIP can significantly decrease the fitting residual because of the reduction of model structure error. Many modelers may have already solved EIP via the trial-and-error method by adjusting not only the model parameters, but also the model structure.

The dimension (degrees of freedom) of a distributed parameter (e.g., the hydraulic conductivity of a heterogeneous aquifer) is very high, or even infinite. It cannot be identified by limited data. Parameterization is a method for representing a distributed parameter by a low-dimensional vector. Parameterization can be seen as a model and, more often, a simplified model of a distributed parameter. A distributed parameter, if its structure is unknown, may be represented by different parameterizations with different complexity and patterns (i.e., by different models). In this case, the identification of a distributed parameter becomes an EIP or an adaptive parameterization problem.

Although EIP can be formulated by the same criteria (C-1) and (C-2) used in the CIP, solving an EIP is much more challenging than solving a CIP because it needs the solution of a min–min optimization problem. Moreover, when solving an EIP, the existing data are often insufficient to differentiate between two or more different model structures. In other words, the combination of different model structures and model parameters can fit the existing data equally well and that partly explains why different modelers may come up with different models for the same study site. When the true structure complexity of a system is unknown, discussion on the uniqueness of an EIP solution does not make much sense because data can never be sufficient to differentiate all possible structures.

Unlike the physics-based models, data-driven models do not rely on physical representation of the system; rather, they rely almost exclusively on the information content embedded in the training datasets. The structure and parameters of a data-driven model usually do not possess physical meanings, other than serving as a black-box representation of the actual physical phenomena. Data-driven models are most useful when there is no or little a priori knowledge about the structure of the true system or when it is desirable to replace a physics-based model for expediting computation. Solving EIP is an important topic for constructing both physics-based and data-driven models.

Part II of this book (Chaps. 6–8) is dedicated to the solution of EIP in both deterministic and statistical frameworks. In Chap. 6, various methods for parameterizing a deterministic function or a random field are introduced. Principal component analysis (PCA) and other linear and nonlinear dimension reduction methods, as well as their applications to inverse solution, are covered in this chapter. Model structure identification and hyperparameter estimation are the main topics of Chap. 7, in which various adaptive parameterization approaches, the level set method, multiscale inversion, and geostatistical inversion are covered. Methods for constructing data-driven models and machine learning are introduced in Chap. 8.

1.4 Model Reliability

1.4.1 Model Uncertainty Analysis

Using the traditional process of model construction described in Sect. 1.1.4, a model obtained by solving either CIP or EIP is such one that "cannot be rejected" by the existing data but "may not be acceptable" for prediction or other purposes of model application, as it is tested only by one or a limited number of excitations. Keep in mind that not all models are useful. Even a calibrated model may contain significant uncertainty in its structure, boundary conditions, and identified parameters. When model uncertainties and the inherent randomness of the modeled system are mixed, the assessment of model reliability could become extremely difficult. Uncertainty analysis attempts to find all possible unrejectable models and then assesses the relative reliability or the risk of using the identified model. When the risk level is unacceptable, new data must be collected and the model construction process must be repeated.

In recent years, various Bayesian- and Monte Carlo-based methods have been used to analyze the model uncertainty caused by uncertainty in estimated parameters (e.g., Beven and Freer 2001; Vrugt et al. 2005; Makowski et al. 2002; Wang and Zabaras 2005; Oliver et al. 2008; Gupta et al. 2012; Doucet et al. 2000). In the statistical framework, the uncertainty of model prediction can be assessed through sampling the posterior distribution of estimated parameters, which is a challenging problem by itself. When the number of parameters increases, the number of required samples can become very large. A common mistake in EWR modeling is that the sampling process is terminated prematurely before convergence is reached. When EIP is solved for model identification, sampling methods may become very ineffective because the number of candidate structures may be very large or even infinite. When a Monte Carlo-based method is used, the computational cost is generally very high. Up to date, the model uncertainty caused by model structure error has been considered only by a few researchers in the field of EWR modeling (e.g., Duan et al. 2007; Neuman 2003; Gupta et al. 2005; Ye et al. 2004; Sun et al. 1998). Therefore, more studies are needed on this challenging subject.

1.4.2 Problems and Difficulties in Model Construction

1.4.2.1 Scale Dependence Problem

A EWR system, such as an aquifer or a watershed, is naturally heterogeneous at multiple scales. The measured system states (e.g., pressure, concentration, soil moisture content, and temperature) and system parameters (e.g., hydraulic conductivity, land cover, soil properties, dispersion coefficient, and reaction rate) may be associated with different scales depending on the measurement equipment and measurement method. In a numerical model, all state variables and parameters must be upscaled or downscaled to the same discrete scale of the numerical model (the grid size). Unfortunately, without additional assumptions and constraints, the number of possible solutions of downscaling is infinite. The upscaled value of a parameter, such as the hydraulic conductivity, depends not only on the small-scale distribution of the parameter, but also globally on the control variables and boundary conditions. Moreover, the discrete scale of a model may vary with the accuracy requirement of model application and the computational capacity. As a result, the scale used for solving the inverse problem may be different from that used for solving the forward problem. Dealing with the scaling problem is thus crucial in EWR modeling.

1.4.2.2 Model Complexity Problem

Determining the complexity of a EWR model is a very difficult problem for a modeler. In practice, most modelers solve CIP to estimate the model parameters using a fixed model structure. If the fitting residual is large, the structure of the conceptual model will be adjusted and the model parameters are identified again. Using EIP, the model structure is automatically determined by available data. If we have more data, a complex structure can be identified; if we have less data, only a simple model can be identified. The resulting model complexity, of course, is generally inappropriate because the complexity of a useful model depends on not only the complexity of the modeled system, but also the goals of model application. Ideally, we would like to build a model that has the same structure as the real system being modeled. Unfortunately, making lifelike models for EWR systems is neither feasible nor necessary, at least in the foreseeable future. In practice, the structure of a useful EWR model should be simple enough so that it can be calibrated with limited data; at the same time, it should be complex enough so that the most important characteristics of the system are captured by the model. But, *how complex is complex*? This is a very difficult question for a EWR modeler. Most failure cases of EWR model applications are caused by the use of models with an inappropriate level of complexity, either overcomplicated or oversimplified.

1.4.2.3 Data Sufficiency Problem

Modelers often complain about the insufficiency of data for constructing a reliable model. But*, how sufficient is sufficient*? This question cannot be answered before an appropriate model complexity is known because the identification of a more complex model structure needs the support of more data.

The optimal data collection strategy, or the optimal experimental design (OED), should provide the maximal information content for model calibration with the minimal cost. There are two inherent difficulties: first, for nonlinear model calibration, OED depends on the identified parameters, which are unknown at the design stage, and second, the computational effort of OED may become unaffordable if the model structure also needs to be identified. Moreover, if we cannot even determine how many data are sufficient, the optimal design problem does not make much sense.

1.4.2.4 Reliability Assessment Problem

A model can transform the excitation–response relationship of a system into an input–output relationship of a computer code. This functionality makes the modeling method an indispensable tool for scientists and engineers. Optimal management of water resources, prediction of contaminant plume locations, natural hazard forecasts, recovery of contaminant source histories, and feasibility study for decision making are all typical applications of EWR modeling. But, we must assure the *reliability of a model* before it is used in practice. For example, when a model is used for decision making, we should make sure that the model can correctly predict the system states for all feasible scenarios.

The goal of uncertainty analysis is to answer the reliability problem of a constructed model. When the system structure is complex and unknown, however, the existing uncertainty analysis methods may not be effective because the number of nonrejectable model structures is too large to handle.

In its original sense, a model of a system should give the same response (output) as the actual system does for any excitation (input). Unfortunately, constructing such an absolutely and universally reliable model for a EWR system is generally beyond reach. Due to the complexity of system structure and the insufficiency of observation data, we are unable to construct a lifelike model, and we cannot assess "how complex is complex" for the model structure and "how sufficient is sufficient" for the existing data. In practice, we can only find such a model that is tested by limited data and has a relative reliability. But, *how reliable is reliable*?

1.4.3 Goal-Oriented Modeling

This problem, fortunately, can be answered by the project manager or the decision maker. The answer actually is criterion (C-3) of model construction criteria listed under Sect. 1.3.2 (i.e., the constructed model should be reliable for the predetermined goals of model application). It does not require a constructed model to be absolutely reliable, but the model must be reliable (satisfying certain accuracy requirement) for the specified model applications. For example, a modeler may be asked to construct a mass transport model for predicting the arrival time of a contaminant plume to a specified location with a prediction error less than 10 days.

It is different from the traditional process of model construction, where criteria (C-1) and (C-2) are used to formulate the inverse problem and criterion (C-3) is only tested after inverse solution. In contrast, the GIP directly incorporates criterion (C-3) into its formulation as an objective or a constraint. A typical goal-oriented model construction process is as follows: (i) The goal of model application is used to determine an appropriate level of model complexity; (ii) the so-determined model complexity is then used to determine the data sufficiency; and (iii) if the existing data are insufficient, a robust design is used to guide the new data collection. From this procedure, we can see that solution of GIP is not simply a data-fitting problem, and it actually integrates all components of model construction. Model reliability, which is at the center of GIP, is used to determine the model complexity, the model complexity is used to determine the data sufficiency, and the data sufficiency is then used to guide the experimental design for new data collection. The computational effort of solving GIP, however, is huge. Solving a GIP requires the solution of a series of EIP, and solving an EIP requires the solution of a series of CIP.

Part III of the book (Chaps. 9–12) contributes to the topic of model reliability. Chapter 9 introduces various data assimilation methods that allow a model to be updated continuously to improve its reliability whenever new data become available. Methods used for uncertainty quantification are introduced systematically in

Chap. 10. The effects of model parameter uncertainty and model structure uncertainty on model outputs are assessed. To construct a more reliable model, we need more data. Design of informative and cost-effective data collection strategies is the subject of Chap. 11, in which OED is formulated into a multiobjective optimization problem. The criteria of optimal design for linear model inversion are derived. For nonlinear model inversion, Bayesian and robust design methods, especially interval-identifiability-based robust design, are introduced. In Chap. 12, after the goal-oriented forward problem is described, GIP is formulated and solved in both the deterministic and statistical frameworks. When the existing data are insufficient, a cost-effective experimental design method for GIP is given. Finally, the goal-oriented, pilot-point method described in the last section of Chap. 12 is offered as a promising approach for developing distributed parameter models.

1.5 Review Questions

1. Classify the models in Examples 1.1–1.9.
2. Use an example in your study field to show how a conceptual model is constructed.
3. If the information on $\mathbf{q}, \mathbf{p}, \mathbf{b}$ is incomplete, can we say: (a) the forward problem may be ill-posed, or (b) the forward problem must be ill-posed?
4. Find or develop a code for solving the model given in Example 1.5 numerically and verify the accuracy of the numerical solution using the analytical solution given in (1.2.2).
5. Assume that concentration $C(x,t)$ is known in (1.1.6), solving the decay rate R from the equation. Is the problem well-posed?
6. Compare the similarities and differences between CIP and EIP?
7. Why the model complexity determined by solving CIP and EIP may not be appropriate?
8. Why the data sufficiency problem cannot be answered without knowing the model complexity?
9. In your area of study, what types of data are available for model calibration? How is the model structure determined?

Chapter 2
The Classical Inverse Problem

The classical inverse problem (CIP) defined in Chap. 1 assumes no model structure error; thus, only model parameters need to be identified. In this chapter, we consider the solution of CIP for single state variable models. Parameter identification of multistate variable models will be considered in the next chapter. When a single state variable u and the unknown parameter θ are functions of space and/or time, the model equation $\mathcal{L}(u, \theta) = 0$ is an operator equation and its forward solution $u = \mathcal{M}(\theta)$ is a mapping from parameter space $\mathbb{P}$ to state space $\mathbb{U}$. Basic concepts and terminologies of functional analysis, such as function space, operator, mapping, and norm, are provided in Appendix A.

In Chap. 1, we showed that CIP is a typical ill-posed problem. An accurate solution of CIP does not exist when observation error is present. Even in the absence of observation error, the solution of CIP may still be nonunique and unstable. Section 2.1 introduces the basic concepts and theory of CIP, as well as the definition of extended well-posedness. Using the "fitting data" criterion of inverse problem formulation, we can obtain a quasi-solution by solving an optimization problem. We will show that when the inverse problem is extended well-posed and the conditions of quasi-identifiability are satisfied, the quasi-solution will approach the accurate inverse solution when the observation error reaches zero.

Linear inversion theory is relatively well established. Section 2.2 introduces methods for obtaining pseudoinverse solutions for both over- and underdetermined linear systems. Both singular value decomposition (SVD) and truncated singular value decomposition (TSVD) will be discussed, followed by a general procedure for linearizing mildly nonlinear problems.

Nonlinear inversion is challenging and still remains an area of active research. Optimization problems resulting from nonlinear inversion can be difficult to solve and the solutions are generally not unique. In Sect. 2.3, we shall restrict ourselves to cases in which the CIP is extended well-posed in a known region and the objective function for optimization is convex in the region. Consequently, the inverse solution can be found by using a local optimization algorithm. Common numerical methods for local optimization and norm selection problems will be briefly discussed.

N-Z. Sun, A. Sun, *Model Calibration and Parameter Estimation,*
DOI 10.1007/978-1-4939-2323-6_2

The last section of this chapter, Sect. 2.4, focuses on Gauss–Newton method and its modified versions. These algorithms are especially effective when the least squares norm (i.e., L_2-norm) is used for inversion.

2.1 Approximate Solutions

2.1.1 The Direct Method for Inverse Solution

Recall from Sect. 1.3.3 that the inverse solution θ can be obtained either directly by solving the following model equation when u is given,

$$\mathcal{L}(u, \theta) = 0,$$

or indirectly by using the forward solution,

$$u = \mathcal{M}(\theta).$$

For a distributed parameter model, its state u can only be observed or measured at a limited number of times and locations, and the observed data are always "contaminated" by errors resulting from instrumentation, postprocessing, and scaling. Therefore, we must consider three spaces in the study of practical inverse problems: the *parameter space* $\mathbb{P}$, *state space* $\mathbb{U}$, and *observation space* $\mathbb{F}$. These three spaces contain $\theta, u = \mathcal{M}(\theta)$, and $u_D = \mathcal{D}(u)$, respectively, as their elements, where $\mathcal{D}(\cdot)$ is called an observation operator that specifies when and where u is sampled.

Figure 2.1 shows the relationships among the three spaces in the forward direction, in which $P_{ad} \subset \mathbb{P}$ is an admissible region of θ; $U_{ad} \subset \mathbb{U}$ is the value region of the forward solution $u = \mathcal{M}(\theta)$, where $\theta \in P_{ad}$; and $F_{ad} \subset \mathbb{F}$ is the value region of observation operator $u_D = \mathcal{D}(u)$, where $u \in U_{ad}$. The real observed counterpart of u_D is denoted by u_D^{obs}. Because u_D^{obs} contains observation error, it is generally not in F_{ad}.

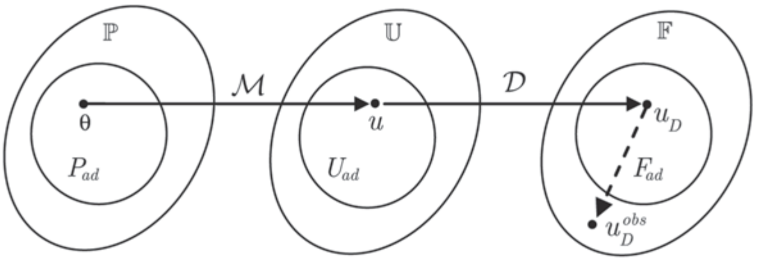


Fig. 2.1 Relationships among the three spaces of inverse problems: *parameter space* $\mathbb{P}$, *state space* $\mathbb{U}$, and *observation space* $\mathbb{F}$. The letter $\mathcal{M}$ denotes the forward model, $\mathcal{D}$ the observation operator, and the subscript ad denotes an admissible region

In mathematics, the theory of CIP has been developed for the general case that $\mathbb{P}$, $\mathbb{U}$, and $\mathbb{F}$ are infinite-dimensional function spaces. We assume that all these spaces are *Banach space* which, by definition, is a function space that contains finite-dimensional Euclidean spaces as its subspaces. A Banach space is furnished by various norms and measures, such as "distance" and "limit" (see Appendix A). We will use $\|e\|$ to denote the norm of an element e measured in the space that it belongs to.

Example 2.1 *The* $\mathbb{P}$, $\mathbb{U}$, and $\mathbb{F}$ *space of a mass transport model*
Assume that the distributed source term $E(\mathbf{x})$ in Eq. (1.1.12) is unknown and all other parameters in the equation are given. The concentration $C(\mathbf{x},t)$ is measured at ten locations every month for a year. In this case, the parameter space $\mathbb{P}$ and state space $\mathbb{U}$ are function spaces. The observation space $\mathbb{F}$ is an N-dimensional Euclidean space with $N = 120$. The distance between two trial source terms, $E_1(\mathbf{x})$ and $E_2(\mathbf{x})$, in the state space $\mathbb{U}$ can be measured by using the corresponding concentration distributions

$$\|C(E_1) - C(E_2)\|_2^2 = \int_0^T \int_{(\Omega)} [C(E_1,\mathbf{x},t) - C(E_2,\mathbf{x},t)]^2 d\mathbf{x}dt, \tag{2.1.1}$$

where $\|\cdot\|_2$ is the L_2-norm, $[0,T]$ is the total duration of observation, and Ω represents the spatial domain. Their difference in the observation space $\mathbb{F}$ can be measured by

$$\|C_D(E_1) - C_D(E_2)\|_2^2 = \sum_{j=1}^{12}\sum_{i=1}^{10} [C(E_1,\mathbf{x}_i,t_j) - C(E_2,\mathbf{x}_i,t_j)]^2. \tag{2.1.2}$$

Obviously, an infinite-dimensional function $E(\mathbf{x})$ cannot be identified by N data. To solve the inverse problem, the unknown distributed parameter needs to be discretized and/or parameterized first. For instance, we can divide the parameter space into eight homogeneous zones with constant source strengths $(E_1, E_2, \cdots, E_8)$. The parameter space $\mathbb{P}$ then becomes an eight-dimensional Euclidean space. ■

Let θ^t be the true parameter, $u(\theta^t)$ be the true system state, and u_D^{obs} be the observed values of $u_D(\theta^t)$ with observation error. The forward relationship between them can be represented by

$$\theta^t \underset{(\mathcal{M})}{\rightarrow} u(\theta^t) \underset{(\mathcal{D})}{\rightarrow} u_D(\theta^t) \approx u_D^{obs}. \tag{2.1.3}$$

In reality, θ^t, $u(\theta^t)$, and $u_D(\theta^t)$ are all unknown and we only have access to u_D^{obs}. The *direct method of inversion* attempts to find θ^t from u_D^{obs} by reversing the direction in (2.1.3). An inverse mapping (see Appendix A) $\mathcal{D}^{-1}$ of $\mathcal{D}$ and an inverse mapping $\mathcal{M}^{-1}$ of $\mathcal{M}$ are needed in order to complete this process

$$\theta^t \underset{(\mathcal{M}^{-1})}{\leftarrow} u(\theta^t) \underset{(\mathcal{D}^{-1})}{\leftarrow} u_D(\theta^t) \approx u_D^{obs}, \tag{2.1.4}$$

where $\mathcal{D}^{-1}$ can be considered an interpolation/extrapolation operator from u_D to u. Because of observation error and the error introduced during interpolation and extrapolation, using $\mathcal{D}^{-1}$ we can only obtain an approximation of $u(\theta^t)$, which is denoted as $\tilde{u} = \mathcal{D}^{-1}(u_D^{obs})$. Finding $\mathcal{M}^{-1}(\tilde{u})$ is equivalent to solving θ from the model equation

$$\mathcal{L}(\tilde{u}, \theta) = 0, \quad \theta \in P_{ad}. \tag{2.1.5}$$

This is an ill-posed problem: when $\tilde{u}$ contains error, (2.1.5) does not have an accurate solution, and no $\theta \in P_{ad}$ can make $\mathcal{L}(\tilde{u}, \theta)$ equal to zero. We may attempt to find an *approximate solution* to (2.1.5), but the problem of finding an approximate solution is still ill-posed. As we have shown in Example 1.11, even when $\tilde{u}$ is accurate, the inverse mapping $\mathcal{M}^{-1}$ may still be nonunique and unstable.

Although the direct method has these inherent difficulties, its simplicity and effectiveness are attractive and, thus, it continues to receive attention by the research community (e.g., Sun 1994; Zhan and Yortsos 2001; Irsa and Zhang 2012).

2.1.2 The Indirect Method of Inversion

Let us combine the forward mapping $\mathcal{M}$ and the sampling mapping $\mathcal{D}$ in (2.1.3) to form a composite mapping $\mathcal{DM}$ such that $\mathcal{DM}(\theta) = u_D(\theta)$. The inverse solution can then be obtained by solving the following equation

$$\mathcal{DM}(\theta) = u_D^{obs}, \quad \theta \in P_{ad}. \tag{2.1.6}$$

Like in the case of (2.1.5), an accurate solution of (2.1.6) does not exist because its right-hand side contains observation errors. Approximate solutions of (2.1.6) can be obtained indirectly through a trial-and-error process (see Sect. 1.3.3). First, a guessed parameter $\theta \in P_{ad}$ is used to calculate the model output $u_D(\theta)$, which is then compared to u_D^{obs}. The "distance" between them or the "mismatch" can be measured in the observation space by a norm $S(\theta) = \left\| u_D(\theta) - u_D^{obs} \right\|$. If the value of $S(\theta)$ is large, we try another θ. These steps are repeated until the value of $S(\theta)$ cannot be reduced further. The best solution of (2.1.6) is then chosen to be the minimizer of $S(\theta)$

$$\theta_{qs} = \arg\min_{\theta} \left\| u_D(\theta) - u_D^{obs} \right\|, \quad \theta \in P_{ad}. \tag{2.1.7}$$

We call θ_{qs} the *quasi-solution* of (2.1.6). It is obtained based on the criterion (C-1) of inverse problem formulation mentioned in Chap. 1 and recapitulated below:

(C-1) The model outputs should fit the observed data as close as possible.

The CIP is thus transformed into a single-objective optimization problem. The minimum, $S(\theta_{qs}) = \left\| u_D(\theta_{qs}) - u_D^{obs} \right\|$, is the residual of data fitting. Figure 2.2 shows a

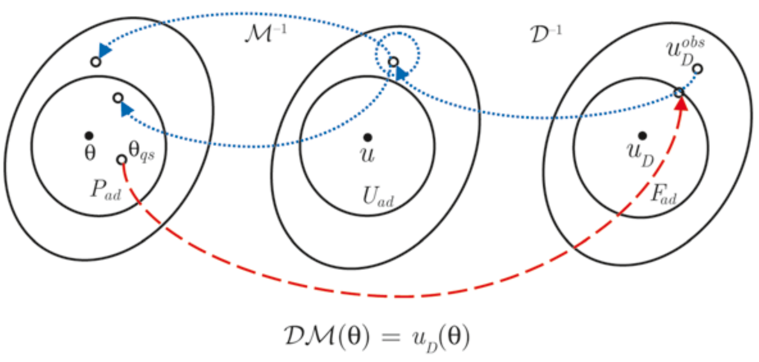


Fig. 2.2 Comparison between direct (*dotted line*) and indirect (*dashed line*) methods of inversion. The direct method requires the knowledge of inverse mapping $\mathcal{D}^{-1}$ and $\mathcal{M}^{-1}$, while the indirect method does not. All symbols are defined in the text

comparison of the direct (dotted lines) and indirect methods (dashed line) of inversion. The former uses the reverse direction and, thus, needs the inverse mappings $\mathcal{D}^{-1}$ and $\mathcal{M}^{-1}$, whereas the latter simply applies the forward mapping repeatedly and avoids the difficulty of finding $\mathcal{D}^{-1}$ and $\mathcal{M}^{-1}$.

Example 2.2 *Identification of mass transport parameters*
The canonical 1-D mass transport model given in Example 1.9 has an analytical solution, $C(x,t) = f(D_L, R; x, t)$, where the two mass transport parameters, D_L and R, can be estimated using the following set of nonlinear equations

$$f(D_L, R; x_i, t_j) = C_{i,j}^{obs}, \quad i = 1, 2, \ldots I, \quad j = 1, 2, \ldots J, \tag{2.1.8}$$

where $C_{i,j}^{obs}$ is the observed concentration at location x_i and time t_j, and I and J are the total numbers of sampling locations and times, respectively. When $C_{i,j}^{obs}$ contain observation errors, no (D_L, R) can satisfy all of the equations in (2.1.8). But, we can obtain a quasi-solution that is a minimizer of the following optimization problem,

$$\theta_{qs} = \arg\min_{\theta} \sum_{i=1}^{I} \sum_{j=1}^{J} \left(f(\theta; x_i, t_j) - C_{i,j}^{obs} \right)^2, \quad \theta \in P_{ad}, \tag{2.1.9}$$

where θ is a two-dimensional vector with components (D_L, R). For mass transport in porous media, the dispersion coefficient D_L is related to flow velocity v as $D_L = \alpha_L v$, where α_L is the longitudinal dispersivity to be estimated. If R can be estimated in the lab, then θ reduces to a scalar α_L. ■

So, how do we know that the quasi-solution θ_{qs} is really close to the true parameter θ^t? Let us introduce the concept of the *extended well-posedness*. The inverse problem (2.1.6) is said to be extended well-posed if (Tikhonov and Arsenin 1977)

- The problem has a solution θ^t in a subset $P_0 \subset P_{ad}$;
- The solution is unique in P_0: $u_D(\theta) = u_D(\theta^t)$ implies $\theta = \theta^t$ for any $\theta \in P_0$;
- The solution is stable in P_0: there exists a constant $c > 0$ such that

$$\left\|\theta - \theta^t\right\| \le c\left\|u_D(\theta) - u_D(\theta^t)\right\|, \quad \theta \in P_0. \tag{2.1.10}$$

Thus, when $u_D(\theta) \to u_D(\theta^t)$, we must have $\theta \to \theta^t$. In some situations, the inverse solution may not be unique in P_{ad} but may be unique in a subset $P_0 \subset P_{ad}$. The extended well-posedness assumes that P_0 can be determined on the basis of prior information. For example, a domain expert may suggest a reasonable range for finding the unknown parameter.

Now we show that the quasi-solution defined in (2.1.7) is indeed close to the true parameter when the observation error is sufficiently small. Let η be the norm of observation error. Because of the assumption of no model error, we have

$$\left\|u_D(\boldsymbol{\theta}^t) - u_D^{obs}\right\| \le \eta. \tag{2.1.11}$$

From (2.1.7), $u_D(\theta_{qs})$ should be closer to u_D^{obs} than $u_D(\theta^t)$ is. Therefore, we also have

$$\left\|u_D(\theta_{qs}) - u_D^{obs}\right\| \le \eta. \tag{2.1.12}$$

Combining the above two equations gives

$$\left\|u_D(\theta_{qs}) - u_D(\theta^t)\right\| \le \left\|u_D(\theta_{qs}) - u_D^{obs}\right\| + \left\|u_D(\theta^t) - u_D^{obs}\right\| = 2\eta. \tag{2.1.13}$$

Then, from the stability condition (2.1.10), we obtain $\left\|\theta_{qs} - \theta^t\right\| \le 2c\eta$, which implies that when $\eta \to 0$ we must have $\theta_{qs} \to \theta^t$. An essential requirement of this conclusion is that a unique and stable quasi-solution θ_{qs} exists. In other words, the optimization problem (2.1.7) is well-posed.

2.1.3 Well-Posedness of the Quasi-solution

In the previous section, we showed that the true parameter θ^t cannot be recovered in the presence of observation error, even if the inverse problem is well-posed. We may instead solve (2.1.7) to obtain a quasi-solution, θ_{qs}. If θ_{qs} is unique, stable, and close to θ^t, we say that the inverse problem is *well-solved* and the unknown parameter is *quasi-identifiable*. As shown in the previous section, quasi-identifiability can be achieved if the following requirements are satisfied

- A quasi-solution $\theta_{qs} \in P_0$ can be found by solving the optimization problem (2.1.7);
- The quasi-solution θ_{qs} is unique in P_0 and continuously dependent on observation data;

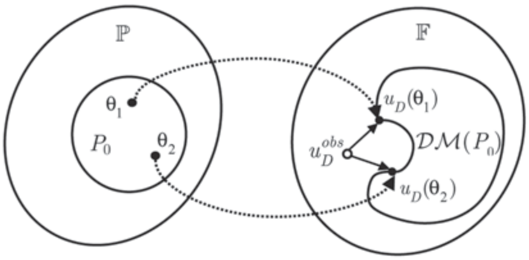


Fig. 2.3 Illustration of the possible nonuniqueness of quasi-solutions. Both $\theta_1 \in P_0$ and $\theta_2 \in P_0$, $u_D(\theta_1) \neq u_D(\theta_2)$, but their distances to u_D^{obs} are the same

- The inverse problem is extended well-posed in P_0; and
- The observation error is small.

The well-posedness of the optimization problem and the well-posedness of the inverse problem are combined in the above requirements. To satisfy these requirements, appropriate assumptions and conditions are needed not only on the inverse problem but also on data. As an example of the possible nonuniqueness of quasi-solutions, Fig. 2.3 shows that there exist $\theta_1 \in P_0$ and $\theta_2 \in P_0$ such that $u_D(\theta_1) \neq u_D(\theta_2)$, but their distances to u_D^{obs} are the same when the range set $\mathcal{DM}(P_0)$ is nonconvex. Chavent (2009) presented a theoretical study of this topic, including finding sufficient conditions for the combined well-posedness.

Solution of the optimization problem (2.1.7) depends on the complexity of model $\mathcal{M}(\theta)$, the quantity and quality of data u_D^{obs}, the norm defined in observation space $\mathbb{F}$, and the optimization algorithm to be used. The complexity of a model is characterized by the degrees of freedom (DOF) of θ. A more complex model has more parameters and, therefore, requires more information to support its identification. Data insufficiency is the main factor that leads to nonuniqueness and instability of inverse solutions. In reality, it is common that the difference between $u_D(\theta_1)$ and $u_D(\theta_2)$ is within the range of observation error, but θ_1 is significantly different from θ_2 (Fig. 2.3). More discrepancy between $u_D(\theta_1)$ and $u_D(\theta_2)$ can be revealed if we either increase the information content or reduce the observation error. Three approaches exist that can make the information content match the complexity of a model:

- Using additional information and assumptions to make the inverse problem well-posed;
- Replacing the original model by another one that has less complexity so that the existing data would be sufficient for inversion; or
- Increasing the quantity and/or improving the quality of observation data so that the model with given complexity can be identified.

The first approach will be considered starting from Chap. 3. If the first approach cannot give a satisfactory answer, the second approach may be used to reduce the difficulty of inversion but at the expense of introducing model structure errors to the identified model. If the identified model is deemed unreliable for the required model application, then the third approach, which includes selection of model complexity and design of new data collection strategy, must be used. All of these approaches and their related topics will be introduced successively in this book.

In the remainder of this chapter, we will focus on the methods for finding a quasi-solution in the deterministic framework and under the following assumptions: (i) no model error is present; (ii) the unknown parameter is an m-dimensional vector $\boldsymbol{\theta} \in \mathbb{R}^m$; and (iii) observation data provide sufficient information to make the parameter quasi-identifiable in a predetermined region $P_0 \subset \mathbb{R}^m$ and, thus, a unique and stable quasi-solution in the region can be found by using a local optimization method.

2.1.4 Parameterization and Discretization

Assume that a function, $\theta = f(\mathbf{x})$, defined in a spatial (or temporal) region, Ω, is the unknown parameter. When the DOF of $f(\mathbf{x})$ is infinite or high, we would not have sufficient data to make $f(\mathbf{x})$ identifiable. We have to find a low-DOF function, $\hat{f}(\mathbf{x})$, to approximate the original $f(\mathbf{x})$ for inversion. This process is known as *parameterization*, which is a topic that will be covered in full detail later in Chap. 6. In the following, we will briefly describe three basic parameterization methods to facilitate further discussion of topics in this chapter.

2.1.4.1 Parameterization by a Piecewise Constant Function

Parameterization by piecewise constant functions is the most commonly used parameterization method because of its simplicity and its role in stabilizing the inverse solution. In this method, the definition region of $f(\mathbf{x})$, Ω, is divided into m zones, $\{\Omega_i \mid i = 1, 2, \cdots, m\}$, and $\hat{f}(\mathbf{x})$ is a piecewise constant function over these zones,

$$\hat{f}(\mathbf{x}) = f_i, \quad \text{when } \mathbf{x} \in \Omega_i. \tag{2.1.14}$$

Identification of $f(\mathbf{x})$ is then replaced by identification of the m-dimensional vector

$$\mathbf{f}_m = (f_1, f_2, \cdots, f_m)^T, \quad \mathbf{f}_m \in R_{ad}^m, \tag{2.1.15}$$

where R_{ad}^m is the admissible region of $\mathbf{f}_m$ in the m-dimensional Euclidean space $\mathbb{R}^m$. The piecewise constant method is also referred to as the *zonation* method and can be represented in the following general form (see Appendix A):

$$f(\mathbf{x}) \approx \sum_{i=1}^{m} f_i \phi_i(\mathbf{x}), \tag{2.1.16}$$

where the basis function $\phi_i(\mathbf{x}) = 1$, when $\mathbf{x} \in \Omega_i$; and $\phi_i(\mathbf{x}) = 0$, otherwise. Zonation can be applied to any continuous or discontinuous functions.

2.1.4.2 Parameterization by the Finite Element Method

The finite element method (FEM) parameterizes the unknown function $f(\mathbf{x})$ in the same form as that in (2.1.16), in which m is the number of nodes, $f_i = f(\mathbf{x}_i)$ is the value of the function at node $\mathbf{x}_i$, and $\phi_i(\mathbf{x})$ is a local interpolation basis function satisfying: (i) $\phi_i(\mathbf{x}_j) = 1$ when $j = i$ and $\phi_i(\mathbf{x}_j) = 0$ otherwise; (ii) $\phi_i(\mathbf{x}) \neq 0$ only in elements containing the node $\mathbf{x}_i$, and (iii) in each element, $\phi_i(\mathbf{x})$ is a low-degree polynomial function, such as a linear function. In forward solution, $f(\mathbf{x})$ is the unknown state variable; once all basis functions are determined, all nodal values of $\mathbf{f}_m$ can be obtained by solving the finite element discrete equations. For inverse solution, $f(\mathbf{x})$ is the unknown distributed parameter and the elements of $\mathbf{f}_m$ are the unknowns to be identified.

Note that (i) the FEM local interpolation is especially suitable for representing multiscale parameters and (ii) the element size used for inversion (i.e., discretizing distributed parameters) is typically much larger than that used for forward solution.

2.1.4.3 Parameterization by Kriging

Kriging is a statistical interpolation method that originated in geostatistics for estimating the value of a spatially varying function $f(\mathbf{x})$ at an unmeasured location $\mathbf{x}$, using its measured values at a set of locations $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$. Let f_i be the measured value of $f(\mathbf{x})$ at location $\mathbf{x}_i$ $(i = 1, 2, \cdots, m)$. The minimum variance estimate of $f(\mathbf{x})$ is given by

$$f(\mathbf{x}) \approx \sum_{i=1}^{m} f_i \lambda_i(\mathbf{x}), \tag{2.1.17}$$

where the kriging weights, $\{\lambda_i(\mathbf{x})\}_{i=1}^{m}$, are obtained by solving a set of linear equations. A detailed discussion of kriging methods will be given in Chap. 6. For now, it suffices to know that when kriging is used as a parameterization method for inversion, the function values at specified locations become the unknowns to be identified.

2.1.4.4 Discrete Parameterization

Before a forward model can be solved numerically, all state variables and distributed parameters must be represented by their nodal values through discretization. From the perspective of parameterization, discretization naturally creates a full parameterization, after which a distributed parameter $\theta = f(\mathbf{x})$ is replaced by an N-dimensional vector $\mathbf{f}_N$, where N is the number of nodes. However, the vector $\mathbf{f}_N$ may not be identifiable if its DOF is large. We need a reparameterization process to further reduce $\mathbf{f}_N$ to an m-dimensional vector $\mathbf{f}_m (m \ll N)$ for inverse solution. Vectors $\mathbf{f}_N$ and $\mathbf{f}_m$ are related by a linear transformation

$$\mathbf{f}_N = \mathbf{G}\mathbf{f}_m. \tag{2.1.18}$$

Equation (2.1.18) can be seen as a discrete form of parameterization, where the $N \times m$ transformation matrix **G** depends on the method used for parameterization. For example, in the case of zonation, the elements g_{ij} of **G** satisfy

$$g_{ij} = \begin{cases} 1, & \text{node } i \in \text{ zone } j \\ 0, & \text{node } i \notin \text{ zone } j \end{cases}$$

for $i = 1, 2, \cdots, N$ and $j = 1, 2, \cdots, m$.

2.1.4.5 Application to Inverse Solution

Parameterization turns an infinite-dimensional distributed parameter into a low-dimensional vector, which may be identifiable with limited data. In the general model equation $\mathcal{L}(u, \theta) = 0$, θ denotes all physical or nonphysical model parameters which may be scalars, vectors, and functions. After parameterization, θ is expressed as a vector

$$\boldsymbol{\theta}_m = (\theta_1, \theta_2, \cdots, \theta_m)^T. \tag{2.1.19}$$

The inverse problem then becomes identification of the vector $\boldsymbol{\theta}_m$. Elements of $\boldsymbol{\theta}_m$ may include parameters that have different dimensions and magnitudes. Thus, using raw values of $\boldsymbol{\theta}_m$ directly during inversion may cause loss of accuracy of the inverse solutions. If the range of each element of $\boldsymbol{\theta}_m$ can be estimated from prior information, we can transform $\boldsymbol{\theta}_m$ into a dimensionless form. Assume the range of variation of θ_i is $[a_i, b_i]$, then $\theta_i^\circ = (\theta_i - a_i) / (b_i - a_i)$ is a dimensionless variable that varies in the interval $[0, 1]$. We can then identify elements of the normalized vector during inversion. Similar scaling techniques are also used extensively by machine learning algorithms, as we will see in Chap. 8.

2.2 Linear Model Identification

2.2.1 Linear Model and Normal Equation

Linear model inversion is the simplest inverse problem, which requires solving the following system of algebraic equations (see Appendix C)

$$\mathbf{G}\boldsymbol{\theta}_m = \mathbf{d}, \tag{2.2.1}$$

where $\mathbf{G}$ is an $n \times m$ matrix, $\boldsymbol{\theta}_m \in \mathbb{R}^m$ is the unknown parameter vector, and $\mathbf{d} \in \mathbb{R}^n$ is the data vector. For clarity, we will drop the subscript m on $\boldsymbol{\theta}_m$ where no confusion should occur. Linear model inversion serves as a basis for inverting more complex linear and nonlinear models governed by ODEs and PDEs. In Chap. 1, we showed that using a numerical method (e.g., FEM or FDM), a PDE is first reduced to a system of ODEs and then further reduced to a system of algebraic equations

$$\mathcal{L}_i(\mathbf{u}, \boldsymbol{\theta}) = 0, \quad i = 1, 2, \cdots, n, \tag{2.2.2}$$

where $n = n_S \times n_T$, n_S is the number of nodes and n_T is the number of time steps, $\mathbf{u} \in \mathbb{R}^n$ is the discretized state variable, and $\boldsymbol{\theta} \in \mathbb{R}^m$ is the unknown parameter vector. If all equations in (2.2.2) are linear with respect to $\boldsymbol{\theta}$, we have a system of linear equations like (2.2.1), in which both $\mathbf{G}$ and $\mathbf{d}$ are dependent on $\mathbf{u}$. The system of equations (2.2.2) can be used to obtain the inverse solution $\boldsymbol{\theta}$ if $\mathbf{u}$ is approximated using its measurements (e.g., via interpolation and extrapolation), and $\mathbf{G}$ and $\mathbf{d}$ are calculated. This is the *direct method of inversion*.

When the forward solution is obtained from (2.2.2), we have

$$\mathcal{M}_i(\boldsymbol{\theta}) = u_i^{obs}, \quad i = 1, 2, \cdots, n, \tag{2.2.3}$$

where u_i^{obs} $(i = 1, 2, \cdots, n)$ are observed values of the state variable. If all of these equations are linear with respect to $\boldsymbol{\theta}$, we again obtain a system of linear equations like (2.2.1), where $\mathbf{G}$ is an $n \times m$ matrix and $\mathbf{d}$ consists of observed values u_i^{obs}. This is the *indirect method of inversion*.

Therefore, both direct and indirect methods involve the solution of (2.2.1) when the model is linear. Unfortunately, the solution of (2.2.1) is inherently an ill-posed problem. Three cases are possible:

- *Underdetermined case:* This is the case when the dimension of parameter is larger than the number of data ($m > n$); the solution of (2.2.1) is nonunique in this case.
- *Uniquely determined case:* This is the case when the dimension of parameter is equal to the number of data ($m = n$); the solution of (2.2.1) may be unstable when $\mathbf{G}$ is close to singular in this case.
- *Overdetermined case:* This is the case when the dimension of parameter is less than the number of data ($m < n$); if (2.2.1) is an inconsistent system, the solution of (2.2.1) does not exist because no $\boldsymbol{\theta}$ can satisfy all equations exactly in this case.

Example 2.3 *Identification of contaminant release history*
Let us revisit the 1-D mass transport problem in Example 1.5, for which the governing PDE is

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial x}\left(D_L \frac{\partial C}{\partial x}\right) - v\frac{\partial C}{\partial x} - RC + s(t), \tag{2.2.4}$$

subject to

$$C(x,t)\,|_{t=0} = C_0(x),$$
$$C(x,t)\,|_{x=0} = C_{in}(t),$$
$$\left.\frac{\partial C}{\partial x}\right|_{x=L} = 0.$$

Our goal is to recover the unknown release history of a point source that exists at a known location. This is a special case of the more general contaminant source identification problem. Here, we would like to identify the shape of source function $\theta(t) = s(t)$ in the time interval $[0, T]$ based on n concentration samples obtained at different locations x_i and times t_i

$$\{C^{obs}(x_i, t_i) \mid i = 1, 2, \cdots, n\}.$$

First, the unknown release history $\theta(t)$ needs to be parameterized into an m-dimensional vector $\boldsymbol{\theta}$. We apply the zonation method to discretize the time domain $[0, T]$ into m nonoverlapping subintervals such that the source strength is constant in each subinterval

$$\theta(t) = \theta_j \text{ in } [t_{j-1}, t_j], \quad j = 1, 2, \cdots, m.$$

The source strengths are components of the unknown vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \cdots, \theta_m)^T$. A linear system for inversion can be assembled from the following equation

$$C(x_i, t_i, \boldsymbol{\theta}) = C^{obs}(x_i, t_i), \quad i = 1, 2, \cdots, n, \tag{2.2.5}$$

where $C(x_i, t_i, \boldsymbol{\theta})$ are model outputs. The source term in (2.2.4) is now replaced by its parameterization

$$s(t) = \sum_{j=1}^{m} \theta_j s_j^0(t),$$

where the basis function $s_j^0(t) = 1$, when $t \in [t_{j-1}, t_j]$, and 0 otherwise. For reasons that will soon become clear, the basis function $s_j^0(t)$ is also called a unit-pulse source. Because (2.2.4) is a linear PDE, the forward solution of the model can be obtained by superposition. Let $C_j^0(x,t)$ be the solution of the model when $s_j^0(t)$ is used as its source term, then $C(x,t) = \sum_{j=1}^{m} \theta_j C_j^0(x,t)$. After running the forward model m times, the system of equations (2.2.5) is converted to the form of a linear system (2.2.1), $\mathbf{G}\boldsymbol{\theta} = \mathbf{d}$, where the elements of $n \times m$ matrix $\mathbf{G}$ and n-dimensional vector $\mathbf{d}$ are $G_{ij} = C_j^0(x_i, t_i)$ and $d_i = C^{obs}(x_i, t_i)$, respectively. The method of assembling the system of equations is known as the impulse-response method (Sun et al. 2006). It is similar to derivation of hydrographs from unit hydrographs in hydrology.

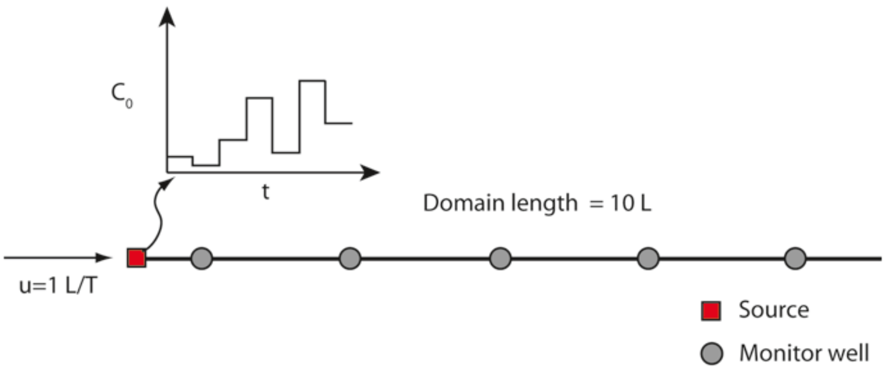


Fig. 2.4 Setup of the source identification problem

As an illustration, let us assume that the length of the 1-D domain is 10 [L], $v = 1$ [L/T], and $D_L = 0.1$ [L^2/T]. The source is located at the origin and we would like to estimate its release history during time interval $(0, 7)$ [T]. The time domain is discretized into seven equal subintervals and the source strength is assumed constant within each subinterval. Thus, the number of unknowns is equal to $m = 7$. For testing purpose, we assume that the "true" average source strengths ($\boldsymbol{\theta}^t$) in all subintervals are known, which are $\{10, 5, 30, 80, 20, 100, 50\}$ [M/L^3], respectively (Fig. 2.4). These values are used to generate the hypothetical concentration observations to be used by inversion.

Say the concentration is measured at five locations $\{1,3,5,7,9\}$ [L] and six sampling times $\{0.5,0.8,1.2,1.8,2.0,10.0\}$ [T]. The total number of data is $n = 30$, which leads to an overdetermined case because $m < n$. If concentration is measured at a location $x = 1$ [L] for seven sampling times $\{0.5,0.8,1.2,1.8,2.0,5.0,10.0\}$ [T], the problem becomes a uniquely-determined case ($m = n$). If concentration is measured at $x = 1$ [L] for only five times at $\{1.0, 2.0, 3.0, 5.0, 10.0\}$ [T], we have an underdetermined problem at hand ($m > n$). The solutions to these different cases will be discussed in subsequent examples. ■

The *least squares solution* to (2.2.1) is an approximate solution to (2.1.7), which minimizes the distance between the right-hand side and the left-hand side measured by the L_2-norm (i.e., $\|\mathbf{G}\boldsymbol{\theta} - \mathbf{d}\|_2$). It can be obtained by solving the *normal equation* defined as follows

$$\mathbf{G}^T\mathbf{G}\boldsymbol{\theta} = \mathbf{G}^T\mathbf{d}. \tag{2.2.6}$$

When $\mathbf{G}^T\mathbf{G}$ is invertible, the least squares solution is simply

$$\boldsymbol{\theta}_{LS} = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{d}, \tag{2.2.7}$$

which is the solution for the overdetermined case.

Example 2.4 *Release history identification for the overdetermined case*
Consider the linear system that was obtained for the overdetermined case in Example 2.3

$$[\mathbf{G}]_{30\times 7}[\boldsymbol{\theta}]_{7\times 1} = [\mathbf{d}]_{30\times 1}, \tag{2.2.8}$$

where the subscripts correspond to matrix/vector dimensions. If the right-hand side **d** does not contain measurement error, the least squares solution of (2.2.8) fully recovers the "true" source strengths

$$\boldsymbol{\theta}_{LS} = \{10.0, 5.0, 30.0, 80.0, 20.0, 100.0, 50.0\}.$$

If we perturb each observation data by 5 % random error, the accuracy of inverse solution degrades significantly

$$\boldsymbol{\theta}_{LS} = \{9.6, 5.6, 31.8, 71.4, 41.3, 82.1, 54.0\},$$

which has a root-mean-square error (RMSE) of 11.1. The RMSE is often used to measure the goodness-of-fit of an estimator

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{\theta}_i - \theta_i)^2},$$

where $\hat{\boldsymbol{\theta}} = \{\hat{\theta}_i\}_{i=1}^{m}$ is the estimator and $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^{m}$ is the estimated parameter vector. For this example, $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_{LS}$ and $\boldsymbol{\theta} = \boldsymbol{\theta}^t$. If the magnitude of random observation error is increased to 10 %, the solution becomes even worse

$$\boldsymbol{\theta}_{LS} = \{10.1, 4.1, 29.2, 117.7, -43.2, 133.0, 46.7\},$$

which includes a nonphysical negative value and the RMSE is 30.5. These results show that the quasi-solution of a well-posed inverse problem can be sensitive to measurement errors and it is only close to the true solution when the observation error is sufficiently small. ■

The Matlab backslash operator (\) is used to obtain the least squares solutions for the above example. Other high-level programming languages provide similar tools, such as `lstsq` in Numerical Python (NumPy) and `lm` in R.

2.2.2 *Estimation Using Singular Value Decomposition*

The least squares solution $\boldsymbol{\theta}_{LS}$ defined by (2.2.7) exists only when $\mathbf{G}^T\mathbf{G}$ is full rank. If $\mathbf{G}^T\mathbf{G}$ is singular, $\boldsymbol{\theta}_{LS}$ does not exist. In this section, we introduce a powerful technique for obtaining the *pseudoinverse solution* of the system of linear equations (2.2.1). The technique is based on *singular value decomposition* (SVD).

The SVD provides a factorization of real or complex matrix (Golub and Van Loan 1996; Strang 2006). Using SVD, any $n \times m$ matrix $\mathbf{G}$ can be factored into

$$\mathbf{G} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T, \tag{2.2.9}$$

in which $\mathbf{U}$ is an $n \times n$ matrix whose columns are orthogonal and form a set of basis vectors spanning the data space $\mathbb{R}^n$ (see Appendix C); $\mathbf{V}$ is an $m \times m$ matrix whose columns are also orthogonal and form a set of basis vectors spanning the parameter space $\mathbb{R}^m$; $\boldsymbol{\Sigma}$ is an $n \times m$ diagonal matrix, and the entries of which are called the *singular values* of $\mathbf{G}$. The columns of $\mathbf{U}$ and $\mathbf{V}$ are called *singular vectors* of $\mathbf{G}$. By convention, the singular values on the diagonal of $\boldsymbol{\Sigma}$ are arranged in a descending order. If only the first r singular values are nonzero,

$$s_1 \ge s_2 \ge \cdots \ge s_r > s_{r+1} = \cdots = s_{\min(n,m)} = 0, \tag{2.2.10}$$

we can construct a compact form of SVD

$$\mathbf{G} = \mathbf{U}_r\boldsymbol{\Sigma}_r\mathbf{V}_r^T, \tag{2.2.11}$$

where $\mathbf{U}_r$ contains the r leftmost columns of $\mathbf{U}$, $\mathbf{V}_r$ contains the r leftmost columns of $\mathbf{V}$, and $\boldsymbol{\Sigma}_r$ is the upper-left block of $\boldsymbol{\Sigma}$ with r nonzero singular values on its diagonal

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{2.2.12}$$

Using SVD (2.2.11), we can calculate the inverse of $\mathbf{G}$ as

$$\mathbf{G}^\dagger = \mathbf{V}_r\boldsymbol{\Sigma}_r^{-1}\mathbf{U}_r^T, \tag{2.2.13}$$

in which $\mathbf{G}^\dagger$ is known as the Moore–Penrose pseudoinverse of $\mathbf{G}$. The pseudoinverse is a generalized version of the conventional inverse matrix $\mathbf{G}^{-1}$. It can be shown that when $\mathbf{G}^{-1}$ exists, $\mathbf{G}^\dagger = \mathbf{G}^{-1}$; and even when $\mathbf{G}^{-1}$ does not exist, $\mathbf{G}^\dagger$ still exists and is unique (Golub and Van Loan 1996).

With the introduction of $\mathbf{G}^\dagger$, the SVD pseudoinverse solution for problem (2.2.1) can be defined as

$$\boldsymbol{\theta}^\dagger = \mathbf{G}^\dagger\mathbf{d} = \mathbf{V}_r\boldsymbol{\Sigma}_r^{-1}\mathbf{U}_r^T\mathbf{d}. \tag{2.2.14}$$

Let $\mathbf{u}_{r,i}$ and $\mathbf{v}_{r,i}$ be the *i*-th column of $\mathbf{U}_r$ and $\mathbf{V}_r$, respectively, the matrix product of the right-hand side of the above equation gives

$$\boldsymbol{\theta}^\dagger = \sum_{i=1}^{r} \frac{\mathbf{u}_{r,i}^T\mathbf{d}}{s_i}\mathbf{v}_{r,i}, \tag{2.2.15}$$

where s_i are singular values. The Moore–Penrose pseudoinverse solution can be obtained using the `pinv` routine that is available in Matlab, NumPy, and R. All three packages also provide an SVD routine named `svd`.

The SVD is a useful tool for linear model inversion. The pseudoinverse solution $\boldsymbol{\theta}^\dagger$ actually is a quasi-solution $\boldsymbol{\theta}_{qs}$ for problem (2.1.7) when the L_2-norm is used. For the overdetermined case ($m < n$) and when $(\mathbf{G}^T\mathbf{G})$ is invertible, we have

$$\begin{aligned}(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T &= (\mathbf{V}_r\boldsymbol{\Sigma}_r\mathbf{U}_r^T\mathbf{U}_r\boldsymbol{\Sigma}_r\mathbf{V}_r^T)^{-1}\mathbf{V}_r\boldsymbol{\Sigma}_r\mathbf{U}_r^T \\ &= \mathbf{V}_r\boldsymbol{\Sigma}_r^{-1}\boldsymbol{\Sigma}_r^{-1}\mathbf{V}_r^T\mathbf{V}_r\boldsymbol{\Sigma}_r\mathbf{U}_r^T \\ &= \mathbf{V}_r\boldsymbol{\Sigma}_r^{-1}\mathbf{U}_r^T \\ &= \mathbf{G}^\dagger.\end{aligned} \tag{2.2.16}$$

Therefore, the pseudoinverse solution $\boldsymbol{\theta}^\dagger$ in this case is equal to $\boldsymbol{\theta}_{LS}$ obtained by the normal equation (2.2.7), but the calculation of $\boldsymbol{\theta}^\dagger$ by (2.2.15) is more effective. When $(\mathbf{G}^T\mathbf{G})$ is not invertible, (2.2.7) cannot be used to calculate $\boldsymbol{\theta}_{LS}$, but $\boldsymbol{\theta}^\dagger$ can still be calculated.

For the underdetermined case ($m > n$), the solution of $\mathbf{G}\boldsymbol{\theta} = \mathbf{d}$ is nonunique. With SVD, the system can be rewritten as

$$\mathbf{G}\boldsymbol{\theta} = \mathbf{d} \to \mathbf{U}_r^T(\mathbf{U}_r\boldsymbol{\Sigma}_r\mathbf{V}_r^T)\boldsymbol{\theta} = \mathbf{U}_r^T\mathbf{d} \to \boldsymbol{\Sigma}_r\mathbf{V}_r^T\boldsymbol{\theta} = \mathbf{U}_r^T\mathbf{d}. \tag{2.2.17}$$

Thus, its least squares solutions must satisfy

$$\begin{aligned}\min_{\boldsymbol{\theta}}\left\|\mathbf{G}\boldsymbol{\theta} - \mathbf{d}\right\|_2^2 &= \min_{\boldsymbol{\theta}}\left\|\boldsymbol{\Sigma}_r\mathbf{V}_r^T\boldsymbol{\theta} - \mathbf{U}_r^T\mathbf{d}\right\|_2^2 \\ &= \sum_{i=1}^{r}\left(s_i\mathbf{v}_{r,i}^T\boldsymbol{\theta} - \mathbf{u}_{r,i}^T\mathbf{d}\right)^2 + \sum_{i=r+1}^{n}\left(\mathbf{u}_{r,i}^T\mathbf{d}\right)^2.\end{aligned} \tag{2.2.18}$$

According to (2.2.18), the pseudoinverse solution $\boldsymbol{\theta}^\dagger$ is such a solution that its L_2-norm in $\mathbb{R}^m$ is the smallest among all possible solutions because all $\theta_i^\dagger = 0$ for $i \geq r+1$, viz.

$$\boldsymbol{\theta}^\dagger = \arg\min\left\|\boldsymbol{\theta}\right\|_2^2, \text{ s. t. } \mathbf{G}\boldsymbol{\theta} = \mathbf{d}. \tag{2.2.19}$$

We have mentioned that for the uniquely determined case ($m = n$), the solution of $\mathbf{G}\boldsymbol{\theta} = \mathbf{d}$ may become unstable when matrix $\mathbf{G}$ is nearly singular. Although the pseudoinverse solution $\boldsymbol{\theta}^\dagger$ always exists and is unique even when the matrix $\mathbf{G}$ is rank deficient ($r < \min(m, n)$), its stability, however, must be considered, especially when used for model inversion.

2.2.3 Truncated Singular Value Decomposition

Let us consider the stability problem of a pseudoinverse solution, namely, whether the solution is continuously dependent on data. An error in data, $\Delta\mathbf{d}$, will cause an error, $\Delta\boldsymbol{\theta}^{\dagger}$, in the resulting inverse solution. The stability of the solution can be assessed using the relative errors of solution and data

$$\frac{\left\|\Delta\boldsymbol{\theta}^{\dagger}\right\|_2}{\left\|\boldsymbol{\theta}^{\dagger}\right\|_2} \le \kappa(\mathbf{G})\frac{\left\|\Delta\mathbf{d}\right\|_2}{\left\|\mathbf{d}\right\|_2}, \tag{2.2.20}$$

where $\kappa(\mathrm{G})$ is called the matrix *condition number* of $\mathbf{G}$. Larger condition numbers mean that the solution is less stable. A linear system is said to be *ill-conditioned* when its condition number is much greater than 1. Using SVD, the condition number can be calculated as

$$\kappa(\mathbf{G}) = s_1 / s_r, \tag{2.2.21}$$

where s_1 and s_r are the maximal and the minimal nonzero singular value of $\mathbf{G}$, respectively. As can be seen from (2.2.15), when s_r is close to zero, a small error in $\mathbf{d}$ will cause a large error in $\boldsymbol{\theta}^{\dagger}$.

Linear systems derived from ill-posed inverse problems, such as (2.2.2) and (2.2.3), are also ill-conditioned in many situations. Therefore, we frequently encounter ill-conditioned linear systems in the study of inverse problems. Consider the case that the forward solution is linear with respect to $\boldsymbol{\theta}$ and (2.2.1) is derived from (2.2.3), where the elements g_{ij} of matrix $\mathbf{G}$ are $\partial u_{D,i} / \partial\theta_j$, $i = 1, 2, \cdots, n$, and $j = 1, 2, \cdots, m$. If all observations are insensitive to a certain component θ_j of $\boldsymbol{\theta}$, then all elements in the *j*th column of $\mathbf{G}$ are close to zero which, in turn, means that the corresponding singular value of $\mathbf{G}$ is also close to zero. Consequently, the system is ill-conditioned and the identified θ_j has large uncertainty. In this case, if we cannot collect additional data that are sensitive to θ_j, we need to drop θ_j from $\boldsymbol{\theta}$, which can be done by using the *truncated singular value decomposition* (TSVD).

In TSVD, only the first k singular values $s_1 \ge s_2 \ge \cdots \ge s_k$ are retained. All other singular values that are less than s_k are truncated. As a result, the following approximation of the original matrix and corresponding pseudoinverse solution are obtained

$$\mathbf{G}_{ts} = \mathbf{U}_k\boldsymbol{\Sigma}_k\mathbf{V}_k^T, \quad \boldsymbol{\theta}_{ts}^{\dagger} = \mathbf{V}_k\boldsymbol{\Sigma}_k^{-1}\mathbf{U}_k^T\mathbf{d} = \mathbf{G}_{ts}^{\dagger}\mathbf{d}. \tag{2.2.22}$$

Comparing to $\boldsymbol{\theta}^{\dagger}$ obtained by SVD, the solution $\boldsymbol{\theta}_{ts}^{\dagger}$ obtained by TSVD has a smaller number of DOF ($k < r$) but is more stable and robust because the matrix condition number of $\mathbf{G}_{ts}$ is reduced to s_1 / s_k. The reduction of solution instability comes at the expense of an increase in fitting residuals. Therefore, determination of the truncation threshold s_k needs to make a trade-off between minimizing the fitting residual and maximizing the stability of the solution. As we will soon see in Chap. 3, this is a type of two-criterion optimization or regularization problem. We remark that $\mathbf{G}\boldsymbol{\theta}_{ts}^{\dagger}$ cannot fit observation data $\mathbf{d}$ accurately even without observation error because of the model structure error caused by truncation.

Example 2.5 *SVD solution of an ill-conditional least squares problems*
In Example 2.4, we showed that adding small noise to data may cause a significant change in the inverse solution. In this example, SVD is used to yield further insight into the problem. The solution instability, as shown below, is caused by the large condition number of the coefficient matrix.

Figure 2.5 shows that the maximum and the minimum singular values of matrix $\mathbf{G}$ in (2.2.8) are 1.24 and 1.5×10^{-4}, respectively. The matrix condition number is $\kappa(\mathbf{G}) \approx 8300$, which is calculated using the formula in (2.2.21). If the smallest singular value is removed, κ decreases significantly to 12.7.

Using the TSVD pseudoinverse formula given in (2.2.22) and keeping only the first six singular values (i.e., setting k to 6), we get a TSVD solution

$$\boldsymbol{\theta}_{ts}^{\dagger} = \{10.0, 4.1, 43.5, 56.1, 61.2, 62.9, 60.8\}.$$

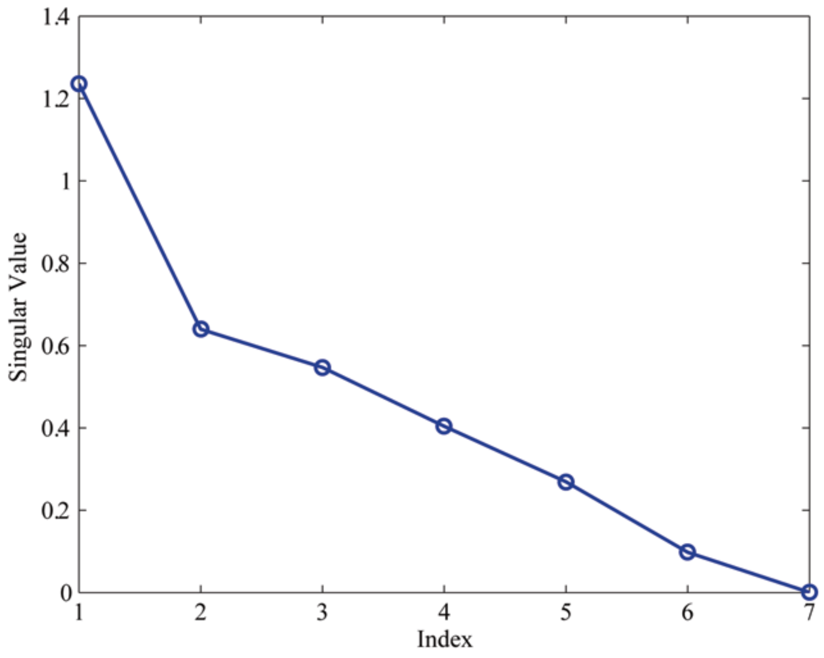


Fig. 2.5. Singular values of the response matrix $\mathbf{G}$ in Example 2.5

for the case in which the data have 10% observation error, and the RMSE is 23.3. The stability of the solution is clearly improved over the same case in Example 2.4. However, when no observation error is present, the following TSVD solution is obtained

$$\boldsymbol{\theta}_{ts}^{\dagger} = \{10, 5, 36.2, 53.1, 65.5, 69.5, 56.1\},$$

which is different from the accurate solution. Generally speaking, any improvement in numerical stability may come at the expense of the loss of accuracy. In this example, the loss of accuracy is caused by the truncation of singular values.

2.2.4 Linearization

Let the parameter space $\mathbb{P}$ and observation space $\mathbb{F}$ be finite dimensional Euclidean spaces such that $\boldsymbol{\theta} \in \mathbb{R}^m$ and $\mathbf{u}_D^{obs} \in \mathbb{R}^n$, respectively. When the model output $\mathbf{u}_D(\boldsymbol{\theta})$ is mildly nonlinear around the parameter $\boldsymbol{\theta}_0$, it can be approximated by a linear function (first-order Taylor's expansion)

$$\mathbf{u}_D(\boldsymbol{\theta}) \approx \mathbf{u}_D(\boldsymbol{\theta}_0) + \mathbf{J}_D(\boldsymbol{\theta} - \boldsymbol{\theta}_0), \tag{2.2.23}$$

where the Jacobian $\mathbf{J}_D$ is defined as

$$\mathbf{J}_D = \left[\frac{\partial u_{D,i}}{\partial \theta_j}\right]_{n \times m}, \; i = 1, \cdots, n, \; j = 1, \cdots, m \tag{2.2.24}$$

in which all partial derivatives of $\mathbf{J}_D$ are evaluated at point $\boldsymbol{\theta}_0$. If $\boldsymbol{\theta}_0$ is close to the true parameter, substitution of (2.2.23) into the inverse solution equation $\mathbf{u}_D(\boldsymbol{\theta}) = \mathbf{u}_D^{obs}$ gives

$$\mathbf{J}_D\boldsymbol{\theta} = \mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta}_0) + \mathbf{J}_D\boldsymbol{\theta}_0. \tag{2.2.25}$$

Equation (2.2.25) can be seen as a linear model in the form of (2.2.1) if we replace $\mathbf{G}$ on the left-hand side by $\mathbf{J}_D$, $\mathbf{d}$ on the right-hand side by $\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta}_0) + \mathbf{J}_D\boldsymbol{\theta}_0$, and multiply both sides by $\mathbf{J}_D^T$. The corresponding normal equation becomes

$$\mathbf{J}_D^T\mathbf{J}_D\boldsymbol{\theta} = \mathbf{J}_D^T\left[\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta}_0) + \mathbf{J}_D\boldsymbol{\theta}_0\right]. \tag{2.2.26}$$

When $\mathbf{J}_D^T\mathbf{J}_D$ is invertible, a least squares solution of the linearized inverse problem can be obtained by solving (2.2.26). Otherwise, we can find a pseudoinverse solution or a TSVD solution. An algorithm for successive linearization can be summarized as:

1. Guess an initial solution $\boldsymbol{\theta}_0$ in the admissible set P_{ad}
2. Use linearization at $\boldsymbol{\theta}_0$ to form (2.2.23)

3. Solve the linear inverse problem (2.2.26) to obtain a solution $\boldsymbol{\theta}_1$
4. Test if a predefined convergence criterion $\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_0\| < \varepsilon$ is reached
5. If yes, stop; otherwise, replace $\boldsymbol{\theta}_0$ by $\boldsymbol{\theta}_1$ and go back to step 2.

Example 2.6 *Linearization example*
The continuity equation for 1-D flow in a rectangular channel is given by (see Example 1.4)

$$\frac{\partial h}{\partial t} + \frac{\partial (hv)}{\partial x} = q, \tag{2.2.27}$$

where $h(x,t)$ is water depth (the state variable), $v(x,t)$ is longitudinal flow velocity, and $q(x,t)$ is lateral inflow rate per unit length of the channel. The kinematic wave equation assumes that the energy grade line is parallel to the channel bottom, and that the flow is steady and uniform. As a result, the momentum conservation equation becomes $S_f = S_0$, and both can be related to flow velocity v using, for example, the Manning's equation

$$S_f = S_0 = \frac{n^2 v^2}{h^{4/3}}, \tag{2.2.28}$$

in which n is the Manning's roughness coefficient. The longitudinal volumetric flow rate, Q, which is a product of channel width (B), water depth (h), and flow velocity (v), can be written using the Manning's equation as

$$Q = vhB = \frac{1.49}{n} S_0^{\,1/2} A h^{2/3}, \tag{2.2.29}$$

where $A = hB$ is the channel's cross-sectional area, the constant 1.49 is used when all quantities are given in English units, and a wide channel is implicitly assumed. Multiplying both sides of (2.2.27) by B gives

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = q_B, \tag{2.2.30}$$

where $q_B = qB$. Solving for h from (2.2.29) and substituting the result into (2.2.30), we get

$$\alpha\beta Q^{\beta-1}\frac{\partial Q}{\partial t} + \frac{\partial Q}{\partial x} = q_B, \tag{2.2.31}$$

where $\alpha = (nB^{2/3} / (1.49 S_0^{1/2}))^{\beta}$ and $\beta = 3/5$.

Given the inflow hydrograph at the channel inlet, the initial condition, and channel parameters, the nonlinear kinematic wave equation (2.2.31) can be discretized by using FDM

$$\alpha\beta\left(\frac{Q_{i+1}^{t}+Q_{i}^{t+1}}{2}\right)^{\beta-1}\left(\frac{Q_{i+1}^{t+1}-Q_{i+1}^{t}}{\Delta t}\right)+\frac{Q_{i+1}^{t+1}-Q_{i}^{t+1}}{\Delta x}=\frac{(q_B)_{i+1}^{t+1}+(q_B)_{i+1}^{t}}{2}, \quad (2.2.32)$$

in which the superscript and subscript correspond to temporal and spatial discretization indices, respectively. From (2.2.32), the following expression for the unknown Q_{i+1}^{t+1} is obtained

$$Q_{i+1}^{t+1}=\frac{\frac{\Delta t}{\Delta x}Q_{i}^{t+1}+\Lambda Q_{i+1}^{t}+\frac{\Delta t}{2}\left((q_B)_{i+1}^{t+1}+(q_B)_{i+1}^{t}\right)}{\frac{\Delta t}{\Delta x}+\Lambda}, \quad (2.2.33)$$

where

$$\Lambda=\alpha\beta\left(\frac{Q_{i+1}^{t}+Q_{i}^{t+1}}{2}\right)^{\beta-1}.$$

The solution progresses from the upstream to downstream.

For demonstration, we use data modified slightly from Chow et al. (1998, Example 9.4.1), where the rectangular channel is 24000-ft long (1 ft = 0.3048 m) and 200-ft wide, the Manning's n is 0.035, S_0 is 0.01, and $q_B = 0$. The initial flow is uniform at 2000 [ft^3/s]. Figure 2.6a shows the inflow hydrograph and two calculated, as well as the forward solution (i.e., routed hydrographs) at several cross sections along the channel. Figure 2.6b plots flow rate as a function of S_0, which shows that Q is a mild nonlinear function of S_0.

Now assume that the actual value of S_0 is unknown and we want to estimate it by using measurements of the flow rate $Q(x,t)$. All observations are taken at the

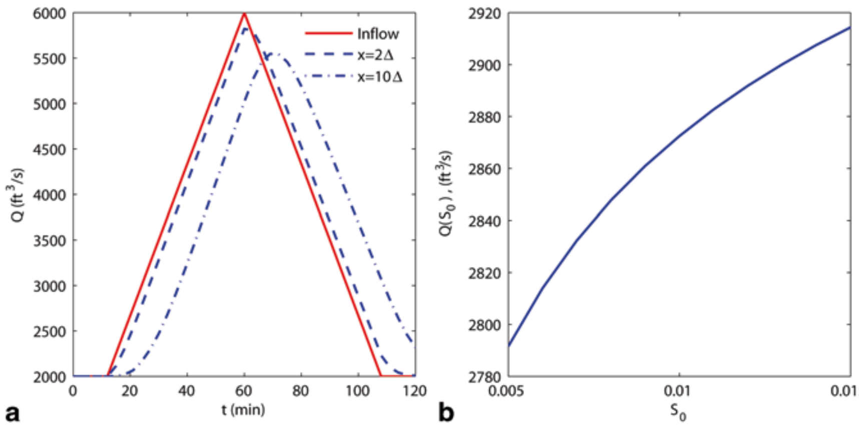


Fig. 2.6 **a** Inflow hydrograph and calculated hydrographs at downstream sections $x = 2000$ and 10000 ft (Δ = 1000 ft), respectively; **b** Plot of flow discharge rate as a function of S_0 at $x = 2000$ ft

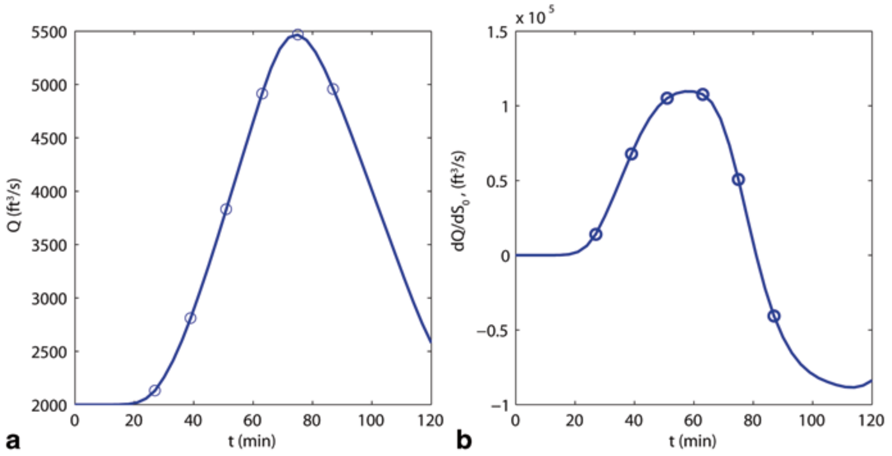


Fig. 2.7 **a** Flow rate observations (*circles*) used for estimating S_0, and **b** sensitivity dQ/dS_0 corresponding to plot (**a**)

station $x = 12000\,[\text{ft}]$ at six different times (circles in Fig. 2.7a). The successive linearization algorithm is used to estimate S_0 iteratively. Starting with an initial guess of $S_0 = 0.005$, we calculate the Jacobian $\mathbf{J}$ which, in this case, is a 6×1 vector because there is only one unknown parameter. Figure 2.7b shows the corresponding sensitivities for the initial iteration (circles). The algorithm converged to the "true" S_0 value of 0.01 in just three iterations. ■

The above example indicates that a nonlinear model can be identified accurately by linearization. However, when $\mathbf{u}_D(\boldsymbol{\theta})$ is highly nonlinear, as it is often the case in EWR modeling, the successive linearization process may not always converge to the inverse solution of the nonlinear model. In this case, we have to solve the nonlinear optimization problem (2.1.7) to find the inverse solution.

2.3 Nonlinear Model Identification

2.3.1 Inverse Solution and Optimization

In Sect. 2.2, the quasi-solution of a CIP is obtained from a minimization problem

$$\min_{\theta} S(\theta), \quad \theta \in P_{ad}, \tag{2.3.1}$$

where $S(\theta) = \left\| u_D(\theta) - u_D^{obs} \right\|$. Equation (2.3.1) is a typical nonlinear optimization problem. As an important mathematical tool, nonlinear optimization has found extensive applications in virtually all fields of science and engineering. For inverse solutions, however, problem (2.3.1) is often difficult to solve because of the complexity of $S(\theta)$ and the irregularity of P_{ad}.

According to the properties of objective functions and constraints, optimization problems can be classified in different ways, such as constrained vs. unconstrained, linear vs. nonlinear, smooth vs. nonsmooth, continuous vs. discrete, single vs. multiple objectives, and so on. The selection of an appropriate optimization algorithm is problem dependent. According to the level of difficulty, optimization problems can be roughly classified into:

- *Easy problems*: The objective function $S(\theta)$ is linear or quadratic and the admissible region P_{ad} is characterized by linear functions. In this case, algorithms of linear programming and quadratic programming can be used to obtain the solution effectively.
- *Slightly difficult problems*: $S(\theta)$ is convex, P_{ad} is either convex or unbinding. In this case, a local minimum can be found by an unconstrained optimization algorithm; the unique local minimum is also the global minimum.
- *Moderately difficult problems*: $S(\theta)$ is nonconvex or P_{ad} is nonconvex and binding, but the dimension of θ is not too high. In this case, a constrained and/or a global optimization algorithm can be used to find the global minimum with modest computational effort.
- *Highly difficult problems*: $S(\theta)$ and P_{ad} are highly nonlinear, and the dimension of θ is also high. In this case, global optimization algorithms may become ineffective and the computation may become intractable.

Depending on the complexity of the model and availability of data, the minimization problem derived from an inverse problem may be an easy one, a slightly difficult one, a moderately difficult one, or a highly difficult one. In EWR modeling, the optimization problem for inversion is often difficult because (i) the evaluation of $S(\theta)$ requires solution of the forward problem, which is time-consuming when the model equation consists of PDEs and the model scale (e.g., number of nodes) is large; (ii) the ill-posedness of the inverse problem makes the solution of the optimization problem nonunique (having multiple local minima) and unstable (sensitive to the observation error) because of insufficient data; and (iii) the dimension of θ is usually high for distributed systems.

Numerical optimization is a major tool for solving inverse problems. In this section, we will introduce the basic concepts and algorithms of *unconstrained optimization* for finding a local minimum. This type of algorithms is suitable for solving an extended well-posed CIP. To ensure that the unknown parameter is quasi-identifiable (see Sect. 2.1.3), we make the following assumptions:

1. The unknown parameter is within a convex region P_{ad} known a priori;
2. The quantity and quality of the observation data are sufficient such that $S(\theta)$ is a strictly convex function over the region.

These assumptions ensure that the boundary of P_{ad} is an unbinding constraint, only one local minimum exists in the region, and the unique local minimum is also the global minimum over the region. In other words, the inverse solution can be found by solving a *slightly difficult* optimization problem.

2.3.2 Basic Concepts of Numerical Optimization

2.3.2.1 Optimality Conditions

Let us first review the necessary and sufficient conditions for an interior point of $P_{ad} \subset \mathbb{R}^m$ to be a minimizer of an objective function $S(\boldsymbol{\theta}) \in \mathbb{R}$. When $S(\boldsymbol{\theta})$ is high-order differentiable in the neighborhood of a point $\boldsymbol{\theta}^* \in P_{ad}$, we have the following Taylor's expansion around $\boldsymbol{\theta}^*$

$$S(\boldsymbol{\theta}) - S(\boldsymbol{\theta}^*) = \mathbf{g}^T(\boldsymbol{\theta} - \boldsymbol{\theta}^*) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \mathbf{H}(\boldsymbol{\theta} - \boldsymbol{\theta}^*) + HOT, \tag{2.3.2}$$

where *HOT* represents high-order terms, $\mathbf{g}$ is the *gradient vector*

$$\mathbf{g} = \nabla S = \left(\frac{\partial S}{\partial \theta_1}, \frac{\partial S}{\partial \theta_2}, \ldots, \frac{\partial S}{\partial \theta_m}\right)^T, \tag{2.3.3}$$

and $\mathbf{H}$ is the *Hessian matrix* (see Appendix C)

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 S}{\partial \theta_1^2} & \dfrac{\partial^2 S}{\partial \theta_1 \partial \theta_2} & \cdots & \dfrac{\partial^2 S}{\partial \theta_1 \partial \theta_m} \\ \dfrac{\partial^2 S}{\partial \theta_1 \partial \theta_2} & \dfrac{\partial^2 S}{\partial \theta_2^2} & \cdots & \dfrac{\partial^2 S}{\partial \theta_2 \partial \theta_m} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \dfrac{\partial^2 S}{\partial \theta_1 \partial \theta_m} & \dfrac{\partial^2 S}{\partial \theta_2 \partial \theta_m} & \cdots & \dfrac{\partial^2 S}{\partial \theta_m^2} \end{bmatrix}. \tag{2.3.4}$$

In Taylor's expansion (2.3.2), all derivatives in $\mathbf{g}$ and $\mathbf{H}$ are evaluated at point $\boldsymbol{\theta}^*$. A point $\boldsymbol{\theta}^*$ is called a *stationary point* of $S(\boldsymbol{\theta})$ when $\mathbf{g}(\boldsymbol{\theta}^*) = \mathbf{0}$. If $S(\boldsymbol{\theta}^*) \leq S(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$ in a neighborhood of $\boldsymbol{\theta}^* \in P_{ad}$, then $\boldsymbol{\theta}^*$ is a local minimizer of $S(\boldsymbol{\theta})$. To determine whether a stationary point is also a minimizer, we must consider the second-order terms in (2.3.2). When $\mathbf{H}$ is positive definite at $\boldsymbol{\theta}^*$, then for any $\boldsymbol{\theta}$ close to $\boldsymbol{\theta}^*$, we must have $(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \mathbf{H}(\boldsymbol{\theta} - \boldsymbol{\theta}^*) > 0$; when $\mathbf{H}$ is positive semidefinite at $\boldsymbol{\theta}^*$, we must have $(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \mathbf{H}(\boldsymbol{\theta} - \boldsymbol{\theta}^*) \geq 0$ for any $\boldsymbol{\theta}$ close to $\boldsymbol{\theta}^*$ (see Appendix C). The necessary and sufficient conditions for $\boldsymbol{\theta}^*$ to be a local minimizer are thus given by

- Necessary conditions, $\mathbf{g}(\boldsymbol{\theta}^*) = \mathbf{0}$ and $\mathbf{H}(\boldsymbol{\theta}^*)$ is positive semi-definite;
- Sufficient conditions, $\mathbf{g}(\boldsymbol{\theta}^*) = \mathbf{0}$ and $\mathbf{H}(\boldsymbol{\theta}^*)$ is positive definite.

These conditions can be derived from (2.3.2) directly. In fact, if $\boldsymbol{\theta}^*$ is not a stationary point or $\mathbf{H}(\boldsymbol{\theta}^*)$ is not positive semidefinite, then in any neighborhood of $\boldsymbol{\theta}^*$, we can find two points $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ such that $S(\boldsymbol{\theta}_1) - S(\boldsymbol{\theta}^*)$ and $S(\boldsymbol{\theta}_2) - S(\boldsymbol{\theta}^*)$ have different signs; therefore, $\boldsymbol{\theta}^*$ cannot be a local minimizer. On the other hand, if $\boldsymbol{\theta}^*$ is a stationary point and $\mathbf{H}(\boldsymbol{\theta}^*)$ is positive definite, the second-order term determines that the right-hand side of (2.3.2) must be positive in a neighborhood of $\boldsymbol{\theta}^*$ and, thus, $\boldsymbol{\theta}^*$ must be a local minimizer.

The positive definiteness of $\mathbf{H}(\boldsymbol{\theta}^*)$ guarantees the local convexity of $S(\boldsymbol{\theta})$ at $\boldsymbol{\theta}^*$ and, thus, $\mathbf{g}(\boldsymbol{\theta}^*) = \mathbf{0}$ becomes the necessary and sufficient condition of minimization. If $S(\boldsymbol{\theta})$ is a differentiable convex function over a convex region, then $\boldsymbol{\theta}^*$ must be a global minimizer of the region.

2.3.2.2 Numerical Differentiation

When objective function $S(\boldsymbol{\theta})$ is given numerically by a computer code instead of by an analytical expression, its gradient vector and Hessian matrix can only be calculated approximately by numerical differentiation. In this case, we do not have a gradient equation $\mathbf{g}(\boldsymbol{\theta}) = \mathbf{0}$ for solving the stationary points. When the forward solution $\mathbf{u}_D(\boldsymbol{\theta})$ is obtained numerically using a computer code, the objective function, $S(\boldsymbol{\theta}) = \left\|\mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs}\right\|$, depends only on $\boldsymbol{\theta}$: for each input $\boldsymbol{\theta}$ to the computer model, we obtain a corresponding output value $S(\boldsymbol{\theta})$. Similarly, we can invoke the model to calculate the gradient vector at a given point $\boldsymbol{\theta}$, for example, by the finite difference approximation:

$$g_i(\boldsymbol{\theta}) = \left.\frac{\partial S}{\partial \theta_i}\right|_{\boldsymbol{\theta}} \approx \frac{S(\boldsymbol{\theta} + \Delta\theta_i \mathbf{e}_i) - S(\boldsymbol{\theta})}{\Delta\theta_i}, \quad i = 1, 2, \cdots, m, \tag{2.3.5}$$

where $\mathbf{e}_i$ is the unit vector along the i-th coordinate axis. Calculating the Hessian matrix of a numerical model by using second-order finite difference approximation is generally infeasible because of the resulting significant numerical error and high computational cost when the dimension of $\boldsymbol{\theta}$ is high. We will introduce various methods for differentiating a numerical model in Chap. 5. To speed up optimization, it is also possible to develop metamodels or reduced-order models using the original computer model. This latter topic will be covered in both Chaps. 8 and 10.

2.3.2.3 Iteration Approach

Numerical optimization uses an iteration process to search for a solution. An iteration process for minimization generally consists of three steps:

1. Choose a starting point $\boldsymbol{\theta}_0 \in P_{ad}$ (an initial guess of the inverse solution);
2. Generate an iteration sequence

$$\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_k, \boldsymbol{\theta}_{k+1}, \cdots \tag{2.3.6}$$

such that the value of the objective function is decreased gradually, namely, $S(\boldsymbol{\theta}_{k+1}) < S(\boldsymbol{\theta}_k)$ for all k;

3. Determine whether convergence has reached.

The iteration sequence (2.3.6) has the following general form

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \lambda_k \mathbf{d}_k, \tag{2.3.7}$$

where vector $\mathbf{d}_k$ is called a displacement direction, and λ_k is a step size along this direction. Different optimization algorithms use different methods to generate $\mathbf{d}_k$ and λ_k, as we will see later in Sect. 2.3.3.

2.3.2.4 Convergence and Stopping Criteria

The iteration sequence (2.3.6) generated by an algorithm is said to be convergent if it reaches a point $\boldsymbol{\theta}^*$ where the value $S(\boldsymbol{\theta}^*)$ is a minimum of the objective function. In other words, for any given numbers $\varepsilon > 0$ and $\delta > 0$, there is an integer K, such that

$$\left\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\right\| < \varepsilon \text{ and } S(\boldsymbol{\theta}_k) < S(\boldsymbol{\theta}^*) + \delta, \text{ for all } k > K. \tag{2.3.8}$$

The integer K is the number of iterations needed to reach an accuracy requirement of convergence. The speed of convergence is measured by the limit

$$\lim_{k\to\infty} \frac{\left\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}^*\right\|}{\left\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\right\|^\alpha} = \beta < \infty, \tag{2.3.9}$$

where α is called the *order of convergence*, and β is called the *convergence factor*. In particular, we have the following cases:

- Linear convergence, when $\alpha = 1$ and $0 < \beta < 1$;
- Superlinear convergence, when $\alpha = 1$ and $\beta = 0$; and
- Quadratic convergence, when $\alpha = 2$ and $0 < \beta < 1$.

We cannot use (2.3.8) directly in practice to determine the convergence and terminate the iteration because $\boldsymbol{\theta}^*$ is unknown. Instead, we have to resort to the following stopping criteria:

1. The displacement becomes small after several successive movements,

$$\left\|\boldsymbol{\theta}_{k+L} - \boldsymbol{\theta}_k\right\| < \varepsilon. \tag{2.3.10}$$

2. The value change of the objective function $S(\boldsymbol{\theta})$ becomes small after several successive movements,

$$S(\boldsymbol{\theta}_k) < S(\boldsymbol{\theta}_{k+L}) + \delta. \tag{2.3.11}$$

   In (2.3.10) and (2.3.11), $\varepsilon > 0$ and $\delta > 0$ are user-specified numbers, and $L \geq 1$ is a predetermined integer (e.g., the dimension of $\boldsymbol{\theta}$).

3. For inverse solution, we have to consider an additional criterion, namely, the value of data fitting residual becomes small,

$$S(\boldsymbol{\theta}_k) = \left\| \mathbf{u}_D(\boldsymbol{\theta}_k) - \mathbf{u}_D^{obs} \right\| < \eta, \tag{2.3.12}$$

   where $\eta > 0$ is an acceptable level of fitting residual measured in the observation space.

When the assumptions made at the end of Sect. 2.3.1 are valid and the number of iterations (K) is large enough, all three criteria listed above should be satisfied simultaneously. A quasi-solution is then obtained. In practice, especially in EWR modeling, however, the following cases may occur because of data and model errors:

1. Criteria 1 and 2 are satisfied but criterion 3 is not (i.e., the fitting residual is not acceptable). In this case, a local minimum may be found, but it is not the inverse solution, and the global minimum is needed.
2. Criteria 2 or 3 are satisfied but criterion 1 is not (i.e., the observation data are insensitive to the identified parameter). In this case, the quantity of observation data is insufficient for distinguishing the identified parameter from other significantly different parameters.
3. Criterion 3 cannot be satisfied regardless of the satisfaction of criteria 1 and 2. This case may be attributed to the bad quality of the observation data, but more often, the model structure error.

We can try to find the global minimum if any one of the three cases identified in the above occurs. But, in general, we have to consider (i) increasing the quantity of observation data, (ii) decreasing the observation error, (iii) using additional information related to the identified parameters, and finally, (iv) modifying the model structure. These topics will be covered in different chapters of this book.

2.3.3 Algorithms for Local Optimization

The algorithms reviewed in this section are designed for finding a local minimum without constraints. They can be sorted into three categories:

- An optimization algorithm is called a *zero-order method* if only the objective function values are utilized.

- An optimization algorithm is called a *first-order method* if the first-order derivatives of the objective function are also utilized.
- An optimization algorithm is called a *second-order method* if the second-order derivatives of the objective function are also utilized.

Generally speaking, the convergence speed of a first-order method is faster than that of a zero-order method, and the speed of a second-order method is faster than that of a first-order method. But, the efficiency of an optimization algorithm is determined by the total computational effort rather than the number of iterations. In other words, an algorithm that requires a larger number of iterations but less computational effort for one iteration may be more efficient than another one that needs fewer iterations but more computational effort for each iteration. The efficiency of an optimization algorithm is problem dependent. From the perspective of inversion, we often use “the number of evaluations of the objective function” or “the number of forward model runs” to gauge the efficiency of an optimization algorithm.

2.3.3.1 Search Along a Direction

Along a displacement $\mathbf{d}_k$, the optimal step size λ_k in (2.3.7) is the solution of the following 1-D optimization problem

$$\lambda_k = \arg\min f(\lambda), \text{ where } f(\lambda) = S(\boldsymbol{\theta}_k + \lambda \mathbf{d}_k) \text{ and } \underline{a} < \lambda < \overline{b}. \quad (2.3.13)$$

The range $(\underline{a}, \overline{b})$ of λ is determined by the admissible region of $\boldsymbol{\theta}$. When $f(\lambda)$ is convex in the range, the minimum must be in a bracket defined by three points $[a, c, b]$ such that $a < c < b$, and $f(c) \leq \min\,(f(a), f(b))$. If we take a new point d in the bracket and calculate $f(d)$, then either $f(c) \leq f(d)$ or $f(c) > f(d)$. For the former case, we can remove the interval $[d, b]$ and use $[a, c, d]$ as a new bracket; for the latter case, we can remove the interval $[a, c]$ and use $[c, d, b]$ as a new bracket (see Fig. 2.8). Repeating this process, we can obtain a sequence of brackets with smaller and smaller lengths and all of which contain the minimum. This process can be terminated when the length of a new bracket is less than a predetermined number ε. The general bracket process uses the following steps to generate a line search sequence:

1. Let $a = \underline{a}$ and $b = \overline{b}$;
2. If $b - a < \varepsilon$, end the search and let $\lambda = (b - a)/2$;
3. Otherwise, find two points c and d in $[a, b]$. If $f(c) \leq f(d)$, then let $b = d$; otherwise, let $a = c$. Go back to step 2.

Points c and d may be determined by different methods. In the *golden section search* method, the points are given by

$$c = b - \tau(b - c) \text{ and } d = a + \tau(b - a), \text{ where } \tau = \frac{\sqrt{5} - 1}{2} \approx 0.618. \quad (2.3.14)$$

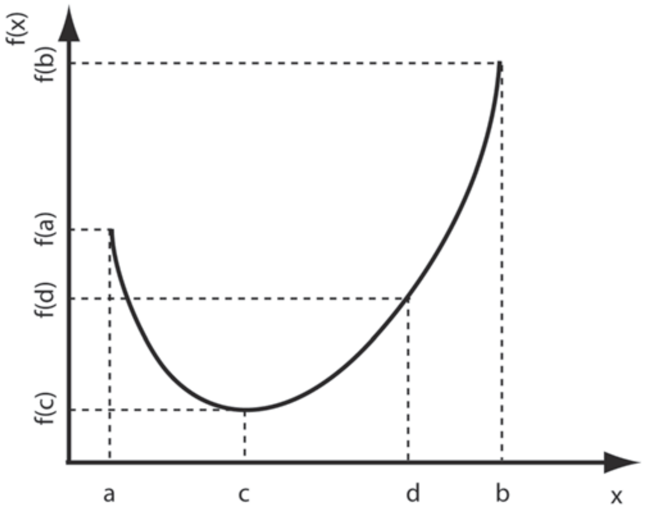


Fig. 2.8 Illustration of a single iteration in the branching process of line search, in which the new bracket is set to $[a, c, d]$

In the *quadratic interpolation method*, c is the mid point of the bracket and d is the minimizer of a quadratic polynomial defined by the three points $(a, f(a))$, $(c, f(c))$, and $(b, f(b))$

$$c = (b - a) / 2 \text{ and } d = c + \frac{c[f(a) - f(b)]}{2[f(b) - 2f(c) + f(a)]}. \tag{2.3.15}$$

The result of an exact line search gives the optimal step size for displacement along a given direction.

2.3.3.2 Search Along Coordinate Directions and the Powell Method

The simplest zero-order algorithm is the *one-at-a-time* line search method in which the search process is performed cycle by cycle. Each cycle consists of *m* displacements along the coordinate directions, $\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_m$. The process of one search cycle is described by

1. Let $\boldsymbol{\theta}_0$ be the search result of the last cycle ($\boldsymbol{\theta}_0$ is equal to the initial guess for the first cycle);
2. For $i = 1, 2, \cdots, m$, find $\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} + \lambda_i \mathbf{e}_i$, where λ_i is determined from $\lambda_i = \arg\min_{\lambda} S(\boldsymbol{\theta}_{i-1} + \lambda \mathbf{e}_i)$.

This method is simple but not useful because its convergence speed is slow. The *Powell method* is an improved zero-order algorithm, in which the displacement directions $\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_m$ used in a cycle are modified gradually from the coordinate directions to a set of *mutually conjugate* directions of the Hessian matrix $\mathbf{H}$, such that $\mathbf{d}_i^T \mathbf{H} \mathbf{d}_j \approx 0$ for $i \neq j$. It can be shown that when the objective function is quadratic and the line search is exact, the minimum can be found by *m* displacements, one by one along the conjugate directions (Press et al. 2007). In the Powell method, the conjugate directions are generated gradually during the iteration process. The following algorithm describes one search cycle of the Powell method:

1. Let $\boldsymbol{\theta}_0$ be the search result of the last cycle and let $\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_m$ be the search directions updated in the last cycle. For the first cycle, $\boldsymbol{\theta}_0$ is the initial guess and $\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_m$ are set to the coordinate directions.
2. For $i = 1, 2, \cdots, m$, find $\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} + \lambda_i \mathbf{d}_i$, where $\lambda_i = \arg\min_{\lambda} S(\boldsymbol{\theta}_{i-1} + \lambda \mathbf{d}_i)$.
3. Let $\mathbf{d}_{m+1} = \boldsymbol{\theta}_m - \boldsymbol{\theta}_0$ and find $\boldsymbol{\theta}_{m+1} = \boldsymbol{\theta}_m + \lambda_{m+1} \mathbf{d}$, where $\lambda_{m+1} = \arg\min_{\lambda} S(\boldsymbol{\theta}_m + \lambda \mathbf{d})$.
4. Use $\boldsymbol{\theta}_{m+1}$ as the result of search of this cycle and update the search directions for the next cycle by letting $\mathbf{d}_i = \mathbf{d}_{i+1}$ for $i = 1, 2, \cdots, m$.

In this process, after *m* displacements are completed one by one along the search directions, an additional line search along a new direction $\mathbf{d}_{m+1} = \boldsymbol{\theta}_m - \boldsymbol{\theta}_0$ is used to generate the search results of this cycle, and the search directions are updated for the next cycle. Therefore, one cycle in the Powell method contains $m+1$ displacements, and the search directions gradually become mutually conjugate with the increase of the number of iterations.

2.3.3.3 Steepest Decent and Conjugate Gradient Methods

The *steepest descent method* is one of the oldest first-order optimization methods. For the kth iteration, the new displacement direction is given by $\mathbf{d}_k = -\mathbf{g}(\boldsymbol{\theta}_k)$, which is the negative gradient direction or the steepest descent direction of $S(\boldsymbol{\theta})$ at $\boldsymbol{\theta}_k$. The step size λ_k is determined by a line search along this direction. When $S(\boldsymbol{\theta})$ is given numerically, its gradient $\mathbf{g}(\boldsymbol{\theta}_k)$ can be calculated by the finite difference approximation as shown in (2.3.5), which requires solving the forward model $m+1$ times (recall that m is the dimension of $\boldsymbol{\theta}$). The steepest descent method has been widely used for inverse solutions. Its convergence speed, however, is slow when the initial guess $\boldsymbol{\theta}_0$ is not in the neighborhood of the minimizer.

The *conjugate gradient method* is an improved first-order algorithm, in which the search directions used in each cycle are modified gradually to the conjugate directions. In this method, the search process for one cycle is given by the following algorithm:

1. Let $\boldsymbol{\theta}_0$ be the search result of the last cycle.
2. Let $\mathbf{d}_1 = -\mathbf{g}(\boldsymbol{\theta}_0)$ and find $\boldsymbol{\theta}_1 = \boldsymbol{\theta}_0 + \lambda_1 \mathbf{d}_1$, where $\lambda_1 = \arg\min_{\lambda} S(\boldsymbol{\theta}_0 + \lambda \mathbf{d}_1)$.

3. For $i = 2, 3, \cdots, m$, let $\mathbf{d}_i = -\mathbf{g}_{i-1} + \beta_{i-1}\mathbf{d}_{i-1}$, where $\mathbf{g}_{i-1} = \mathbf{g}(\theta_{i-1})$, and find $\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} + \lambda_i \mathbf{d}_i$, where $\lambda_i = \arg\min_{\lambda} S(\boldsymbol{\theta}_{i-1} + \lambda \mathbf{d}_i)$.
4. Report $\boldsymbol{\theta}_m$ as the search result of this cycle.

In this process, the current search direction is a linear combination of the current steepest decent direction and the direction of the previous displacement. The coefficient β_i is selected to make the search directions conjugate. In the F-R algorithm (Fletcher and Reeves 1964; Press et al. 2007), β_i is given by

$$\beta_i = \left\|\mathbf{g}_i\right\|^2 / \left\|\mathbf{g}_{i-1}\right\|^2,$$

while in the P-R algorithm (Polak and Ribiere 1969; Press et al. 2007), β_i is given by

$$\beta_i = \mathbf{g}_i \cdot \Delta\mathbf{g}_i / \left\|\mathbf{g}_{i-1}\right\|^2,$$

where $\Delta\mathbf{g}_i = \mathbf{g}_i - \mathbf{g}_{i-1}$. In general, the P-R algorithm is more efficient than the F-R algorithm.

2.3.3.4 Newton and Quasi-Newton Methods

The *Newton method* is a second-order optimization algorithm that requires calculation of the Hessian matrix. When we have a point $\boldsymbol{\theta}_k$ in the iteration sequence, we expect to find a point $\boldsymbol{\theta}_{k+1}$ such that the gradient $\mathbf{g}(\boldsymbol{\theta}_{k+1}) = \mathbf{0}$ (i.e., satisfying the necessary condition of minimization). Using Taylor's expansion of $\mathbf{g}(\boldsymbol{\theta})$ at point $\boldsymbol{\theta}_k$, we have the following equation

$$\mathbf{0} = \mathbf{g}(\boldsymbol{\theta}_{k+1}) = \mathbf{g}(\boldsymbol{\theta}_k) + \mathbf{H}(\boldsymbol{\theta}_k)(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k) + HOT. \tag{2.3.16}$$

Ignoring the high-order terms and denoting $\mathbf{g}(\boldsymbol{\theta}_k)$ by $\mathbf{g}_k$ and $\mathbf{H}(\boldsymbol{\theta}_k)$ by $\mathbf{H}_k$, respectively, we obtain

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mathbf{H}_k^{-1}\mathbf{g}_k. \tag{2.3.17}$$

This equation can be used to generate the iteration sequence. Compared to the general equation (2.3.7), we see that the displacement direction in the Newton method is given by $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$ (called the Newton direction) and the step size λ_k is always equal to one. The Newton algorithm converges fast and no line search is needed. When the objective function is quadratic, the minimum can be found by one displacement from any point. For the general case, of course, an iteration process is needed. Finding the Newton direction for each step of iteration requires the calculation of the Hessian matrix which, unfortunately, is computationally prohibitive when the forward run of a model is expensive. Therefore, the Newton method is not practical from the perspective of inverse solutions.

To avoid the calculation of the Hessian matrix, several *quasi-Newton methods* have been proposed. The common idea behind these methods is the replacement of the inverse of the Hessian matrix, $\mathbf{H}_k^{-1}$, by a symmetric and positive definite matrix, $\mathbf{M}_k$, which is modified gradually during the iteration process. In a quasi-Newton method, the process of generating $\boldsymbol{\theta}_{k+1}$ from $\boldsymbol{\theta}_k$ can be summarized as

1. Let $\mathbf{d}_k = -\mathbf{M}_k\mathbf{g}_k$; if $k = 0$, set $\mathbf{M}_0 = \mathbf{I}$ (the identity matrix);
2. Find $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \lambda_k\mathbf{d}_k$, where $\lambda_k = \arg\min_{\lambda} S(\boldsymbol{\theta}_k + \lambda\mathbf{d}_k)$;
3. Update $\mathbf{M}_k$ to $\mathbf{M}_{k+1}$ for the next iteration.

An often used quasi-Newton algorithm, which was presented by Broyden, Fletcher, Goldfarb, and Shanno and commonly known as the BFGS method (Fletcher 1970), suggests the following updating formula:

$$\begin{aligned}\mathbf{M}_{k+1} = \mathbf{M}_k &+ \left(1 + \frac{\Delta\mathbf{g}_k^T\mathbf{M}_k\Delta\mathbf{g}_k}{\Delta\boldsymbol{\theta}_k^T\Delta\mathbf{g}_k}\right)\frac{\Delta\boldsymbol{\theta}_k\Delta\boldsymbol{\theta}_k^T}{\Delta\boldsymbol{\theta}_k^T\Delta\mathbf{g}_k} \\ &- \left(\frac{\Delta\boldsymbol{\theta}_k\Delta\mathbf{g}_k^T\mathbf{M}_k + \mathbf{M}_k\Delta\mathbf{g}_k\Delta\boldsymbol{\theta}_k^T}{\Delta\boldsymbol{\theta}_k^T\Delta\mathbf{g}_k}\right),\end{aligned} \tag{2.3.18}$$

where $\Delta\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k$ and $\Delta\mathbf{g}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$. The convergence speed of the BFGS method is superlinear when the line searches are exact. Detailed discussions on the quasi-Newton methods and more updating formulae can be found in (Fletcher 1987; Luenberger and Ye 2008; Nocedal and Wright 2006).

A variation of the BFGS method is the limited memory BFGS (L-BFGS) method, in which the matrix $\mathbf{M}_{k+1}$ is not calculated by (2.3.18); instead, it is obtained implicitly by using $\boldsymbol{\theta}_k, \boldsymbol{\theta}_{k-1}, \cdots, \boldsymbol{\theta}_{k-r+1}$ and $\mathbf{g}_k, \mathbf{g}_{k-1}, \cdots, \mathbf{g}_{k-r+1}$ (i.e., the results of r previous iterations, where r is an integer less than 10). Because the full matrix $\mathbf{M}_{k+1}$ is not formed and stored, L-BFGS is more efficient than BFGS when the dimension of $\boldsymbol{\theta}$ is high. Detailed discussions and algorithms of L-BFGS can be found in Nocedal and Wright (2006).

Both BFGS and steepest descent are available as options for Hessian update in the Matlab function, `fminunc`, which can solve unconstrained nonlinear optimization problems. SciPy provides an implementation of BFGS in function `fmin_bfgs`, L-BFGS in `fmin_l_bfgs_b`, and conjugate-gradient (P-R algorithm) in `fmin_cg`. The R language command `optim` is a general-purpose optimization routine that implements BFGS, L-BFGS, and conjugate gradient (F-R algorithm) methods.

2.3.3.5 Inexact Line Search

All gradient-based algorithms of optimization, including various quasi-Newton methods, require using exact line search to assure convergence when the objective function is nearly quadratic. But, this requirement may make an algorithm inefficient, especially when it is used to solve an inverse problem and the forward solution is time-consuming. In fact, when the current point in the search sequence is not close to the minimum, an exact line search is not helpful for fast convergence. Therefore, appropriate stopping criteria for inexact line search have been developed that can increase the efficiency while ensuring the convergence of an iteration process. The following strong Wolfe conditions have been incorporated into most algorithms in common optimization software packages:

$$S(\boldsymbol{\theta}_k + \lambda_k \mathbf{d}_k) \le S(\boldsymbol{\theta}_k) + c_1 \lambda_k \mathbf{d}_k^T \mathbf{g}(\boldsymbol{\theta}_k), \tag{2.3.19}$$

$$\left|\mathbf{d}_k^T \mathbf{g}(\boldsymbol{\theta}_k + \lambda_k \mathbf{d}_k)\right| \le c_2 \left|\mathbf{d}_k^T \mathbf{g}(\boldsymbol{\theta}_k)\right|, \tag{2.3.20}$$

where $0 < c_1 < c_2 < 1$. Note that $\mathbf{d}_k^T \mathbf{g}(\boldsymbol{\theta}_k) < 0$ because $\mathbf{d}_k$ is a descent direction. For a step size λ_k, the first condition (2.2.19) ensures that the value of the objective function decreases sufficiently, and the second condition (2.3.20) ensures that the length of the gradient reduces sufficiently. The coefficient c_1 is usually chosen to be close to zero, while c_2 is close to one. For example, BFGS method sets $c_1 = 10^{-4}$ and $c_2 = 0.9$, whereas the conjugate gradient method sets $c_1 = 10^{-4}$ and $c_2 = 0.1$ (Nocedal and Wright 2006).

Example 2.7 *Estimation of parameters in a soil water retention model*
Modeling water movement in the vadose zone requires knowledge of soil water retention characteristics. Simply speaking, a soil water retention model describes how water content varies with pressure head and, therefore, reflects a soil's water-holding capacity. A widely used soil water retention model is the van Genuchten model (Van Genuchten 1980), which expresses saturation as a nonlinear function of pressure head

$$S_e = \left[1 + (\alpha|\psi|)^n\right]^{-m}, \; S_e = \frac{\theta_w - \theta_r}{\theta_s - \theta_r}, m = 1 - 1/n, \tag{2.3.21}$$

where θ_w is volumetric water content; θ_s and θ_r denote the saturated and residual water content, respectively; S_e is a normalized quantity called effective water saturation; ψ is pressure head [L]; and α and n are fitting parameters that characterize the shape of the retention curve.

In this example, we would like to estimate parameters of the van Genuchten's model using measurements of ψ and θ_w. We assume that θ_s and θ_r are known and only parameters α and n need to be estimated (i.e., $\boldsymbol{\theta} = \{\alpha, n\}$). The experimental data reported in van Genuchten (1980) are used here for demonstration (open circles in Fig. 2.9a). The optimization problem (2.3.1) can be cast as

$$\min_{\boldsymbol{\theta}=\{\alpha,n\}} S\{\boldsymbol{\theta}\}, \tag{2.3.22}$$

in which the objective function is the discrepancy (L_2-norm) between observed and simulated water content,

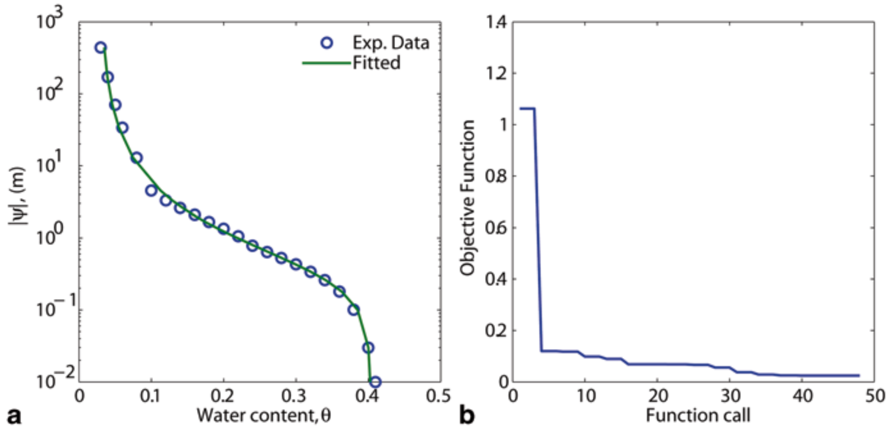


Fig. 2.9 **a** Experimental data (o) and the fitted van Genuchten model (*solid line*); and **b** convergence history of the minimization process

$$S = \left\| \boldsymbol{\theta}_w^{obs} - \boldsymbol{\theta}_w \right\|_2^2, \tag{2.3.23}$$

where $\boldsymbol{\theta}_w^{obs}$ and $\boldsymbol{\theta}_w$ are observed and simulated water content values at different pressure heads. ■

The unconstrained minimization problem (2.3.22) was solved using the Matlab function, `fminunc`, which implements a quasi-Newton method for medium-scale problems. Given initial guesses of $\alpha = 2$ and $n = 1$, the final solution returned by `fminunc` is $\alpha = 2.90$ and $n = 1.58$. The fitted van Genuchten model is shown in Fig. 2.9a (solid line). Figure 2.9b shows the convergence history of the solution process, from which a monotonic reduction in objective function value can be observed.

The optimization problem has multiple local minima and, thus, the quasi-solution returned by local optimization algorithms is dependent on initial guesses. For example, if the initial guesses are changed to $\alpha = 0.8$ and $n = 0.5$, the estimated parameters become $\alpha = 2.75$ and $n = 1.60$.

2.3.4 Norm Selection

The objective function $S(\theta) = \left\| u_D(\theta) - u_D^{obs} \right\|$ for inverse solutions is a misfit function, depending on which norm is furnished in the observation space $\mathbb{F}$ (see Appendix A). The selection of different norms may affect both the accuracy and stability of inverse solutions. When observation data are provided over a continuous spatial and/or temporal region (Ω), the most popular and also the one that we have used extensively so far is the L_2-norm, also known as the Euclidean distance

$$\left\| u_D(\theta) - u_D^{obs} \right\|_2 = \left\{ \int_\Omega \left[u_D(\theta, x) - u_D^{obs}(x) \right]^2 dx \right\}^{1/2}. \tag{2.3.24}$$

L_2-norm is a special case of the general L_p-norm (also known as the Minkowski distance) defined by

$$\left\| u_D(\theta) - u_D^{obs} \right\|_p = \left\{ \int_\Omega \left| u_D(\theta, x) - u_D^{obs}(x) \right|^p dx \right\}^{1/p}, \tag{2.3.25}$$

where $1 \le p < \infty$. In particular, we have the L_1-norm (also known as the city-block distance)

$$\left\| u_D(\theta) - u_D^{obs} \right\|_1 = \int_\Omega \left| u_D(\theta, x) - u_D^{obs}(x) \right| dx. \tag{2.3.26}$$

When the observation error is normally distributed with zero mean, we will see in Chap. 4 that the quasi-solution θ_{qs} associated with the L_2-norm is an unbiased estimation of the true parameter θ^t. This is one of the most important advantages of using L_2-norm. However, when the observation error is not normally distributed or when outliers exist, the use of L_2-norm tends to amplify their effect and lead to an unacceptable inverse solution. In this case, it is better to use the more robust L_1-norm.

When only n observation data are available, the observation space $\mathbb{F}$ reduces to an n-dimensional space, and the L_2- and L_1-norm become, respectively,

$$\left\| \mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs} \right\|_2^2 = \sum_{i=1}^{n} [u_{D,i}(\boldsymbol{\theta}) - u_{D,i}^{obs}]^2, \tag{2.3.27}$$

and

$$\left\| \mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs} \right\|_1 = \sum_{i=1}^{n} \left| u_{D,i}(\boldsymbol{\theta}) - u_{D,i}^{obs} \right|, \tag{2.3.28}$$

where $u_{D,i}(\boldsymbol{\theta})$ is the ith component of $\mathbf{u}_D(\boldsymbol{\theta})$ corresponding to the ith observation $u_{D,i}^{obs}$. If we want to minimize the maximal misfit error, the following L_∞-norm can be used:

$$\left\| \mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs} \right\|_\infty = \max_{1 \le i \le n} \left| u_{D,i}(\boldsymbol{\theta}) - u_{D,i}^{obs} \right|. \tag{2.3.29}$$

In addition to L_p-norms, other functions (not necessarily norms) can also be used to measure the misfit error. For example, the Kullback–Leibler (KL) misfit function is defined by

$$K\left(\mathbf{u}_D^{obs}, \mathbf{u}_D(\boldsymbol{\theta}) \right) = \sum_{i=1}^{n} u_{D,i}^{obs} \left[\log u_{D,i}^{obs} - \log u_{D,i}(\boldsymbol{\theta}) \right]. \tag{2.3.30}$$

In the KL-misfit function, the misfit errors are measured in the logarithm scale. As a result, the contribution of a misfit error is decreased when it is associated with a large observation value. To balance this impact, observation values are used as weights.

Weights can also be applied to an L_p-norm. For example, the weighted L_2-norm is defined by

$$\left\| \mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs} \right\|_{\mathbf{W}}^2 = \sum_{i=1}^{n} w_i^2 [u_{D,i}(\boldsymbol{\theta}) - u_{D,i}^{obs}]^2, \tag{2.3.31}$$

where $\{w_1, w_2, \cdots, w_n\}$ is a set of weighting coefficients. The generalized least square (GLS) norm is defined by

$$\left\| \mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs} \right\|_{\mathbf{W}}^2 = \left[\mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs} \right]^T \mathbf{W} \left[\mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs} \right] \tag{2.3.32}$$

where $\mathbf{W}$ is an $n \times n$ positive-definite symmetric matrix.

The advantage of using a weighted norm is that the relative importance of each observation can be adjusted. In practice, we often use the following rules to determine the weighting coefficients:

- More accurate observations are assigned with larger weights, while less accurate observations are assigned with smaller weights.
- Observations that represent the state of the system in a large spatial and/or temporal region are assigned with larger weights, while smaller weights are assigned to observations that are dense in a small region.
- Observations that are sensitive to important model parameters are assigned to larger weights, while smaller weights are assigned to others.

In Chap. 4, we will learn how to find the optimal weighting coefficients when observation error statistics are known.

Example 2.8 *Parameter estimation using different norms*
We illustrate the effect of norm selection on parameter estimation using the nonlinear optimization problem considered in Example 2.7. In addition to L_2-norm shown in (2.3.23), both L_1- and L_∞-norm are considered

$$S = \left\| \boldsymbol{\theta}_w^{obs} - \boldsymbol{\theta}_w \right\|_1 \text{ and } S = \left\| \boldsymbol{\theta}_w^{obs} - \boldsymbol{\theta}_w \right\|_\infty .$$

Starting with the same initial guesses as before (i.e., $\alpha = 2,\ n = 1$), different solutions are obtained (see the two columns under the header "Original" in Table 2.1). In general, the difference in estimated α values is larger than that in n because the latter appears as an exponent term. Recall that L_∞-norm attempts to minimize the maximal misfit and thus its results deviate more from the other two norms. Figure 2.10 plots the three solutions (lines) against experimental data (circles).

To test the effect of measurement noise, 5 % multiplicative random measurement error is applied to the original pressure head data. In this case, the L_2-norm results are

Table 2.1 van Genuchten parameters estimated using different norms

Norm	Original		5% error		Outliers	
	α	n	α	n	α	n
L_1	2.95	1.57	3.06	1.54	2.97	1.57
L_2	2.90	1.58	2.92	1.57	3.44	1.54
L_∞	2.48	1.64	2.60	1.62	2.34	1.63

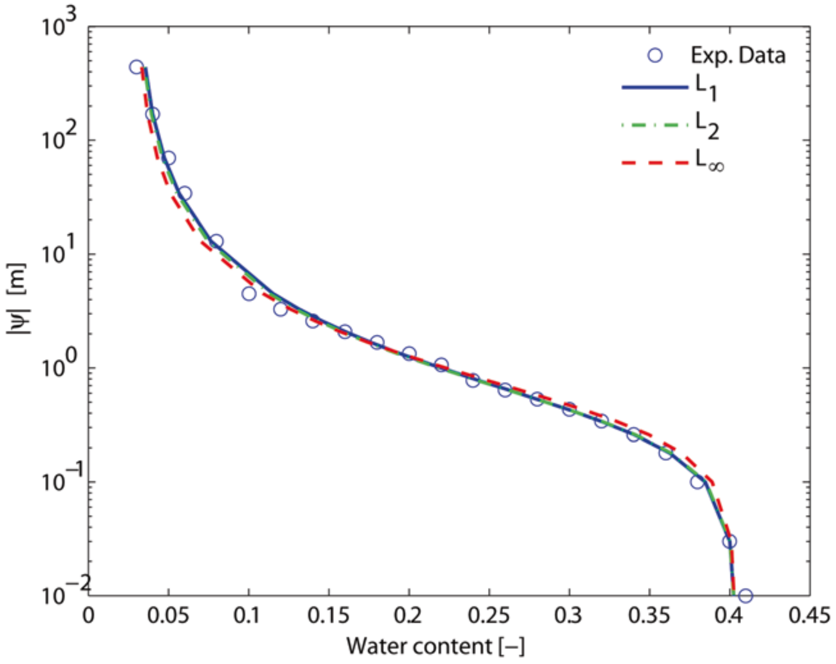


Fig. 2.10 Comparison of fitted van Genuchten models using different norms

slightly changed from its original estimates (columns under 5% error in Table 2.1). However, the L_2 -norm is more sensitive to outliers. In the next test, three pressure head values are effectively made as "outliers" (columns under "Outliers" in Table 2.1), causing L_2 -norm results to show significant deviation from its original values.

2.4 The Gauss–Newton Method

2.4.1 The Gauss–Newton Method for Least Squares

For inverse solution, the objective function $S(\boldsymbol{\theta})$ can be evaluated by the following general process

$$\boldsymbol{\theta} \in P_{ad} \xrightarrow[\mathcal{M}]{} \mathbf{u}(\boldsymbol{\theta}) \xrightarrow[\mathcal{D}]{} \mathbf{u}_D(\boldsymbol{\theta}) \xrightarrow[\mathcal{F}]{} S(\boldsymbol{\theta}) \tag{2.4.1}$$

where $\mathcal{M}$ is the forward solution, $\mathcal{D}$ is an observation design, and $\mathcal{F}$ measures the misfit to the observed data. All optimization algorithms introduced in Sect. 2.3.3 can then be implemented for finding the inverse solution, no matter which norm is selected or how a misfit function is defined.

In case of using L_2-norm, however, the Gauss–Newton method provides an efficient optimization algorithm, in which the Hessian matrix is calculated approximately by using the Jacobian. Let us consider the least squares problem given by

$$\min_{\boldsymbol{\theta} \in P_{ad}} S(\boldsymbol{\theta}), \text{ where } S(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n} f_i^2(\boldsymbol{\theta}) \text{ and } f_i(\boldsymbol{\theta}) = \left[u_{D,i}(\boldsymbol{\theta}) - u_{D,i}^{obs}\right]. \tag{2.4.2}$$

From (2.4.2), we can calculate the first-order derivatives of $S(\boldsymbol{\theta})$ by

$$\frac{\partial S}{\partial \theta_p} \approx \sum_{i=1}^{n} f_i \frac{\partial f_i}{\partial \theta_p}, \quad (p = 1, 2, \cdots, m), \tag{2.4.3}$$

and the second-order derivatives of $S(\boldsymbol{\theta})$ by

$$\frac{\partial^2 S}{\partial \theta_p \partial \theta_q} = \sum_{i=1}^{n} \left(\frac{\partial f_i}{\partial \theta_p} \frac{\partial f_i}{\partial \theta_q} + f_i \frac{\partial^2 f_i}{\partial \theta_p \partial \theta_q} \right). \tag{2.4.4}$$
$$(p = 1, 2, \cdots, m; \quad q = 1, 2, \cdots, m)$$

In (2.4.4), the second-order term in the right-hand side summation may be ignored because the value of fitting residual $f_i(\boldsymbol{\theta})$ is small when $\boldsymbol{\theta}$ is not too far from the minimizer. Thus, we have

$$\mathbf{H} = \left[\frac{\partial^2 S}{\partial \theta_p \partial \theta_q} \right]_{m \times m} \approx \left[\sum_{i=1}^{n} \frac{\partial f_i}{\partial \theta_p} \frac{\partial f_i}{\partial \theta_q} \right]_{m \times m} = \mathbf{J}_D^T \mathbf{J}_D, \tag{2.4.5}$$

where

$$\mathbf{J}_D = \left[\frac{\partial u_{D,i}}{\partial \theta_j} \right]_{n \times m} \tag{2.4.6}$$

is the Jacobian or the sensitivity matrix of observations with respect to the parameter. Substituting (2.4.5) into the Newton algorithm (2.3.17) and noting that the gradient in (2.4.3) can be represented by $\mathbf{g} = \mathbf{J}_D^T \mathbf{f}$, where $\mathbf{f} = (f_1, f_2, \cdots, f_n)^T$, we have the following iteration sequence

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - (\mathbf{J}_{D,k}^T \mathbf{J}_{D,k})^{-1} \mathbf{J}_{D,k}^T \mathbf{f}_k, \tag{2.4.7}$$

where $\mathbf{J}_{D,k} = \mathbf{J}_D(\boldsymbol{\theta}_k)$ and $\mathbf{f}_k = \mathbf{f}(\boldsymbol{\theta}_k)$. The displacement direction

$$\Delta\boldsymbol{\theta}_k = -(\mathbf{J}_{D,k}^T \mathbf{J}_{D,k})^{-1} \mathbf{J}_{D,k}^T \mathbf{f}_k \tag{2.4.8}$$

is called the Gauss–Newton direction. When the weighted L_2 -norm in (2.3.31) is used, it is easy to show that the Hessian matrix in (2.4.5) is replaced by $\mathbf{H} \approx \mathbf{J}_D^T \mathbf{W}^T \mathbf{W} \mathbf{J}_D$, where $\mathbf{W}$ is an $n \times n$ diagonal matrix with $w_1, w_2, \cdots, w_n$ as entries, and the Gauss–Newton direction is replaced by

$$\Delta\boldsymbol{\theta}_k = -(\mathbf{A}_k^T \mathbf{A}_k)^{-1} \mathbf{A}_k^T \mathbf{f}_k, \tag{2.4.9}$$

where $\mathbf{A}_k = \mathbf{W}\mathbf{J}_D(\boldsymbol{\theta}_k)$. For a linear model, (2.4.3) is exact, the Gauss–Newton direction becomes the Newton direction, and the inverse solution can be obtained by only one step along this direction. For a nonlinear model, however, (2.4.3) may contain significant error and the Gauss–Newton direction may not be a good approximation of the Newton direction. In this case, the following difficulties are often encountered: first, the iteration process may become divergent. We may see that the values of the objective function at an updated point are increased rather than decreased $(S(\boldsymbol{\theta}_{k+1}) > S(\boldsymbol{\theta}_k))$; second, the matrix $\mathbf{J}_D^T \mathbf{J}_D$ may become singular or nearly singular. The Gauss–Newton direction thus becomes undetermined and unstable, and the updated point $\boldsymbol{\theta}_{k+1}$ may be even out of the admissible region. Several modified Gauss–Newton methods have been developed to avoid these difficulties.

2.4.2 Modified Gauss–Newton Methods

In the Gauss–Newton method, the step size is always equal to one. The first modification is to add a line search in each step of iteration to assure that the value of the objective function is decreased during the iteration process. The iteration sequence still has the general form

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \lambda_k \mathbf{d}_k, \tag{2.4.10}$$

but the displacement direction $\mathbf{d}_k$ becomes the Gauss–Newton direction and λ_k is determined by a line search along this direction. This method is simple but not efficient because too many additional function evaluations are needed for the line searches.

The second method modifies the displacement direction according to

$$\mathbf{d}_k = \begin{cases} \Delta\boldsymbol{\theta}_k, \ \text{the Gauss–Newton direction when } \mathbf{g}_k^T \Delta\boldsymbol{\theta}_k < 0 \\ -\mathbf{g}_k, \ \text{the steepest descent direction when } \mathbf{g}_k^T \Delta\boldsymbol{\theta}_k \geq 0 \end{cases}. \tag{2.4.11}$$

Equation (2.4.11) implies that when the calculated Gauss–Newton direction is not consistent with the descent direction, it is replaced by the latter to assure that the value of the objective function is decreased during the iteration. This modification drops the convergence rate to a lower order.

The Levenberg–Marquardt method is another popular way to assure the convergence of the iteration process, in which the displacement in the Gauss–Newton algorithm (2.4.9) is replaced by

$$\Delta\boldsymbol{\theta}_k = -(\mathbf{A}_k^T\mathbf{A}_k + \lambda\mathbf{I})^{-1}\mathbf{A}_k^T\mathbf{f}_k, \tag{2.4.12}$$

where $\mathbf{I}$ is the identity matrix, and λ is a coefficient. When $\lambda = 0$, the Levenberg–Marquardt method becomes the Gauss–Newton method, while when λ is large, $\Delta\boldsymbol{\theta}_k$ turns to the steepest descent direction and its size approaches zero. Therefore, the descent condition $S(\boldsymbol{\theta}_{k+1}) < S(\boldsymbol{\theta}_k)$ can always be expected by increasing the value of λ. If this condition is not satisfied for an initial value of λ, then λ is multiplied by a factor β (e.g., $\beta = 10$) and $\Delta\boldsymbol{\theta}_k$ is recalculated until the descent condition is satisfied. In general, the Levenberg–Marqurdt method is more efficient than the line search method.

Example 2.9 *Parameter estimation using Gauss–Newton method*
Let us consider the minimization problem (2.3.22) in Example 2.7 and solve the problem using algorithms introduced in this section. The Matlab function `lsqnonlin` is used, which offers three different choices of algorithms, Gauss–Newton, Levenberg–Marquardt, and conjugate-gradient. Starting with the initial guesses $\alpha = 2,\ n = 1$, all three algorithms give the same solution $\alpha = 2.9,\ n = 1.58$ at the end. The number of iterations/functions calls for Gauss–Newton, Levenberg–Marquardt and conjugate-gradient are 5/29, 7/24, and 9/30, respectively. Figure 2.11

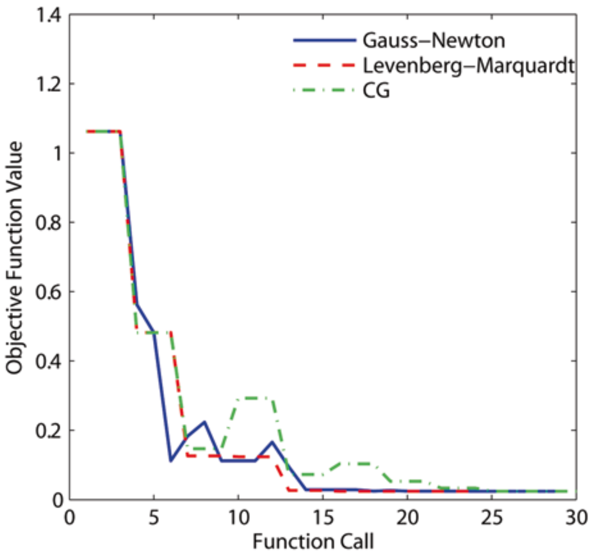


Fig. 2.11 Comparison of convergence histories of three nonlinear optimization algorithms

compares the convergence history of the three methods. The Levenberg–Marquardt method has a steady decrease in objective function values over the iterations, whereas the other two methods oscillate somewhat during the iterations. Both the Gauss–Newton and Levenberg–Marquardt methods show faster convergence speed than the conjugate-gradient method.

2.4.3 Application to Inverse Solution

In Chap. 4, we will show that when the observation error is normally distributed, the weighted L_2-norm is the most acceptable selection for measuring the fitting residual in the observation space. Therefore, most software packages choose a modified Gauss–Newton method, the Levenberg–Marquardt algorithm, for solving the CIP. The core code of the algorithm is short and easy to combine with any forward solution code. The inverse solution code uses the forward solution code to calculate the fitting residual in (2.4.2) and the sensitivity matrix in (2.4.6), and then uses a linear system solver to obtain an updated inverse solution (2.4.12) in each iteration. With a software package, the user only needs to determine which model parameters are to be identified and what are their guessed values and admissible ranges.

Example 2.10 *Using the Gauss–Newton method for inverse solution*
Figure 2.12 shows a hypothetical aquifer *ABCD*, 1200 [m] long, 600 [m] wide, and 50 [m] thick. The head at boundary section *AB* remains constant at 100 [m], while all other boundary sections are no-flow boundaries. The initial head is 100 [m] everywhere. A recharge well is located at • with injection rate 1000 [m^3/day] and an extraction well is located at o with a pumping rate of 4000 [m^3/day]. Four observation wells are located at × where the head is measured three times at 0.1, 0.5, 1.0 [day]. The total number of observation data is thus equal to 12. The aquifer consists of two homogeneous zones (see Fig. 2.12) characterized by different values of hydraulic conductivity K_1, K_2, and storage coefficient S_1, S_2. Our purpose is to estimate the four groundwater flow parameters with the observation data.

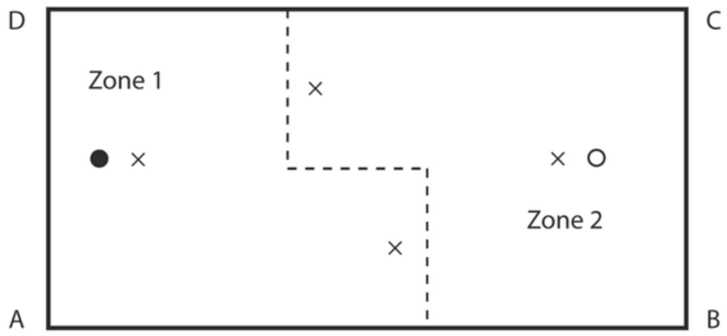


Fig. 2.12 Configuration of the hypothetical aquifer

Table 2.2 Results of the inverse solution obtained by the Gauss–Newton method

	True	*Initial*	*It.-1*	*It.-2*	*It.-3*	*It.-4*	$\sigma = 0.1$	$\sigma = 0.5$
K_1	5.00	3.00	5.25	5.04	5.00	5.000	5.31	6.83
K_2	10.00	3.00	6.29	8.40	9.81	10.00	9.67	9.30
S_1	0.001	0.003	0.002	0.001	0.001	0.001	0.001	0.004
S_2	0.002	0.003	0.001	0.002	0.002	0.002	0.002	0.002
$S(\hat{\boldsymbol{\theta}})$		1.36	0.95	0.05	0.0003	0.0001	0.001	0.02
RMSE		0.34	0.28	0.06	0.00	0.00	0.02	0.04

For testing purpose, in this example, we assume that the "true" parameter values are known and the observation data are obtained by (i) using the "true" parameters as model input to solve the forward problem to find a set of model output according to the observation design and (ii) adding artificial observation error to this set of data to obtain the observed data for inversion. The observation error is assumed to be normally distributed with zero mean and a certain standard deviation.

As shown in Table 2.2, when the Gauss–Newton method is used, the inverse solution converges fast. When the observation error is less than 0.005 (truncation error), the fitting residual $S(\boldsymbol{\theta})$ decreases from 1.36 to 0.0001 after only four iterations and the true parameter values are identified exactly. Moreover, the inverse solution is unique and independent of the selection of initially guessed values in the admissible region. After adding Gaussian observation error with zero mean and standard deviation σ up to 0.5 [m], the solution becomes inaccurate but stable. If we add two more observation times at 2.0 and 5.0 days, an accurate inverse solution can be obtained with the same level of observation error.

2.5 Review Questions

1. Can we find the true physical parameters of a system by model inversion? Under what conditions can we find their satisfactory approximate values by model inversion?
2. Define the parameter space, state space, and observation space for the 1-D mass transport model given in Example 1.5. What are the operators $\mathcal{L}, \mathcal{M}$, and $\mathcal{D}$ for this model when the assumptions in Example 1.9 are satisfied?
3. Compare the direct method and the indirect method of model inversion.
4. Create an example of linear model inversion to show the underdetermined, uniquely determined, and overdetermined cases. What are the inverse solutions obtained by SVD for these three cases?
5. What are the advantages and disadvantages of using TSVD?
6. Why linearization is not always feasible for model inversion?

7. Complete the following steps: (a) use $C(x,t) = \mathcal{M}(R, D, v, x, t)$ in Eq. (1.2.2) as the forward solution operator, (b) define the observation operator $\mathcal{D}$ by designing a set of observation locations and times, (c) specify R^t and D^t as the "true" values of parameters R and D and use them as model inputs, (d) run the model to find the model calculated concentrations $\mathbf{C}_D = \mathcal{D}\mathcal{M}(R^t, D^t)$, (e) add "observation error" to $\mathbf{C}_D$ as the "observed" data $\mathbf{C}_D^{obs}$, (f) select a numerical optimization code, for example, from the Matlab function `fminunc`, (g) develop a code for calculating the objective function defined by the L_2-norm, (h) specify R^0 and D^0 as the initial guess to solve the inverse problem. Answer the following questions: Is this a CIP? Can R^t and D^t be found approximately? Can the flow velocity be estimated too? Try to use different designs of observation, add different levels of observation error, and compare the effect of different stopping criteria.
8. Draw a flow chart of using the Levenberg–Marquardt Gauss–Newton algorithm for model inversion.
9. As a course project, select a frequently used model in your field, for which you have a forward solution code. Follow the steps in Question 7 to form a CIP. Then find out if a set of hypothetical parameter values can be identified correctly by inverse solution.

Chapter 3
Multiobjective Inversion and Regularization

In Chap. 2, we cast the classical inverse problem (CIP) into an optimization problem based on criterion (C-1), which requires that the model outputs fit the observed data as much as possible. When existing data cannot support the identification of a complex model, the objective function may become nonconvex and the optimal solution may become unstable. As a result, a satisfactory inverse solution cannot be obtained by simply choosing a different optimization method or by selecting an alternative misfit function. The key to reducing this inherent difficulty of the inverse problem is to gather more information.

Criterion (C-2) of inversion (Sect. 1.4) requires the use of prior information as much as possible. The inverse problem becomes a bicriterion optimization problem after incorporating prior information. In Sect. 3.2, we will show how this problem is formulated and solved by using multiobjective optimization (MOO). The use of prior information in such a way can be seen as a special case of the regularization method to be introduced in Sect. 3.3. Regularization provides a general framework for increasing the stability of inverse solutions at the price of the possible loss of accuracy.

Only the inversion of a single-state model is considered in Chap. 2. Environmental and water resources (EWR) models are often multistate models characterized by a set of coupled equations, as shown in Chap. 1. The state variable of one equation may depend on states or parameters in other equations which, in turn, implies that measurements of one state variable may provide information for identifying the states and/or parameters in other equations. For example, concentration measurements can be used to identify the hydraulic conductivity of subsurface formations, remotely sensed surface temperature data can be used to estimate soil moisture, and hydraulic heads can be used to infer recharge rates. The inversion of a multistate model is called a *coupled inverse problem*. It allows fusion of multiple sources of data for parameter identification. We show how to formulate the coupled inverse problem into an MOO problem in Sect. 3.4.

In Chap. 2, the optimal parameter was obtained by minimizing a single objective function defined by a predetermined misfit function in the observation space. Because of errors in observation data and also in the model structure, the parameters

N-Z. Sun, A. Sun, *Model Calibration and Parameter Estimation,*
DOI 10.1007/978-1-4939-2323-6_3

identified based on minimizing only a single objective function may not be acceptable when measured by other misfit functions. For example, in the inversion of a mass transport model, the transport parameters identified by fitting a limited number of noisy concentration observations may not fit the plume arrival times or peak concentration values well. Section 3.5 considers a *multiobjective method* that uses multiple misfit functions for inverse solution. The obtained parameter values represent the best trade-offs among different objectives. This multiobjective method actually exceeds the scope of CIP because it considers the model structure error. We introduce the method in this chapter because it has the same mathematical form and uses the same solution method as that is used for the coupled inverse problem (i.e., MOO). Various algorithms for solving MOO, including recently developed Multiobjective evolutionary algorithms, are introduced in Sect. 3.6.

3.1 Multiobjective Optimization

Mathematically, an MOO problem is represented by

$$\min_{\boldsymbol{\theta}}\left\{f_1(\boldsymbol{\theta}), f_2(\boldsymbol{\theta}), \cdots, f_K(\boldsymbol{\theta})\right\},\ \boldsymbol{\theta} \in P_{ad}, \tag{3.1.1}$$

where $f_i(\boldsymbol{\theta})$ $(i = 1, 2, \cdots, K)$ are objective functions ($\mathbb{R}^m \to \mathbb{R}$) to be minimized. In general, a feasible solution that minimizes all objective functions of an MOO problem does not exist. A point $\boldsymbol{\theta}^* \in P_{ad}$ is called a *noninferior solution* or a *Pareto-optimal solution* of problem (3.1.1) if there does not exist another point $\boldsymbol{\theta} \in P_{ad}$ such that (i) $f_i(\boldsymbol{\theta}) \le f_i(\boldsymbol{\theta}^*)$ for all indices $i = 1, 2, \cdots K$ and (ii) the strict inequality $f_j(\boldsymbol{\theta}) < f_j(\boldsymbol{\theta}^*)$ holds for at least one index $j = 1, 2, \cdots K$. In other words, we say that $\boldsymbol{\theta}^*$ is not *dominated* by other solutions. The set of all noninferior solutions form the *Pareto front*. The solution of an MOO problem generally consists of two tasks:

- *Optimization task*, which aims to find a set of Pareto-optimal solutions
- *Decision-making task*, which selects the most preferred solution from the set

In the deterministic framework, the first task is completed by transforming the MOO problem into a constrained single-objective optimization (SOO) problem. There are different approaches to implement such a transformation. The *weighted sum method* (or the weighting method) solves the following problem:

$$\begin{aligned}&\min_{\boldsymbol{\theta}} S_{\mathbf{w}}(\boldsymbol{\theta}),\ \boldsymbol{\theta} \in P_{ad},\\ &\text{where } S_{\mathbf{w}}(\boldsymbol{\theta}) = \sum_{i=1}^{K} w_i f_i(\boldsymbol{\theta}_i) \text{ and} \sum_{i=1}^{K} w_i = 1\,,\end{aligned} \tag{3.1.2}$$

in which w_i $(i = 1, 2, \cdots, K)$ are weight coefficients. Different weighting coefficients will lead to different Pareto-optimal solutions. A shortcoming of the weighting method is that not all noninferior solutions can be found by changing the values of weighting coefficients when the problem is nonconvex. Instead of problem (3.1.2), the ε-*constraint method* solves the following problem:

$$\min_{\boldsymbol{\theta}} f_j(\boldsymbol{\theta}),\ \boldsymbol{\theta} \in P_{ad},$$

subject to (3.1.3)

$$f_i(\boldsymbol{\theta}) \le \varepsilon_i \text{ for all } i = 1, 2, \cdots, K, \text{ but } i \ne j.$$

In (3.1.3), ε_i is the upper bound of $f_i(\boldsymbol{\theta})$. By choosing different $j = 1, 2, \cdots, K$ and setting different upper bounds, different Pareto-optimal solutions are obtained. This constraint method requires more computational effort than the weighting method. It works for both convex and nonconvex problems, but it cannot locate promising Pareto-optimal solutions that are just outside of the boundary of the constraints.

The second task of MOO solution is completed either by an interactive process or by a noninteractive process. For the former, a decision maker determines how to pick a solution from the Pareto solution set based on his/her preference, while for the latter, a single solution is chosen based on certain criteria. For detailed discussions on this topic, readers may refer to monographs or textbooks on MOO (e.g., Tan et al. 2005; Sawaragi et al. 1985; Knowles et al. 2008; Coello Coello et al. 2007). In this chapter, we show how a noninteractive process can be used to find the best trade-off between the accuracy and stability of the solution.

The study of various algorithms for solving MOO problems is an active research area by itself. Both deterministic (branch and bound) and so-called evolutionary algorithms have been used to solve MOO. *Multiobjective evolutionary algorithms* (MOEA) have become one of the fastest developing fields since 1990s (Deb 2001; Branke 2008; Coello Coello et al. 2007). MOEA can search for several Pareto-optimal solutions simultaneously and have been used for multiobjective inversion in hydrology for more than a decade. We will give a brief introduction of these algorithms in Sect. 3.6.

3.2 The Second Criterion of Inverse Problem Formulation

3.2.1 Inversion with Prior Information

We have shown that the solution of an inverse problem can be transformed into an optimization problem based on the "fitting data" criterion (C-1) of inverse problem

$$\boldsymbol{\theta}_{qs} = \arg\min_{\boldsymbol{\theta}} \left\| \mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs} \right\|,\ \boldsymbol{\theta} \in P_{ad}, \tag{3.2.1}$$

where $\boldsymbol{\theta}_{qs}$ is the quasi-solution defined in Sect. 2.1.2. When the inverse problem is ill-posed, however, the optimization problem (3.2.1) cannot give us a unique and stable solution. In Sect. 2.2, we showed that after discretization and linearization, the linear model derived from an ill-posed problem becomes ill-conditioned. The inversion may be stabilized using methods such as truncated singular value decomposition (TSVD), but at the expense of increased fitting residual. As a result, the inverse solution may not be close to the true solution.

If $\mathbf{u}_D^{obs}$ alone cannot provide sufficient information for identifying $\boldsymbol{\theta}$, we can use other available information to help solve the inverse solution. Any information on the unknown parameter $\boldsymbol{\theta}$ obtained independently of the observation data $\mathbf{u}_D^{obs}$ is called *prior information*. In EWR modeling, some form of prior information almost always exists, such as direct measurements of $\boldsymbol{\theta}$, feasible ranges of $\boldsymbol{\theta}$ determined from physical reasoning, and guessed values of $\boldsymbol{\theta}$ that are elicited from experts or extracted from previous studies. Prior information may be more or less reliable and can be used in different ways. In Chap. 2, prior information was used to limit the admissible region, making the unknown parameters quasi-identifiable. In this section, prior information is incorporated directly into the inverse problem formulation as the second criterion:

(C-2) The prior information should be used as much as possible.

For example, if we have a prior guess, $\boldsymbol{\theta}_p$, of the unknown parameter $\boldsymbol{\theta}$, criterion (C-2) requires that the identified $\boldsymbol{\theta}$ should not be too far from it. In the ideal case when $\boldsymbol{\theta}_p$ is absolutely reliable, the distance between $\boldsymbol{\theta}_p$ and $\boldsymbol{\theta}$ in the parameter space, $\left\|\boldsymbol{\theta}-\boldsymbol{\theta}_p\right\|$, should be minimized. Combining criteria (C-1) and (C-2), we obtain a bicriterion optimization problem for determining the inverse solution.

3.2.2 Formulation of the Bicriterion Inverse Problem

The inverse problem formulated by criteria (C-1) and (C-2) is a bicriterion optimization problem

$$\begin{aligned}&\min_{\boldsymbol{\theta}}\left\{f_1(\boldsymbol{\theta}), f_2(\boldsymbol{\theta})\right\},\ \boldsymbol{\theta}\in P_{ad},\\ &\text{where } f_1(\boldsymbol{\theta})=\left\|\mathbf{u}_D(\boldsymbol{\theta})-\mathbf{u}_D^{obs}\right\| \text{ and } f_2(\boldsymbol{\theta})=\left\|\boldsymbol{\theta}-\boldsymbol{\theta}_p\right\|.\end{aligned} \tag{3.2.2}$$

Problem (3.2.2), which consists of two objectives, can be considered a special case of the general MOO problems.

The first task of solving problem (3.2.2) is to find its Pareto-optimal solutions. When the weighting method is used, a Pareto-optimal solution $\boldsymbol{\theta}_\alpha$ is obtained by solving the following SOO problem for a given weighting coefficient α:

$$\begin{aligned}&\min_{\boldsymbol{\theta}} S_\alpha(\boldsymbol{\theta}),\ \boldsymbol{\theta}\in P_{ad},\\ &\text{where } S_\alpha(\boldsymbol{\theta})=\left\|\mathbf{u}_D(\boldsymbol{\theta})-\mathbf{u}_D^{obs}\right\|+\alpha\left\|\boldsymbol{\theta}-\boldsymbol{\theta}_p\right\|.\end{aligned} \tag{3.2.3}$$

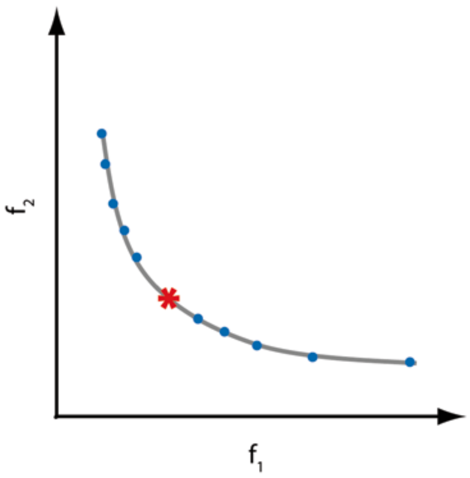


Fig. 3.1 Pareto front of a bicriterion minimization problem (*solid line*), where the dots correspond to Pareto-optimal solutions and asterisk corresponds to the best trade-off solution

The collection of all Pareto-optimal solutions often forms an L-shaped curve (i.e., Pareto front) in the objective space, as illustrated in Fig. 3.1 for a bicriterion minimization problem.

The second task of solving the bicriterion inverse problem (3.2.2) is to select a solution from the L-curve. The solution of a bicriterion optimization problem is actually a process of making tradeoffs between Pareto-optimal solutions—an increase in $f_1(\boldsymbol{\theta})$ is used to compromise for the reduction in $f_2(\boldsymbol{\theta})$ and vice versa. The final selection is usually made by a decision maker. More discussion on the L-curve and other methods for selecting trade-off solutions will be given in Sect. 3.3.3. For the current case, we use an additional criterion to make an appropriate trade-off between the stability and accuracy of the inverse solution. That is, the inverse solution is dependent on our belief in the observation data and in the prior information.

Let η be the norm of the observation error and Δ be the reliability range of the prior information defined by $\left\|\boldsymbol{\theta}^t - \boldsymbol{\theta}_p\right\| < \Delta$, where $\boldsymbol{\theta}^t$ is the true parameter. When η is known, the *discrepancy principle* states that the best we can do is to select a solution $\boldsymbol{\theta}_\alpha$ that satisfies $\left\|\mathbf{u}_D(\boldsymbol{\theta}_\alpha) - \mathbf{u}_D^{obs}\right\| = \eta$. Any attempt to decrease the fitting residual further will not help to bring the identified parameter closer to the true parameter and may even render the inverse solution unstable. Using a small weighting coefficient α to make $\left\|\mathbf{u}_D(\boldsymbol{\theta}_\alpha) - \mathbf{u}_D^{obs}\right\| < \eta$ is called *overfitting the data* as the solution is forced to recover the random observation error rather than the true system state. At the same time, we should also avoid *overbelieving the prior information*. When Δ is known, we should not use a large α to enforce $\left\|\boldsymbol{\theta}_\alpha - \boldsymbol{\theta}_p\right\| < \Delta$ because the true parameter may not be in this range. Therefore, an appropriate solution $(\boldsymbol{\theta}_\alpha)$ should neither overfit the data nor overbelieve the prior information.

In practice, however, both η and Δ are difficult to estimate, especially when model errors are involved. Therefore, the selection of an appropriate weighting coefficient for inverse solution is still a challenging problem. We will return to this topic in Sect. 3.3.3 when discussing regularization. In practice, we may follow a trial-and-error process: starting with a small α to solve problem (3.2.3), if the solution is unique and stable and the fitting residual is acceptable, we can accept the solution as an approximate inverse solution and terminate the process; otherwise, if the solution is unstable, we need to increase the value of α and solve the problem (3.2.3) again. One should keep in mind, however, that the fitting residual is increased at the same time when α is increased.

The SOO problem (3.2.3) can be solved by using an appropriate optimization method introduced in Chap. 2. For example, if L_2-norm is used as a measure of distance for both the observation space $\mathbb{F}$ and parameter space $\mathbb{P}$, the problem becomes

$$\begin{aligned}&\min_{\boldsymbol{\theta}} S_\alpha(\boldsymbol{\theta}),\ \boldsymbol{\theta} \in P_{ad}\\ &\text{where } S_\alpha(\boldsymbol{\theta}) = \left\| \mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs} \right\|_2^2 + \alpha \left\| \boldsymbol{\theta} - \boldsymbol{\theta}_p \right\|_2^2 .\end{aligned} \tag{3.2.4}$$

Because the objective function of this problem is in the form of sum of squares, it can be solved effectively by using the Gauss–Newton method or any other appropriate method described in Chap. 2.

Example 3.1 *Bicriterion inversion of the Monod kinetics model*
Let us consider the following microbial kinetics model that couples the rate of biomass growth with the rate of substrate utilization (Monod's equation):

$$\begin{cases} \dfrac{dX}{dt} = \mu X - k_e X, \\ \dfrac{dS}{dt} = -\dfrac{\mu}{Y} X \end{cases} \tag{3.2.5}$$

subject to

$$S(t)\,|_{t=0} = S_0 \ , \ X(t)\,|_{t=0} = X_0,$$

where S and X are the substrate (nutrient) and biomass concentrations, respectively; μ is the specific growth rate given by the Monod's equation (Example 1.1)

$$\mu = \frac{\mu_{\max} S}{(k_S + S)},$$

where $\mu_{\max}$ is the maximum growth rate, k_S is the Monod's coefficient (see Eq. (1.1.1)), Y is the yield coefficient, and k_e is a known endogenous decay coefficient.

Let us denote the parameter vector of the Monod's model as $\boldsymbol{\theta} = \{\mu_{\max}, k_S, Y\}$. The main goal of this example is to identify $\boldsymbol{\theta}$ using measurements of S and X. Assume that the "true" model parameters are $\mu_{\max} = 0.15\text{mg/L}$, $k_S = 0.4\text{mg/L}$,

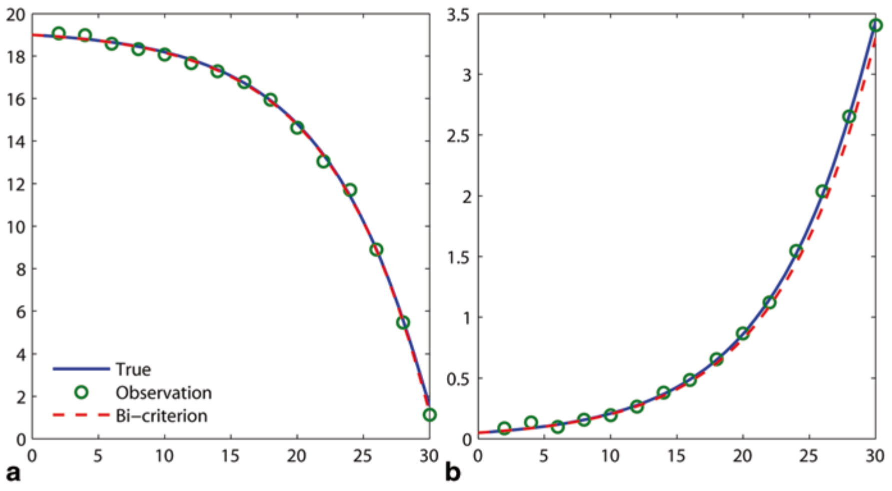


Fig. 3.2 Plots of "true" (*solid line*) and "noisy" observations (*open circles*) of **a** the substrate concentration S and **b** the biomass concentration X

$Y = 0.2\text{mg/mg}$, and $k_e = 0.004\text{L/h}$, and the initial conditions are $S_0 = 19\text{mg/L}$ and $X_0 = 0.05\text{mg/L}$. The forward solution given in (3.2.5) is solved numerically using the fourth-order Runge–Kutta algorithm (`ode45` function in MATLAB). For demonstration, the "true" profiles $S(t)$ and $X(t)$ are generated and plotted in Fig. 3.2a and b, respectively. Observations are taken every 2 h for 30 h. The "observed values" of $S(t)$ and $X(t)$ (open circles) are then perturbed by adding zero-mean Gaussian noise with standard deviations of 0.2 and 0.02 mg/L, respectively, to their "true" values. These "noisy" observations and a prior guess $\boldsymbol{\theta}_p = \{0.1, 0.5, 0.1\}$ are now used to estimate $\boldsymbol{\theta}$.

Fig. 3.3 plots the variation of two objectives or criteria, $\left\|\mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs}\right\|_2^2$ and $\left\|\boldsymbol{\theta} - \boldsymbol{\theta}_p\right\|_2^2$, for different values of the weighting factor α, which is increased from 0.01 to 0.5 in 0.02 increments from the left to right of the plot. For each α, a solution ($\times$) is obtained using the MATLAB minimization function, `fminunc`. The collection of all solutions forms an L-shaped curve, as shown in Fig. 3.3. For this example, the α value that makes the best trade-off between the two criteria is 0.31 (the circled value in Fig. 3.3), which corresponds to a solution $\boldsymbol{\theta} = \{0.15, 0.12, 0.19\}$. Although this is not the best estimation of the unknown parameters, it represents the best trade-off between the two criteria among all feasible solutions. The trade-off solution is plotted in Fig. 3.2 (dashed lines). Readers may refer to Sect. 3.3.3 for an in-depth discussion of the L-curve method.

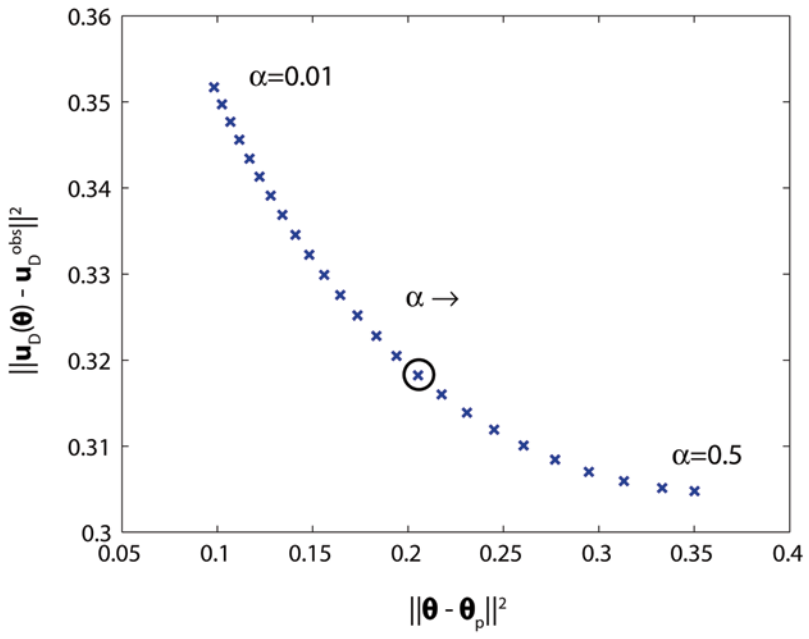


Fig. 3.3 Plot of bicriterion values for different weighting coefficient values, in which α increases from 0.01 to 0.5 from the left to the right of the plot and the corresponding solutions are labeled by ×

3.2.3 *Inversion with Constrained Optimization*

The general form of constrained optimization is given by

$$\min_{\boldsymbol{\theta}} S(\boldsymbol{\theta})$$
$$\text{subject to } \begin{cases} g_r(\boldsymbol{\theta}) \le 0,\ r = 1,\dots,n_g, \\ h_l(\boldsymbol{\theta}) = 0,\ l = 1,\dots,n_h, \end{cases} \tag{3.2.6}$$

where $g_r(\boldsymbol{\theta})$ $(r = 1,2,\dots,n_g)$ are inequality constraints and $h_l(\boldsymbol{\theta})$ $(l = 1,2,\dots,n_h)$ are equality constraints. This problem can be converted to an unconstrained optimization problem by defining a new objective

$$S_\alpha(\boldsymbol{\theta}) = S(\boldsymbol{\theta}) + \alpha H(\boldsymbol{\theta}), \tag{3.2.7}$$

where $\alpha H(\boldsymbol{\theta})$ is called a *penalty term*, α the *penalty coefficient*, and $H(\boldsymbol{\theta})$ the *penalty function* defined by

$$H(\boldsymbol{\theta}) = \sum_{r=1}^{n_g} \max\{g_r(\boldsymbol{\theta}), 0\} + \sum_{l=1}^{n_h} |h_l(\boldsymbol{\theta})|. \tag{3.2.8}$$

When a point $\boldsymbol{\theta}$ is feasible (i.e., satisfying all constraint conditions), this penalty term is equal to zero (no penalty); otherwise, the penalty term is positive and increases with the increase of α. As a result, an infeasible point cannot be a minimizer. By appropriately selecting the value of α, the solution of the constrained problem (3.2.6) can be obtained by solving an unconstrained problem that minimizes the objective function $S_\alpha(\boldsymbol{\theta})$ in (3.2.7). The complexity on the feasible region is thus transferred into the complexity in the objective function. The advantage of the penalty method is that it is easy to use and converges to a feasible solution when $\alpha \to \infty$. However, the solution may become unstable when α increases. The penalty method is also criticized for giving inexact solution (i.e., inexact fulfillment of the constraints; Han 1979). The other commonly used method for converting constrained optimization problems into unconstrained ones is the Lagrange multiplier approach, which seeks to satisfy all constraints and to solve for the optimal multiplier simultaneously (Fletcher 1987). This is the approach behind some popular tools such as MATLAB's `fmincon` function. However, both methods may return local minima.

When the constraint method is used to solve the bicriterion inverse problem, a Pareto solution can be obtained by solving either

$$\min_{\boldsymbol{\theta}} \left\| \mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs} \right\|, \text{ s. t. } \left\| \boldsymbol{\theta} - \boldsymbol{\theta}_p \right\| \le \delta \text{ and } \boldsymbol{\theta} \in P_{ad}, \tag{3.2.9}$$

or

$$\min_{\boldsymbol{\theta}} \left\| \boldsymbol{\theta} - \boldsymbol{\theta}_p \right\|, \text{ s. t. } \left\| \mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs} \right\| \le \varepsilon \text{ and } \boldsymbol{\theta} \in P_{ad} \tag{3.2.10}$$

where δ and ε are given numbers. Different values of δ in problem (3.2.9) or ε in problem (3.2.10) lead to different Pareto-optimal solutions. Note that when the penalty method is used to solve the constrained problem (3.2.9), the constraint method gives the same SOO problem as the weighting method (3.2.3). Reducing δ can improve the stability of the inverse solution, while reducing ε can improve the accuracy. However, as explained in Sect. 3.2.2, we should not let $\varepsilon < \eta$; otherwise, the overfitting problem will occur. On the other hand, in order to avoid the overbelieving problem, we should not let $\delta < \Delta$.

Prior information is often given in the form of an interval constraint that assigns the upper and lower bounds to each component of the unknown parameter vector $\boldsymbol{\theta}$: $\underline{\theta}_i \le \theta_i \le \overline{\theta}_i$ $(i = 1, 2, \cdots, m)$. In this case, the admissible region is an m-dimensional hyperbox. When the penalty method is used, we can define the following penalty function:

$$H(\boldsymbol{\theta}) = \sum_{i=1}^{m} \left\{ \max\left(\theta_i - \overline{\theta}_i, 0\right) + \max\left(\underline{\theta}_i - \theta_i, 0\right) \right\}. \tag{3.2.11}$$

Many software packages for inverse solution, however, often adopt a simple strategy to make the solution fall within a given range: when a component θ_i of $\boldsymbol{\theta}_n$

obtained during the iteration process falls out of range, set $\theta_i = \bar{\theta}_i$ if $\theta_i > \bar{\theta}_i$, or set $\theta_i = \underline{\theta}_i$ if $\theta_i < \underline{\theta}_i$.

3.3 Regularization

3.3.1 Tikhonov Regularization

Regularization is a general method for stabilizing the solution of an ill-posed problem through the trade-off between accuracy and stability. From this sense, parameterization, TSVD, and the use of prior information can be seen as some type of the regularization method. In the deterministic framework, the general form of Tikhonov regularization is given by (Tikhonov and Arsenin 1977)

$$\begin{aligned} &\min_{\theta} S_\alpha(\theta), \\ &\text{where } S_\alpha(\theta) = G(\theta) + \alpha R(\theta), \end{aligned} \tag{3.3.1}$$

θ can be a finite-dimensional vector or an infinite-dimensional function, $G(\theta)$ is the objective function, $\alpha R(\theta)$ is called the regularization term, and $0 < \alpha < \infty$ is the *regularization coefficient.* The following are examples of regularization.

3.3.1.1 Regularization and Bicriterion Inversion

The bicriterion inverse problem considered in Sect. 3.2 is a special case of regularization. When using the weighting method to stabilize the quasi-solution in (3.2.1) and letting $G(\theta) = \left\| u_D(\theta) - u_D^{obs} \right\|$ and $R(\theta) = \left\| \theta - \theta_p \right\|$, we have

$$S_\alpha(\theta) = \left\| u_D(\theta) - u_D^{obs} \right\| + \alpha \left\| \theta - \theta_p \right\|. \tag{3.3.2}$$

This is exactly the same objective function as that was defined in (3.2.3) when θ is a finite-dimensional vector. The regularization term represents prior information and the regularization coefficient plays the role of weighting coefficient.

3.3.1.2 Regularization and Constrained Optimization

A similar link exists between regularization and constrained optimization. When using the penalty method to solve the constrained optimization problem (3.2.6) and letting $G(\theta) = S(\theta)$ and $R(\theta) = H(\theta)$, we can represent the objective function in (3.2.7) through regularization as

$$S_\alpha(\theta) = G(\theta) + \alpha R(\theta), \tag{3.3.3}$$

where the regularization term represents a penalty to the objective function when the constraint is violated. Therefore, from the perspective of constrained optimization, regularization uses a penalty term to stabilize the quasi-solution.

3.3.1.3 Regularization for Well Posedness

If an inverse problem is ill-posed, regularization provides a general mechanism to make the problem conditionally well posed by adding restrictions to the identified parameter θ. For example, when an inverse problem has nonunique solutions, we may select a solution that has the minimum norm in the parameter space by minimizing

$$S_\alpha(\theta) = \left\| u_D(\theta) - u_D^{obs} \right\| + \alpha \left\| \theta \right\|. \tag{3.3.4}$$

Minimizing the objective function (3.3.4) yields a unique and stable inverse solution when the model is linear and $\alpha > 0$ (see next subsection).

3.3.1.4 Regularization for Smoothing

If we know that a distributed parameter is a smooth function, we can use this information to stabilize the inverse solution by adding a regularization term to the objective function to penalize nonsmoothness

$$S_\alpha(\theta) = \left\| u_D(\theta) - u_D^{obs} \right\|_2^2 + \alpha \left\| \partial\theta / \partial x \right\|_2^2. \tag{3.3.5}$$

The second term on the right-hand side of the above equation provides regularization on the gradient of θ. Thus, if θ is nonsmooth, the penalty term will suppress it by penalizing large changes in the gradient and make the inverse solution stable. Note: (i) a general form of the regularization term in (3.3.5) is $\left\| Dx \right\|$, where D is a differentiation operator (e.g., first- or second-order derivative) and (ii) we can combine the regularization terms in (3.3.4) and (3.3.5) to penalize large variations in both the magnitude and smoothness of θ.

3.3.2 Regularization of Linear Models

The Tikhonov regularization theory is well established for linear models (Hansen 2010; Golub and Van Loan 1996). Recall the general form of linear models given in (2.2.1):

$$\mathbf{G\theta} = \mathbf{d}, \tag{3.3.6}$$

where **G** is an $n \times m$ matrix, $\boldsymbol{\theta} \in \mathbb{R}^m$ is the unknown parameter, and $\mathbf{d} \in \mathbb{R}^n$ is the data vector. To stabilize the solution of (3.3.6), setting the *G* and *R* terms in (3.3.1) to $G(\boldsymbol{\theta}) = \|\mathbf{G\theta} - \mathbf{d}\|_2^2$ and $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2$, respectively, we get

$$S_\alpha(\boldsymbol{\theta}) = \|\mathbf{G\theta} - \mathbf{d}\|_2^2 + \alpha\|\boldsymbol{\theta}\|_2^2. \tag{3.3.7}$$

The regularization term in the above equation suggests that a solution with small length is favored. Applying the necessary condition for minimization on (3.3.7) by setting $\partial S_\alpha / \partial\boldsymbol{\theta} = \mathbf{0}$ gives

$$(\mathbf{G}^T\mathbf{G} + \alpha\mathbf{I})\boldsymbol{\theta} = \mathbf{G}^T\mathbf{d} \tag{3.3.8}$$

Substituting the singular value decomposition (SVD) of $\mathbf{G}$ (see (2.2.9)) into the above equation, we obtain

$$(\mathbf{V}\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\mathbf{V}^T + \alpha\mathbf{I})\boldsymbol{\theta} = \mathbf{V}\boldsymbol{\Sigma}^T\mathbf{U}^T\mathbf{d}. \tag{3.3.9}$$

It can be verified that the solution to this equation is (Hansen 2010)

$$\boldsymbol{\theta}_\alpha = \sum_{i=1}^{k} \frac{s_i^2}{s_i^2 + \alpha} \frac{\mathbf{u}_i^T\mathbf{d}}{s_i} \mathbf{v}_i, \tag{3.3.10}$$

where $k = \min(m, n)$; $\mathbf{u}_i$ and $\mathbf{v}_i$ are the *i*th column of $\mathbf{U}$ and $\mathbf{V}$, respectively; and s_i are singular values. The effect of regularization can be clearly seen from the above equation if we compare it to (2.2.15). Because $\alpha > 0$, all terms in the $\boldsymbol{\theta}_\alpha$ expression (3.3.10) are defined, including the terms associated with very small or even zero singular values. In other words, a stable solution $\boldsymbol{\theta}_\alpha$ can always be obtained without using the TSVD. It is also clear from (3.3.10) that a larger α can increase the stability of the solution, but at the expense of introducing biases in the solution.

3.3.3 Selection of Regularization Coefficients

The solution θ_α of a regularized optimization problem depends on the regularization coefficient α. If α is set too large, θ_α becomes inaccurate because of the large fitting residual. On the other hand, if α is set too small, θ_α becomes unstable because of the ill-posedness of the problem and the effect of observation error. Although the problem of selecting an appropriate α is well studied for linear models, it is still an open problem for complex and nonlinear models. There are several methods available depending on what extra information we have.

3.3.3.1 The L-curve Method

The *L-curve method*, which has already been discussed in the context of bicriterion optimization under Sect. 3.2, can be used to select an appropriate regularization coefficient if we only have the observation data u_D^{obs} and prior information θ_p without other information. For any $\alpha > 0$, we can find a solution θ_α for the regularized problem (3.3.2), and we have $G(\theta_\alpha) = \left\| u_D(\theta_\alpha) - u_D^{obs} \right\|_2^2$ and $R(\theta_\alpha) = \left\| \theta_\alpha - \theta_p \right\|_2^2$. Let $X_\alpha = \log G(\theta_\alpha)$ and $Y_\alpha = \log R(\theta_\alpha)$, and then for each α, (X_α, Y_α) represents a point in the *XY* plane. By systematically changing the value of α, we obtain an L-curve, as we have done in Example 3.1. The *turning point* or *knee* of the L-curve is defined as either the closest point to the origin or the point that has the maximum curvature on a log–log plot. The α value corresponding to the turning point of the L-curve gives an appropriate balance between the accuracy and stability of the solution. From the perspective of bicriterion optimization, the turning point gives an appropriate trade-off between the two objectives. For a finite-dimensional linear model, we can use an optimization routine described in Chap. 2 to find an α that maximizes the curvature, where for each α the explicit solution θ_α is obtained using (3.3.10).

For a nonlinear model, we may use the linearization techniques described in Sect. 2.3.1 to solve the nonlinear optimization problem (3.3.1) in order to obtain a point on the L-curve. In that case, the L-curve method is ineffective because it requires significant computational effort to find the turning point, and moreover, the turning point may not correspond to the most appropriate regularization coefficient (Lukas 2006).

3.3.3.2 The Discrepancy Principle Method

If we also know the norm η of observation error besides u_D^{obs} and θ_p, the *discrepancy principle* can be used to determine the regularization coefficient. As explained in Sect. 3.2.2, we should not attempt to make the fitting residual $\left\| u_D(\theta_\alpha) - u_D^{obs} \right\| < \eta$ to avoid the overfitting problem. Therefore, the regularization coefficient α should be selected such that θ_α satisfies

$$\left\| u_D(\theta_\alpha) - u_D^{obs} \right\| = \eta. \tag{3.3.11}$$

The so selected α, however, is often too small and any error in η may lead to a large change in the value of α. A modified form of the discrepancy principle accepts a little larger fitting residual by selecting α from

$$\left\| u_D(\theta_\alpha) - u_D^{obs} \right\| = \tau\eta, \tag{3.3.12}$$

where $\tau > 1$ (Resmerita and Otmar 2006; Scherzer 1993). We can obtain an appropriate regularization coefficient through a trial-and-error process by adjusting the value of α until the condition (3.3.12) is approximately satisfied. When the value of η cannot be estimated accurately, it is often replaced by its estimated upper bound.

When the L_2-norm is used and if we know both η and Δ such that $\left\|u_D(\theta^t) - u_D^{obs}\right\|_2 \le \eta$ and $\left\|\theta^t - \theta_0\right\|_2 \le \Delta$, then the regularization coefficient in (3.2.4) can be simply assigned as $\alpha = \eta^2 / \Delta^2$. In this case, we have

$$\begin{aligned} \left\|u_D(\theta_\alpha) - u_D^{obs}\right\|_2^2 &\le S(\theta_\alpha) \le S(\theta^t) \\ = \left\|u_D(\theta^t) - u_D^{obs}\right\|_2^2 &+ \frac{\eta^2}{\Delta^2}\left\|\theta^t - \theta_0\right\|_2^2 \le 2\eta^2. \end{aligned} \tag{3.3.13}$$

Thus,

$$\left\|u_D(\theta_\alpha) - u_D^{obs}\right\|_2 \le \sqrt{2}\eta. \tag{3.3.14}$$

Comparing this equation to (3.3.12), θ_α gives a satisfactory fitting residual up to a factor $1 < \tau < \sqrt{2}$.

3.3.3.3 The Robust Least Squares Method

Robust least squares (RLS), which is a special class of robust optimization methods, can be used to obtain the regularization coefficient for linear models in certain applications. RLS was developed independently by several research groups in late 1990s (El Ghaoui and Lebret 1997; Chandrasekaran et al. 1997; Ben-Tal and Nemirovski 1998). Let us start with the general linear model

$$\mathbf{G\theta} = \mathbf{d},$$

and assume that the model $\mathbf{G}$ and data $\mathbf{d}$ are subjected to unknown but bounded errors

$$\left\|\Delta\mathbf{G}\right\| \le \rho_\mathrm{G} \text{ and } \left\|\Delta\mathbf{d}\right\| \le \rho_\mathrm{d},$$

where Δ represents perturbations from nominal model $\mathbf{G}$ and data $\mathbf{d}$, which may be caused, respectively, by uncertainty in model parameters and measurement errors, and ρ_G and ρ_d are bounds of the perturbations. The actual linear regression problem that needs to be solved becomes

$$\tilde{\mathbf{G}}\mathbf{\theta} = \tilde{\mathbf{d}}$$

where the tilde symbol denotes quantities with uncertainty. The original goal of RLS is to minimize the effect of unknown error on estimates of $\mathbf{\theta}$ by utilizing information of error bounds, which is similar to the discrepancy principle method introduced in the above. With the triangle inequality, it can be shown that

$$\left\|\tilde{\mathbf{G}}\boldsymbol{\theta} - \tilde{\mathbf{d}}\right\| = \left\|(\mathbf{G} + \Delta\mathbf{G})\boldsymbol{\theta} - (\mathbf{d} + \Delta\mathbf{d})\right\| \le \left\|\mathbf{G}\boldsymbol{\theta} - \mathbf{d}\right\| + \rho_{\mathbf{G}}\left\|\mathbf{d}\right\| + \rho_{\mathbf{d}} \quad (3.3.15)$$

The right-hand side of (3.3.15) provides the upper bound of $\left\|(\mathbf{G} + \Delta\mathbf{G})\boldsymbol{\theta} - (\mathbf{d} + \Delta\mathbf{d})\right\|$, and the RLS solution is the solution of the following minimization problem (Chandrasekaran et al. 1997)

$$\min_{\boldsymbol{\theta}}\left\|\mathbf{G}\boldsymbol{\theta} - \mathbf{d}\right\| + \rho_{\mathbf{G}}\left\|\mathbf{d}\right\| + \rho_{\mathbf{d}} \quad (3.3.16)$$

subject to

$$\left\|\Delta\mathbf{G}\right\| \le \rho_{\mathbf{G}} \text{ and } \left\|\Delta\mathbf{d}\right\| \le \rho_{\mathbf{d}}.$$

Sun et al. (2006a) showed that (3.3.16) is equivalent to the so-called cone programming problem given below

$$\min \eta \quad (3.3.17)$$

subject to

$$\left\|\mathbf{G}\boldsymbol{\theta} - \mathbf{d}\right\| \le \eta - \tau \text{ and } \rho_{\mathbf{G}}\left\|\mathbf{d}\right\| \le \tau,$$

in which η and τ are slack variables introduced to facilitate problem solving. A main feature of the above cone programming problem is that it involves two norm constraints (cones). Detailed discussion on cone programming can be found in (Boyd and Vandenberghe 2004). The solution to (3.3.17) is given in the following form (Sun et al. 2006a):

$$\boldsymbol{\theta}_{RLS} = \begin{cases} (\mathbf{G}^T\mathbf{G} + \alpha\mathbf{I})^{-1}\mathbf{G}^T\mathbf{d}, & \text{when } \alpha = \rho_{\mathbf{G}}^2(\lambda - \tau)/\tau > 0 \\ \mathbf{G}^{\dagger}\mathbf{d}, & \text{otherwise} \end{cases}. \quad (3.3.18)$$

Thus, the regularization parameter α is obtained rigorously by solving an optimization problem.

Unlike the discrepancy principle method that makes use of the bound of fitting residual between model outputs and state observations, the RLS assumes bounds on the linear model. It also assumes that a nominal model **G** can be defined, which may correspond to the system under normal conditions (e.g., mean model). The magnitude of $\rho_{\mathbf{G}}$ represents the level of perturbations from the nominal model and may be defined through either uncertainty quantification (see Chap. 10), simulation, or prior knowledge. In-depth discussion on RLS and its application to contaminant source identification problems can be found in (Sun et al. 2006a, b).

Besides the above-mentioned methods, there are several other methods that can be used to determine the regularization coefficient, such as generalized cross-validation (Aster et al. 2013; Golub et al. 1979) and restricted maximum likelihood (Phillips et al. 2002). The MATLAB-based toolbox, Regularization Tools, developed by Hansen (2010), includes a number of useful utility functions for diagnosing the stability of linear systems and for selecting regularization coefficients. For example, the toolbox provides implementations of TSVD, L-curve, and generalized cross-validation methods.

All of the methods mentioned herein are only clearly justified for linear model identification or quadratic objective functions. For complex nonlinear model identification, appropriate regularization coefficients can be found approximately by the following steps:

- At the beginning, use a large value of α to assure that the regularized problem is well posed or well conditioned after linearization.
- Then, gradually decrease the value of α to make the fitting residual smaller while maintaining the stability of the solution.

Algorithms for solving regularized optimization problems will be discussed in Chap. 4.

Example 3.2 *Tikhonov regularization for deconvolution*
Convolution integrals are often used in EWR for simulating a transport process, such as the surface runoff and discharge from a spring (Example 1.2) and mass transport from a point contaminant source (Example 2.3). Its general form is given by

$$g(t) = \int_{t_1}^{t_2} \phi(t-\tau) f(\tau) d\tau, \tag{3.3.19}$$

where $f(\tau)$ is the model input, $g(t)$ is the model output, and $\phi(t-\tau)$ is called the kernel function or transfer function. The *deconvolution* problem solves for the inverse problem, namely, estimating input function $f(\tau)$ based on discrete measurements of output function, $g(t)$.

An inherent challenge of solving this inverse problem is that small perturbations in data $g(t)$ may cause arbitrarily large perturbations in the solution, an aspect that pertains essentially to all ill-posed problems. Let us consider the following function:

$$\varphi(x) = \begin{cases} 1 + \cos(\pi x / 3), & |x| < 3, \\ 0, & |x| \geq 3. \end{cases} \tag{3.3.20}$$

This is a classical problem known as the Phillips problem (Phillips 1962). Let $f(\tau) = \varphi(\tau)$, $\phi(t-\tau) = \varphi(t-\tau)$, and

$$g(t) = (6 - |t|)\left[1 + \frac{1}{2}\cos\left(\frac{\pi t}{3}\right)\right] + \frac{9}{2\pi}\sin\left(\frac{\pi t}{3}\right). \tag{3.3.21}$$

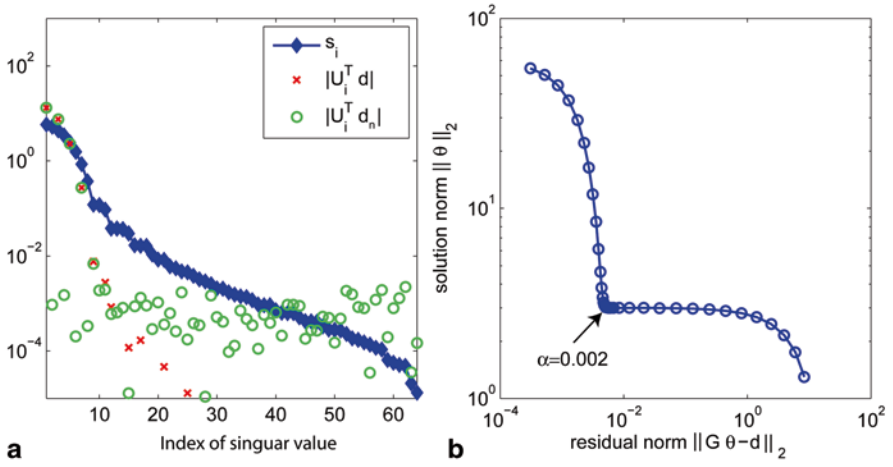


Fig. 3.4 **a** Plots of sorted singular values of $\mathbf{G}$ (*line with diamonds*) and the absolute values of $\left|\mathbf{u}_i^T\mathbf{d}\right|$ ($\times$) and $\left|\mathbf{u}_i^T\mathbf{d}_n\right|$(o) and **b** the L-curve obtained by using different regularization coefficients

The integration interval is $[-3,3]$. Substituting these definitions into (3.3.19) and after discretizing the integrals, we get a linear system of equations in the form of $\mathbf{G\theta} = \mathbf{d}$. We use the `Phillips` function in Regularization Tools (Hansen 2010) to generate the system matrix $\mathbf{G}$ for this example. The function takes a single input, n, as the number of discretization intervals. For $n = 64$, the resulting linear system is ill-conditioned, with a condition number of 4.4×10^5.

Figure 3.4a plots the singular values of $\mathbf{G}$ (line with filled diamond). Also plotted in Fig. 3.4a are the absolute values of $\mathbf{u}_i^T\mathbf{d}$, where s_i is the *i*th singular value and $\mathbf{u}_i$ is the *i*th left singular vector $\mathbf{G}$. Such a plot is referred to as the Picard plot (Hansen 2010) and can be useful for visually examining if $\left|\mathbf{u}_i^T\mathbf{d}\right|$ decays faster than the singular values of the same system matrix. The condition provides a necessary condition to check whether a good solution can be obtained without regularization. From Fig. 3.4a, we see that Picard condition is apparently satisfied when no measurement error exists and $\left|\mathbf{u}_i^T\mathbf{d}\right|$ ($\times$symbol) decays monotonically. To add the effect of noise, the measurement vector $\mathbf{d}$ is perturbed by Gaussian white noise with a variance of 1×10^{-6}, and the resulting perturbed vector is denoted by $\mathbf{d}_n$. Figure 3.4a shows that $\left|\mathbf{u}_i^T\mathbf{d}_n\right|$ ($\circ$ symbol) decreases to 10^{-3}, and then just randomly varies around that level without further decaying. So in this case, even a small amount of random noise can make the system unstable and regularization is needed to obtain a useful solution.

We now proceed to show the selection of regularization coefficient α using the L-curve method for the case with random noise. Figure 3.4b plots the L-curve for different values of the regularization coefficient, from which a turning point can be clearly observed and the corresponding α value is 2×10^{-3}. The `l_curve` function in Regularization Tools was used to generate the L-curve. ■

Tikhonov regularization (3.3.1) adds a regularization term to the original objective of inversion to stabilize the solution, as the regularization term carries additional information on the solution. Detailed discussions on Tikhonov regularization for linear and nonlinear model inversion and its applications can be found in the books of Engl et al. (1996), Doicu et al. (2010), Wang et al. (2010), and Aster et al. (2013).

Generally speaking, any method that transforms an ill-posed problem to a well-posed one can be considered as some type of regularization method. Parameterization, for example, is such an approach that uses a simplified representation of the unknown parameter for inversion. Because the number of degrees of freedom of the inverse solution is lowered, the existing data may become sufficient for identifying the simplified representation. Of course, using parameterization or any regularization methods, we can only obtain an approximation of the unknown parameter. The choice of parameterization method has a profound impact on inverse solutions. In Chap. 6, we will give a detailed account of parameterization methods in both deterministic and stochastic frameworks. In Chap. 8, we will introduce several regularization-based methods for training data-driven models.

In addition to regularizing the parameter space $\mathbb{P}$, we can also regularize the state space $\mathbb{U}$ and the observation space $\mathbb{F}$, such as increasing the quantity of observation data by adding either guessed or interpolated/extrapolated values of the state variable(s), or by improving the quality of the observation data through filtering out observation error.

3.4 Parameter Identification of Multistate Models

3.4.1 Multistate Modeling

As explained in Chap. 1, a EWR system has multiple state variables, such as pressure, temperature, velocity, concentration, and soil moisture content. A physical process that involves two or more state variables is represented by a multistate model (or a coupled model), in which state variables are coupled through a set of governing equations and auxiliary conditions. A multistate model may contain multiple physical parameters that need to be identified. Because of the nature of EWR, a large number of multistate models exist in EWR fields. The following gives an incomplete list:

- Nonisothermal, reactive mass transport models
- Groundwater flow and geomechanical models
- Surface–subsurface water interaction models
- Multiphase flow models
- Biodegradation models

- Colloid-facilitated radioactive transport models
- Land and water resources management models
- Global climate models

An often seen example in groundwater modeling is the coupled groundwater flow and mass transport model. This model consists of a confined aquifer flow equation, a mass transport equation (Example 1.7), the Darcy's law, and two-state equations for density ρ and viscosity μ, respectively,

$$\begin{aligned}
&S\frac{\partial h}{\partial t} = \nabla\cdot[Kb\nabla h] + W,\\
&h|_{t=0} = f_0,\ h|_{\Gamma_1} = f_1,\ -Kb\nabla h\cdot\mathbf{n}|_{\Gamma_2} = f_2,\\
&\frac{\partial \phi C}{\partial t} = \nabla\cdot[\phi\mathbf{D}\nabla C] - \nabla\cdot(\phi\mathbf{V}C) - k_d C,\\
&C|_{t=0} = g_0,\ C|_{\Gamma_1} = g_1,\ -(\mathbf{D}\nabla C - \mathbf{V}C)\cdot\mathbf{n}|_{\Gamma_2} = g_2,\\
&\mathbf{V} = -(K/\phi)\nabla h,\\
&\rho = \rho(C),\ \mu = \mu(\mathrm{C}).
\end{aligned} \tag{3.4.1}$$

This model involves five state variables, namely, the hydraulic head h, concentration C, velocity vector $\mathbf{V}$, fluid density ρ, and viscosity μ. This model is coupled through the Darcy's law and mass conservation: a change in h causes a change in the flow field which, in turn, a change in C. If a change in C affects ρ and μ, the value of hydraulic conductivity $K = k\rho g/\mu$ will be affected, which causes a change in h. Geochemical and geomechanical processes, which are not considered in (3.4.1), may also alter permeability and porosity. The aquifer thickness is b and W is sink/source term, and both are assumed deterministic here. Thus, the unknown parameters of the model include the storativity S, porosity ϕ, adsorption coefficient k_d, hydraulic conductivity K, and hydrodynamic dispersion coefficient tensor $\mathbf{D}$. For an isotropic aquifer, $\mathbf{D}$ is determined by the longitudinal and transverse dispersivities α_L and α_T, respectively.

In general, a multistate model or a coupled model can be represented by the following form:

$$\mathcal{L}(\mathbf{u},\mathbf{q},\mathbf{p},\mathbf{b}) = \mathbf{0}, \tag{3.4.2}$$

where $\mathcal{L}$ represents a set of equations (algebraic equations, integral equations, ODEs, or PDEs), $\mathbf{u} = (u_1, u_2, \cdots, u_k)$ is a set of state variables, $\mathbf{q}$ is a set of control variables, $\mathbf{p}$ is a set of parameters characterizing the system properties, and $\mathbf{b}$ is a set of parameters characterizing the boundary conditions of the system in both spatial and temporal domains. Variables and parameters $\mathbf{u}$, $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{b}$ in (Eq. 3.4.2) can be scalars, vectors, functions, and vector functions. The forward problem is the solution of $\mathbf{u}$ from the equation when $\mathbf{q}$, $\mathbf{p}$, and $\mathbf{b}$ are given. If $\mathbf{q}$, $\mathbf{p}$, and $\mathbf{b}$ are not known completely, they need to be identified using the observed values of $\mathbf{u}$.

3.4.2 Coupled Inverse Problems

A multistate model can be represented simply by $\mathcal{L}(\mathbf{u},\boldsymbol{\theta}) = \mathbf{0}$, where $\boldsymbol{\theta}$ represents all unknown parts of $\mathbf{q}$, $\mathbf{p}$, and $\mathbf{b}$, and its forward solution can be represented simply by $\mathbf{u} = \mathcal{M}(\boldsymbol{\theta})$. For a distributed parameter model, the components of $\mathbf{u}$ and $\boldsymbol{\theta}$ are functions of spatial and temporal variables. The inverse problem of a multistate model or a *coupled inverse problem* (Sun and Yeh 1990) attempts to identify $\boldsymbol{\theta}$ based on observations of multiple states. The advantage of solving a coupled inverse problem is that multiple data sources can be used to increase the identifiability of related parameters. For example, when both groundwater head and concentration measurements are available, we should solve the coupled groundwater flow and mass transport inverse problem to identify the hydraulic conductivity field because both datasets embed useful information about the unknown hydraulic conductivity.

Basic concepts and methods of multistate inversion are very similar to those of single-state inversion. There are, however, several fundamental differences between them that make the coupled inverse problem more complex and challenging. First, different state variables often have different dimensions, scales, and measurement accuracies. Second, there are crossover effects between state variables and parameters. Third, there are more options in the formulation of a performance criterion of inversion and also in the design of experiments for inversion.

We will use $\mathbf{u}_i^{obs}$ to denote a set of observed values of the *i*th state variable u_i $(i = 1, 2, \cdots, k)$. When parameter $\boldsymbol{\theta}$ is used in the forward solution, the distance between model output $\mathbf{u}_i(\boldsymbol{\theta})$ and its corresponding observation $\mathbf{u}_i^{obs}$ is measured by

$$G_i(\boldsymbol{\theta}) = \left\| \mathbf{u}_i(\boldsymbol{\theta}) - \mathbf{u}_i^{obs} \right\|, \ i = 1, 2, \cdots, k. \tag{3.4.3}$$

When the "fitting data" criterion (C-1) is used, a quasi-solution $\boldsymbol{\theta}_{qs}$ of the coupled inverse problem is the one that minimizes the difference between model outputs and observed values for all state variables

$$\boldsymbol{\theta}_{qs} = \arg\min_{\boldsymbol{\theta}} \left\{ G_1(\boldsymbol{\theta}), G_2(\boldsymbol{\theta}), \cdots, G_k(\boldsymbol{\theta}) \right\}, \quad \boldsymbol{\theta} \in P_{ad}. \tag{3.4.4}$$

The components $\theta_1, \theta_2, \cdots, \theta_r$ of $\boldsymbol{\theta}$ may be scalars, vectors, or functions that have different meanings and dimensions. Applying criterion (C-2) (i.e., using prior information as much as possible), we obtain the following r additional objectives to be minimized

$$R_j(\boldsymbol{\theta}) = \left\| \theta_j - \theta_{p.j} \right\|, j = 1, 2, \cdots, r \tag{3.4.5}$$

where $\theta_{p,j}$ is the prior estimate of θ_j. Therefore, the inverse solution with bicriterion (C-1) and (C-2) is given by

$$\boldsymbol{\theta}_{qs} = \arg\min_{\boldsymbol{\theta}} \left\{ G_1(\boldsymbol{\theta}), G_2(\boldsymbol{\theta}), \cdots, G_k(\boldsymbol{\theta}), R_1(\boldsymbol{\theta}), R_2(\boldsymbol{\theta}), \cdots, R_r(\boldsymbol{\theta}) \right\}, \quad \boldsymbol{\theta} \in P_{ad} \tag{3.4.6}$$

The coupled inverse problem is thus cast into an MOO problem. Using the weighting method, we can find the Pareto-optimal solutions of (3.4.6) by solving the following SOO problem:

$$\begin{aligned} &\min_{\boldsymbol{\theta}} S(\boldsymbol{\theta}), \boldsymbol{\theta} \in P_{ad}, \\ &\text{where } S(\boldsymbol{\theta}) = \sum_{i=1}^{k} \omega_i G_i(\boldsymbol{\theta}) + \sum_{j=1}^{r} \alpha_j R_j(\boldsymbol{\theta}). \end{aligned} \tag{3.4.7}$$

The weighting coefficients $(\omega_1, \omega_2, \cdots, \omega_k, \alpha_1, \alpha_2, \cdots, \alpha_r)$ in the above equation also play the role of unifying the dimensions of all terms in the summation. Applying the regularization concept, we can obtain the same objective function (Eq. 3.4.7), but now the regularization terms can be different types for stabilizing the inverse solution.

Example 3.3 *Solve example 3.1 with constraints*
The microbial kinetics model considered in Example 3.1 is actually a multistate problem involving two-state variables, the biomass concentration *X* and substrate concentration *S*. In Example 3.1, the prior information was given in the form of a guessed solution, $\boldsymbol{\theta}_p = \{0.1, 0.5, 0.1\}$. Oftentimes, one can at most define the range of parameter values. Suppose that the lower and upper bounds for the model parameters are available based on prior knowledge

$$\begin{aligned} &0.1 \le \mu_{\max} \le 1.0, \\ &0.1 \le k_S \le 1.0, \\ &0.1 \le Y \le 0.5. \end{aligned} \tag{3.4.8}$$

Equation (3.4.8) can be cast in the following equivalent forms:

$$\underline{\boldsymbol{\theta}} = [0.1, 0.1, 0.1] \text{ and } \overline{\boldsymbol{\theta}} = [1.0, 1.0, 0.5]$$

In this case, a constrained optimization problem needs to be solved

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \left\| \mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs} \right\|_2^2 \\ &\text{subject to} \\ &\left\| \boldsymbol{\theta} - \boldsymbol{\theta}_p \right\|_2^2 \le \delta, \\ &\underline{\boldsymbol{\theta}} \le \boldsymbol{\theta} \le \overline{\boldsymbol{\theta}}. \end{aligned} \tag{3.4.9}$$

All other data are the same as before. The constrained optimization is solved using the MATLAB function `fmincon`.

In the hypothetical example, because the "true" $\boldsymbol{\theta}^t = [0.15, 0.4, 0.2]$ is known, the norm $\left\|\boldsymbol{\theta}^t - \boldsymbol{\theta}_p\right\|_2^2 = 0.023$ can be used as Δ. Let us consider two cases: the overbelieving case, where δ is set to 0.01 ($< \Delta$), and the overfitting case, where δ is set to 0.2 ($> \Delta$). The resulting inverse solutions are $\boldsymbol{\theta}_1 = [0.16, 0.35, 0.24]$ for the first case and $\boldsymbol{\theta}_2 = [0.15, 0.16, 0.19]$ for the second case. Both solutions are different from $\boldsymbol{\theta}^t$, although $\boldsymbol{\theta}_1$ is closer. In practice, the upper bound Δ is inaccessible and a trial-and-error method has to be used.

3.4.3 *Solution of Coupled Inverse Problems*

The SOO problem (Eq. 3.4.7) can be solved by a local optimization algorithm introduced in Chap. 2 when the problem becomes well posed after regularization. Furthermore, when L_2-norm is used for all terms, the problem can be solved effectively by the Gauss–Newton method. The problem is how to determine the weighting coefficients (or how to select a Pareto-optimal solution) for the MOO problem. Because there is no decision maker involved for inverse solution, we have to use additional criteria to determine an appropriate solution. There are several "no-preference" methods available for general MOO problems (Branke 2008). For inverse solution, if we know the upper bound of the norm of observation error, η_i, for state $u_i (i = 1, 2, \cdots, k)$ and the reliability range, Δ_j, for each parameter $\theta_j (j = 1, 2, \cdots, r)$, we suggest using the *neutral compromise solution* by solving

$$\begin{aligned}&\min_{\boldsymbol{\theta}} S_\alpha(\boldsymbol{\theta}), \ \boldsymbol{\theta} \in P_{ad},\\ &\text{where } S_\alpha(\boldsymbol{\theta}) = \sum_{i=1}^{k}\left[\frac{G_i(\boldsymbol{\theta}) - \eta_i/2}{\eta_i}\right] + \alpha\sum_{j=1}^{r}\left[\frac{R_j(\boldsymbol{\theta}) - \Delta_j/2}{\Delta_j}\right],\end{aligned} \tag{3.4.10}$$

where $G_i(\boldsymbol{\theta})$ and $R_j(\boldsymbol{\theta})$ are defined in (3.4.3) and (3.4.5), respectively, and all terms in the objective function are normalized. The idea of this method is using the midpoints of the ranges of all objective functions to determine a Pareto-optimal solution. As described in Sect. 3.2.2, the regularization coefficient α can be determined by gradually increasing it from a small value until the inverse solution becomes stable. Of all Pareto-optimal solutions, the solution determined in this way is often not the closest one to the true parameter. For nonlinear model inversion, determination of the optimal weighting and regularization coefficients is still an open problem.

Example 3.4 *A coupled inverse problem of groundwater flow and mass transport*

Table 3.1 Results of inverse solution

	True	Initial	It. 3	It. 6	It. 9	$\sigma = 0.1$	$\sigma = 0.5$
K_1	5.00	3.00	5.32	5.00	5.00	5.33	5.28
K_2	10.00	3.00	8.16	10.00	10.00	6.06	8.08
S_1	0.001	0.003	0.001	0.001	0.001	0.001	0.001
S_2	0.002	0.003	0.002	0.002	0.002	0.003	0.002
ϕ_1	0.10	0.20	0.09	0.09	0.09	0.08	0.09
$\alpha_{L,1}$	40.0	20.0	44.1	40.7	40.3	80.0	44.4
$\alpha_{T,1}$	6.0	10.0	5.8	5.8	5.9	5.0	5.0
$k_{d,1}$	0.05	0.10	0.05	0.06	0.05	0.04	0.06
ϕ_2	0.20	0.10	0.16	0.12	0.12	0.05	0.30
$\alpha_{L,2}$	60.0	20.0	80.0	64.0	59.8	10.0	80.0
$\alpha_{T,2}$	10.0	12.0	20.0	13.3	9.9	5.0	5.0
$k_{d,2}$	0.10	0.15	0.30	0.19	0.19	0.05	0.30
$S(\hat{\boldsymbol{\theta}})$		55,316	46.2	0.058	0.001	6201	78.5
RMSE		26.3	0.76	0.03	0.003	8.80	1.00

In the study of groundwater contamination, the density ρ and viscosity μ can be considered the same as that of water because of the low concentration. As a result, (3.4.1) reduces to a two-state model with state variables *h* and *C* and the unknown parameters $\{K, S, \phi, \alpha_L, \alpha_T, k_d\}$ (note S denotes storativity here).

Let us use the problem settings in Example 2.10, but add the following assumptions: the inflow water from boundary section *AB* is clean, and the aquifer is free of contamination initially; the concentration of injected water to the aquifer through the injection well is 1000 mg/m^3. Both head and concentration are measured at the four observation wells timed at 0.1, 0.5, 1.0, 2.0, 5.0, 10.0, 20.0, 30.0, 40.0, and 50.0 day, respectively. The total number of observation data is 80, and the total number of unknown parameters is 12 (i.e., $\{K_i, S_i, \phi_i, \alpha_{L,i}, \alpha_{T,i}, k_{d,i} \mid i = 1, 2\}$ for the two zones). First, we attempt to identify all of these parameters simultaneously by solving the coupled inverse problem. In this case, the matrix $\mathbf{J}_D^T \mathbf{J}_D$ is singular because the contaminant plume has not moved to Zone 2; As a result, all observations are not sensitive to the four mass transport parameters, $\left\{\phi_2, \alpha_{L,2}, \alpha_{T,2}, k_{d,2}\right\}$, associated with Zone 2.

To make all 12 unknown parameters identifiable, we need to extend the observation time until the contaminant plume can be measured at least in one observation well of Zone 2. After adding two extra observation times at 200 day and 300 day, respectively, we can obtain the inverse solution. The results are shown in Table 3.1,

Table 3.2 Inverse solution after removing insensitive parameters

	True	Initial	It. 1	It. 2	It. 3	$\sigma = 0.1$	$\sigma = 0.5$
K_1	5.00	3.00	4.27	4.93	5.00	5.00	4.98
K_2	10.00	3.00	10.04	10.00	10.00	9.96	9.83
S_1	0.001	0.003	0.002	0.001	0.001	0.001	0.001
S_2	0.002	0.003	0.002	0.002	0.002	0.003	0.002
$\alpha_{L,1}$	40.0	20.0	35.3	40.3	40.1	40.8	41.8
$\alpha_{T,1}$	6.0	10.0	8.0	5.9	6.0	5.7	5.3
$\alpha_{L,2}$	60.0	20.0	73.3	62.0	59.8	61.6	64.6
$\alpha_{T,2}$	10.0	12.0	17.0	9.7	9.9	13.3	20.0
$S(\hat{\boldsymbol{\theta}})$		1620	78.08	0.059	0.001	0.35	8.81
RMSE		4.50	0.99	0.03	0.003	0.06	0.33

in which the "true" parameter values are listed in column 1, the initial guesses are given in column 2, the next three columns show solutions obtained after 3, 6, and 9 iterations, and the last two columns show solutions obtained under different levels of measurement errors.

From Table 3.1, we can see that (i) when the observation error is only caused by truncation error (0.005), the RMSE of fitting (0.003) becomes less than the observation error after nine iteration steps; (ii) ten of the twelve identified parameters converge to their true values except for ϕ_2 and $k_{d,2}$, and the result cannot be improved unless more data are collected; (iii) the inverse solution becomes unstable when standard deviation of observation error is increased to $\sigma = 0.1$, in which case the Gauss–Newton solution process terminates after three iteration steps because $\mathbf{J}_D^T\mathbf{J}_D$ becomes singular; and (iv) the RMSE of fitting is not always increased with the increase of observation error. The final column shows that the root mean square error (RMSE) of the $\sigma = 0.5$ case is less than that of the $\sigma = 0.1$ case, but the accuracy of the inverse solution becomes worse.

When the quantity and quality of data cannot support a stable inverse solution, we should give up the identification of those parameters that are insensitive to the observation data. In this example, if all ϕ and k_d are measured in the laboratory instead through inversion, the inverse solution will be stable as shown in Table 3.2. ■

Coupled flow and mass transport model inversion is a typical CIP in EWR modeling. In the field of groundwater modeling, examples of previous work are (Sun and Yeh 1990; Wagner and Gorelick 1987; Medina and Carrera 1996; Carrera 1988; Medina and Carrera 2003; Mishra and Parker 1989; Hendricks Franssen and Kinzelbach 2008).

3.5 Parameter Identification with Multiobjectives

In the previous subsection, MOO is used to solve coupled inverse problems that can extract information from multiple data sources to reduce uncertainty of the inverse solution. In this section, MOO will be used again to reduce uncertainty of the inverse solution, but through minimizing multiple measures of model performance. There are many examples of such uses in the EWR. For example, we can use both observed concentration values and solute arrival times to measure performance of a mass transport model. In hydrology, Madsen (2000) used four performance measures (or objectives of optimization) to identify the unknown parameters of a rainfall–runoff model. They are: (i) overall water balance, (ii) overall shape of the hydrograph, (iii) peak flows, and (iv) low flows. More examples of performance measures can be found in the review paper of Efstratiadis and Koutsoyiannis (2010).

We have seen in Chap. 2 that different norms or misfit functions lead to different objective functions for inverse solution. Theoretically, if the model and observation errors are insignificant and the inverse problem is well posed, the identified parameters will be close to the true parameters and the model can produce the same excitation–response relationship as the true system does. In this case, the parameters identified by minimizing one performance measure should also pass the test of other performance measures. For example, if a mass transport model calibrated by using concentration observations is a "correct" simulator, then it should be able to produce the correct arrival times as well. Similarly, for the rainfall–runoff model mentioned in the last paragraph, if the model can correctly produce the observed hydrograph, it should also be able to reproduce the observed peaks and low flows.

In practice, however, a model (or a set of model parameters) obtained by minimizing one performance measure may be different from that obtained by minimizing other measures, due to observation error, model error, and data insufficiency (e.g., using limited data to calibrate a complex model structure). As a result, we are unable to find a model that can correctly represent all attributes of the modeled system. The best we can do is to pick a Pareto-optimal solution that strikes a balance among all performance measures.

Let $\left\{S_1(\boldsymbol{\theta}), S_2(\boldsymbol{\theta}), \cdots, S_k(\boldsymbol{\theta})\right\}$ be a set of performance measures (or a set of objectives to be minimized) for identifying the parameter $\boldsymbol{\theta}$. The multiperformance measure (or the multiobjective) inverse problem aims at finding

$$\boldsymbol{\theta}_{qs} = \arg\min_{\boldsymbol{\theta}} \left\{S_1(\boldsymbol{\theta}), S_2(\boldsymbol{\theta}), \cdots, S_k(\boldsymbol{\theta})\right\},\ \boldsymbol{\theta} \in P_{ad}. \tag{3.5.1}$$

This problem is identical to the coupled inverse problem (Eq. 3.4.4) in form. Because of the complexity of these objective functions, the solution of the problem depends on the efficacy of MOEA algorithms for finding the global minimum, which we will discuss in the next section.

The multiobjective inverse problem introduced here is closely related to the topics on model structure error, model selection, and model uncertainty assessment that we will discuss in details in the successive chapters of the book.

Example 3.5 *Performance measures for calibrating a rainfall–Runoff model using MOO*
The Xinanjiang rainfall–runoff model is a semi-distributed watershed model for humid and semi-humid regions. It was originally developed to forecast inflows to the Xinanjiang Reservoir in China, but has been applied to many other basins since then (Zhao et al. 1980; Zhao 1992; Cheng et al. 2002; Ju et al. 2009). The original Xinanjiang model has 15 parameters, of which seven are used to describe the runoff generation (or water balance) component and the rest are used to describe the runoff routing component. Zhao (1992) later modified the model to add an additional interflow component, and the model now includes 17 parameters. When applying the model, the basin is first divided into a set of subbasins, and the outflow hydrograph from each subbasin is then simulated and routed down the channels to the main basin outlet. Because of its unique model structure, calibration of the Xinanjiang model has become a classical MOO problem over the years. Detailed descriptions of the Xinanjiang model structure and its parameters are provided in Zhao (1992).

Objective functions that listed below are commonly employed to calibrate the rainfall–runoff models (Madsen 2000). These objectives provide different performance measures on how well the simulated streamflow hydrograph fits the phase and magnitudes of the observed hydrograph:

- Overall water quantity balance error, which measures the difference between measured and computed streamflow for all calibration periods

$$\frac{1}{N}\left|\sum_{i=1}^{N}(Q_{obs,i} - Q_i(\boldsymbol{\theta}))\right|, \tag{3.5.2}$$

  where N is the number of calibration periods, $\boldsymbol{\theta}$ represents all model parameters, and Q_i and $Q_{obs,i}$ are simulated and observed streamflows at a basin outlet.

- Overall RMSE,

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(Q_{obs,i} - Q_i(\boldsymbol{\theta}))^2} \tag{3.5.3}$$

- Average RMSE of peak-flow events

$$\frac{1}{M_p}\sum_{j=1}^{M_p}\left[\frac{1}{n_j}\sum_{i}^{n_j}\left(Q_{obs,i} - Q_i(\boldsymbol{\theta})\right)^2\right]^{1/2}, \tag{3.5.4}$$

  where M_p is the number of peak-flow events for the simulation period and n_j the number of time steps used to approximate each peak-flow event, which varies for different events.

- Average RMSE of low-flow events

$$\frac{1}{M_l}\sum_{j=1}^{M_l}\left[\frac{1}{n_j}\sum_{i}^{n_j}\left(Q_{obs,i}-Q_i(\boldsymbol{\theta})\right)^2\right]^{1/2}, \tag{3.5.5}$$

where M_l is the number of low-flow events.

- Nash Sutcliffe coefficient (NSC), which measures the predictive power of the model

$$NSC = 1 - \frac{\sum_{i=1}^{N}(Q_{obs,i}-Q_i(\boldsymbol{\theta}))^2}{\sum_{i=1}^{N}(Q_{obs,i}-\bar{Q}_{obs})^2}, \tag{3.5.6}$$

where $\bar{Q}_{obs}$ is the mean of observations. The value of NSC ranges from $-\infty$ to 1. A value of 1 means a perfect match between model prediction and observations. At value 0, the NSC says that the model prediction is only as good as the mean observation.

The Xinanjiang model has been calibrated against many of the above performance measures using a number of different MOO calibration techniques (Cheng et al. 2002; Ju et al. 2009). For example, Cheng et al. (2002) used genetic algorithms (GA) to calibrate the Xinanjiang model. We will introduce GA in Sect. 3.6.2.

Example 3.6 *Calibration of a groundwater model using MOO*
Remotely sensed data are useful for tracking the movement and storage of water in different stores of the hydrologic cycle. The gravity recovery and climate experiment (GRACE) satellite, for example, provides monthly averaged measurements of Earth's static and time-variable gravity fields, which, after removing the effects of atmospheric and ocean tidal effects, can be used to track the total water storage change (i.e., the sum of all water stored in a vertical column, including snow, surface water, soil moisture, and groundwater; Tapley et al. 2004). The total water storage change (TWSC) data collected from GRACE can be disaggregated to provide estimates on the groundwater storage change. Traditionally, most regional groundwater models are calibrated using in situ water level data. The calibration results are poor when the coverage of groundwater monitoring network is sparse. Therefore, the GRACE-derived groundwater storage change data provide an additional source of information for calibrating regional-scale groundwater models. Sun et al. (2012) formulated an MOO problem to calibrate a regional groundwater model for a regional aquifer in the USA by considering the following two objectives:

- RMSE of modeled and observed water levels at observation locations

$$\left[\frac{\sum_{i=1}^{M} w_i \sum_{j=1}^{n_i} \left(h_{i,j} - h_{i,j}^{obs}\right)^2}{\sum_{i=1}^{M} w_i}\right]^{1/2}, \tag{3.5.7}$$

where $h_{i,j}$ and $h_{i,j}^{obs}$ are simulated and observed heads at the *i*th well and *j*th observation time, M is the total number of wells, n_i is number of observations at each well, and w_i is the weight assigned to each well.

- NSC of simulated and observed groundwater storage changes, which is similar to (3.5.6), but replaces flow rates with groundwater storage changes.

Additional performance measures may include observed and simulated boundary fluxes such as the spring discharge. At the present time, the TWSC derived from GRACE satellite mission contains significant error because of instrumentation and data processing errors (Swenson and Wahr 2006). Disaggregation of groundwater storage changes from TWSC can introduce additional errors because of uncertainties related to estimating other hydrological components (e.g., soil moisture storage). To deal with uncertainty, one approach is to incorporate prior information as additional constraints (e.g., upper and lower bounds) when formulating the MOO problem. Sun et al. (2010) formulated a robust least squares (see Sect. 3.3) problem to deal with uncertainty in GRACE data, in which a priori error bounds on TWSC and soil moisture changes were used as constraints when estimating the unknown specific yields of a regional aquifer.

3.6 Algorithms for Multiobjective Optimization

3.6.1 Deterministic Methods

As discussed in previous sections, the classical optimization methods convert MOO problems into SOO problems by focusing on one Pareto-optimal solution at a time. This is the basic mechanism behind many deterministic algorithms. In principle, many of the algorithms described in Chap. 2 can be applied. However, because the convexity of the objective function is not guaranteed even if the inverse solution is unique in the admissible region, the iteration process of a local optimization method may converge to a useless local minimum, instead of the global minimum (Fig. 3.5). As mentioned in Sect. 2.4.2, at a local minimum, the fitting residual cannot be decreased even there is no model error. Finding the global minimum of a function is much more challenging than finding a local minimum. This section gives only a brief review on deterministic global optimization algorithms. We will see that when the dimension m of the unknown parameter is high, using a deter-

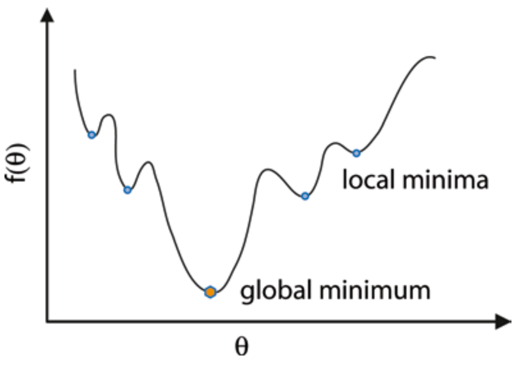


Fig. 3.5 Local optimization algorithms may return local minima (*circles*), instead of a global one (*hexagon*)

ministic global minimization algorithm for inverse solution may become infeasible in practice because of the huge computational effort involved.

The most straightforward method for finding the global minimum is *grid search* or *greedy search*, in which the objective function is evaluated at all nodes of a grid (in parameter space) to find the smallest. A major difficulty associated with this approach is the determination of the block size. Using a larger block size may miss the global minimum when the objective function has a sharp peak within a block; on the other hand, the computational effort may become unaffordable when a small block size is used. An appropriate block size can be determined only when additional information on the mathematical structure of the objective function is available, for example, when the Lipschitz condition

$$\left|S(\boldsymbol{\theta}_1) - S(\boldsymbol{\theta}_2)\right| \leq L\left\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\right\| \tag{3.6.1}$$

is satisfied everywhere and the Lipschitz constant L can be estimated. However, it is often difficult to verify such structures for the inversion of a numerical model.

The *multistart method*, one of the first used global search procedures, is often used for inverse solution when the convexity of an objective function is unknown. It starts a local optimization solver from multiple starting points and stores all solutions found during the search process. The start points can be defined by the user if a priori information is available, or they can be uniformly distributed within predefined bounds. The global minimum is the smallest one of all solutions. The multistart method is often used to test the uniqueness of an inverse solution in real case studies by checking whether all search sequences starting from different initial guesses converge to the same solution. Strictly speaking, however, there is no guarantee that the global minimum could be found by such a heuristic method, unless starting from everywhere.

The *branching and bounding* (BnB) method is a widely used deterministic global optimization algorithm. Assuming that we have a list of subregions of the admissible region: $\mathbb{C} = \left\{\mathbb{P}_1, \mathbb{P}_2, \cdots, \mathbb{P}_K\right\}$, where each subregion may potentially contain the global minimum, and we also have a global minimum $\boldsymbol{\theta}_{cgm}$ found so far. The BnB method implements the following steps:

1. *Selecting*. Select subregions from the list $\mathbb{C}$ for further exploration one by one. The selection may be based on different orders: first select a subregion that contains the currently found global minimizer $\boldsymbol{\theta}_{cgm}$, or a subregion that has the largest size. In the beginning, if we do not have additional information, $\mathbb{C}$ may consist of the entire admissible region, and $\boldsymbol{\theta}_{cgm}$ can be the initial guess of the unknown parameter.
2. *Branching*. Split a selected subregion $\mathbb{P}_k$ into several smaller subregions and delete $\mathbb{P}_k$ from the list $\mathbb{C}$.
3. *Bounding*. For each subregion $\mathbb{P}_{k,i}$ of $\mathbb{P}_k$, estimate the lower bound $\underline{S}(\mathbb{P}_{k,i})$ of the objective function on it as tight as possible. If $\underline{S}(\mathbb{P}_{k,i}) < S(\boldsymbol{\theta}_{cgm})$ and we can find a $\boldsymbol{\theta}_* \in \mathbb{P}_{k,i}$ such that $S(\boldsymbol{\theta}_*) < S(\boldsymbol{\theta}_{cgm})$, then replace $\boldsymbol{\theta}_{cgm}$ by $\boldsymbol{\theta}_*$, $S(\boldsymbol{\theta}_{cgm})$ by $S(\boldsymbol{\theta}_*)$, and add $\mathbb{P}_{k,i}$ to the list $\mathbb{C}$.
4. *Eliminating*. If $\underline{S}(\mathbb{P}_{k,i}) > S(\boldsymbol{\theta}_{cgm})$, then $\mathbb{P}_{k,i}$ can be eliminated.

After the above steps are completed for all subregions, both $\mathbb{C}$ and $S(\boldsymbol{\theta}_{cgm})$ are updated, and the next iteration can start. During this recursive process, $S(\boldsymbol{\theta}_{cgm})$ keeps deceasing, and both the number and size of subregions left in $\mathbb{C}$ become smaller and smaller. The recursion stops when the lower- and upper bounds of the current list are sufficiently close. Detailed discussion on the BnB method and its parallel implementation can be found in Pintér (1996).

The strength of the BnB method is that an entire subregion may be discarded after bounding. But the bounding approach depends on the mathematical structure of the objective function. Different BnB methods must be designed for different problems. For inverse solution, when the objective function is given by an input–output subroutine, there is no accurate and effective method to estimate its lower bound over a subregion.

The *dividing rectangles* algorithm (DIRECT) is a deterministic sampling method introduced by Jones et al. (1993). Assume that the admissible region of the unknown parameter is an m-dimensional rectangle determined by the upper and lower bounds of its components. The initial rectangle is divided into smaller rectangles, the centers of all rectangles are taken as sampling points where the objective function is evaluated, and then the potentially optimal rectangles are selected for refinement. During such an iteration process, the sampling points will form clusters at the locations of local minima and the global minimum can thus be determined. Detailed instructions on how to divide a rectangle and how to select a rectangle for refinement can be found in Finkel and Kelley (2006) and Huyer and Neumaier (1999). Sun et al. (2006b) applied a DIRECT algorithm to identify both contaminant source locations and source release histories by minimizing the difference between observed and calculated concentrations. The algorithm consists of two nested loops: the outer loop sets a trial location, while the inner loop solves a release history identification problem (see Example 2.3); the DIRECT algorithm was used to narrow the trial location regions.

When inverting a EWR model and when the quantity and quality of data are insufficient, the objective function $S(\boldsymbol{\theta})$ may have a flat bottom as shown in Fig. 3.6. In this case, any global optimization method can give an answer.

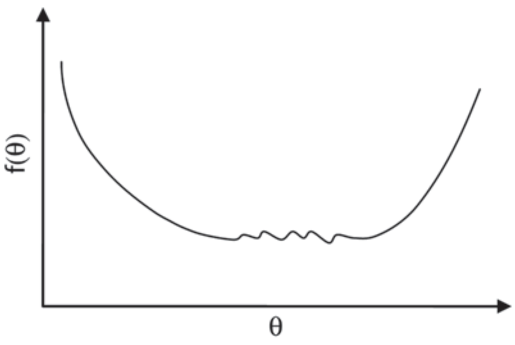


Fig. 3.6 Objective function may have a flat bottom when the information is not sufficient

3.6.2 Genetic Algorithm

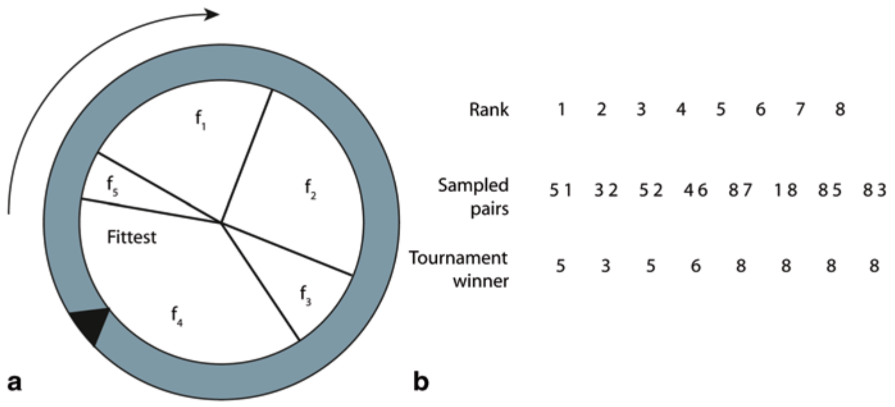
The GA is a class of global optimization algorithms for solving SOO problems. Its search heuristics is inspired by the natural evolution and selection process (Holland 1975). To solve an SOO problem, GA first encodes the decision variables into fixed-length binary strings consisting of 0s and 1s, although other forms of encoding are possible depending on the nature of the problem. These strings, which represent the candidate solutions to the optimization problem, are referred to as chromosomes in analogy to genetic biology. The chromosomes are then evolved by allowing mutations and crossovers between them in hope to generate better solutions.

A basic GA typically consists of the following steps:

1. *Initialization*. Generate the initial population of candidate solutions $S(0)$ by randomly sampling across the search space. If prior information is available, the initial solutions may be seeded in areas where optimal solutions are most likely to be found.
2. *Evaluation and selection*. For any generation $S(n)$, the fitness values of the candidate solutions are evaluated and ranked according to a user-defined objective or fitness function. Those with higher ranks are preferred and are selected to breed a new generation.
3. *Reproduction*. To generate a new generation $S(n+1)$ from its parental generation $S(n)$, two genetic operators are commonly used: *crossover* and *mutation*. The former combines parts of two or more parental chromosomes to create new and possibly better offspring, whereas the latter creates new individuals by modifying the genetic composition of a single individual, which is equivalent to a random perturbation in the vicinity of a candidate solution.

The above steps are repeated until a termination criterion is met. Usually, this happens when there is no improvement in objective function over several consecutive generations or when the maximum number of generations is exceeded.

Various GAs differ in the implementation strategies for steps 2 and 3 in the above. Commonly used selection strategies include: (i) roulette wheel selection, where good solutions are assigned larger slot sizes than the less fit solutions such



Rank	1	2	3	4	5	6	7	8
Sampled pairs	51	32	52	46	87	18	85	83
Tournament winner	5	3	5	6	8	8	8	8

Fig. 3.7 **a** Roulette wheel selection, where there are five candidates and f_i are fitness values; the probability of getting selected is proportional to $f_i / \sum_{i=1}^{5} f_i$. **b** Tournament selection, where the eight candidates are ranked and selected randomly in pairs for tournament. The winners are selected according to their ranks (e.g., 5 is selected from the pair 5–1)

that the good solutions have a better chance to be selected when the roulette wheel spins and (ii) tournament selection, where groups of candidate solutions are randomly selected and competed with each other; the fittest individual in the group wins the tournament.

Fig. 3.7 illustrates how the two different selection strategies work. Roulette wheel selection based on fitness value is known to have scaling problems (i.e., an individual with excellent fitness value is likely to dominate the intermediate population, leading to a decrease in diversity of the offspring). As a fix, rank-based selection is often used instead. In comparison, the tournament selection is popular because it is scaling invariant and it is implicitly elitist (i.e., the fittest individuals in a population are guaranteed to be selected for the next generation). Elitism is considered an important attribute of evolutionary algorithms because it ensures nondegrading performance of an algorithm during evolution.

Crossover and mutation strategies are critical to the efficiency of the GAs. The simplest crossover strategy is substring crossover, in which subsequences of a pair of randomly selected individuals are swapped to form two offspring with a probability, P_c. That is, a uniform random number r is first generated; if $r \le P_c$, the swapping occurs; otherwise, the two offspring are simply clones of their parents. The crossover of chromosomes greatly facilitates GA to approach and eventually find the optimum. However, they do not introduce new information to the population. Mutation is used to maintain diversity in the population by enabling random change in portions of the gene sequence.

The performance of GA depends on population size, the number of generations, and parameters of the selected genetic operators. Successful tuning of these parameters usually requires a considerable amount of insight into the nature of the problem at hand, which is unlikely for complicated problems. Major criticisms of the GA include its (1) computational efficiency, which is a major bottleneck for complex model, and (2) convergence, meaning there is no guarantee for the algorithm to find the global optimum for the termination criterion selected and also the algorithm can return local optima if there is not enough diversity in the population. Numerous variants of the classic GA have been developed to deal with these issues, including parallel GA and hybrids between GA and gradient-based local search algorithms. Detailed descriptions of the GA and its implementation can be found in many textbooks (Goldberg 1989; Mitchell 1996; Sivanandam and Deepa 2007). GA is available in several commercial software and open-source libraries, such as ga function in MATLAB's global optimization toolbox and GAlib (http://lancet.mit.edu/ga).

Example 3.7 *Use GA to estimate parameters of a lumped parameter hydrologic model (HyMOD)*
HyMOD is a lumped parameter rainfall–runoff model that is often used for benchmarking parameter estimation algorithms in the hydrology literature (Vrugt et al. 2008; Boyle et al. 2003; Moradkhani et al. 2005; Wagener et al. 2001; Young 2013). HyMOD conceptualizes a watershed as a simple nonlinear soil moisture reservoir connected with two series of linear reservoirs for routing excess rainfalls: three identical quick-flow tanks (x_1–x_3) for representing short-term surface detention and one slow-flow tank (x_4) for representing groundwater storage (see Fig. 3.10).

For soil moisture storage, HyMOD adopts the probability distribution model (PDM; Moore 2007, 1985), which models the storage capacity C at any point of the watershed as a random variable. The most widely used CDF for C is in the following form:

$$F(C) = 1 - \left(1 - \frac{C}{C_{\max}}\right)^b, \quad 0 \le C \le C_{\max} \tag{3.6.2}$$

where $C_{\max}$ is the maximum soil moisture storage capacity within the watershed and b controls the degree of spatial variability of the soil moisture capacity within the watershed. Rainfall excess (RE) is obtained after subtracting evaporation (ET)

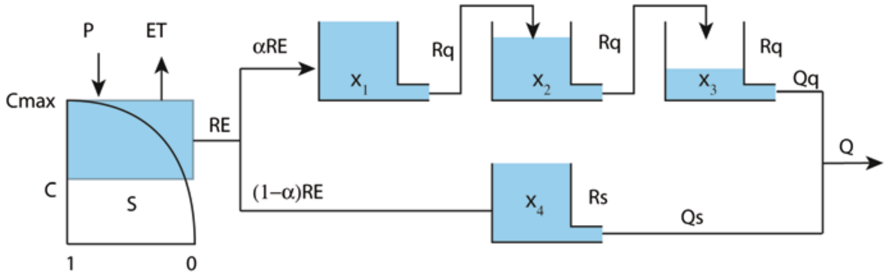


Fig. 3.8 Conceptual diagram of HyMOD, adapted from Moradkhani et al. (2005). Symbols are explained in the text

Table 3.3 Bounds of HyMOD parameters and calibration results

Parameter [Unit]		Minimum	Maximum	Tournament	Roulette
$C_{\max}$	mm	150	350	302.62	259.75
b	–	0.1	1.5	0.59	0.497
α	–	0.6	1.0	0.989	0.987
R_q	day	0.2	0.7	0.485	0.483
R_s	day	0.01	0.1	0.0119	0.0211

and soil moisture storage from precipitation (P). The partition of rainfall excess between the two series of tanks is controlled by a partitioning factor α. The flow routing in quick-flow and slow-flow linear reservoirs is determined by the residence times in tanks R_q and R_s, respectively. Finally, the simulated streamflow at the watershed outlet is taken as the sum of quick flow Q_q and slow flow Q_s. Thus, HyMOD consists of five tunable parameters, $C_{\max}$, b, α, R_q, and R_s.

As an example, a hydrometeorological dataset collected from Leaf River catchment (1944 km^2) located in north of Collins, Mississippi, USA (USGS Station ID 02472000), is used to develop a HyMOD model. The five parameters are calibrated using MATLAB's `ga` function. The dataset includes daily precipitation, streamflow, and potential evapotranspiration time series from 1948 to 2003, all in mm. Data from 1952 to 1962 (inclusive) are used for calibration. The upper and lower bounds of HyMOD parameters are taken from the literature (Moradkhani et al. 2005) and listed in Table 3.3. The value of fitness function is the RMSE between simulated and observed flow rates. Figure 3.9 shows the evolution histories of the mean (+) and the best solution (•) of each generation, obtained using tournament and roulette wheel selection algorithms, respectively. After 51 generations, the RMSE of the best solution returned by the former is 1.1968 mm/day, while for the latter, it is 1.1974 mm/day. The tournament algorithm shows faster convergence in this case. The estimated HyMOD parameters are slightly different, especially for $C_{\max}$. ■

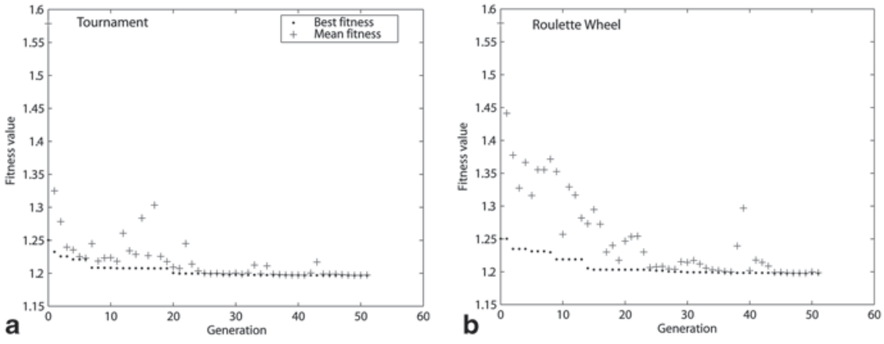


Fig. 3.9 Evolution history of GA by using **a** tournament and **b** roulette wheel selection algorithms

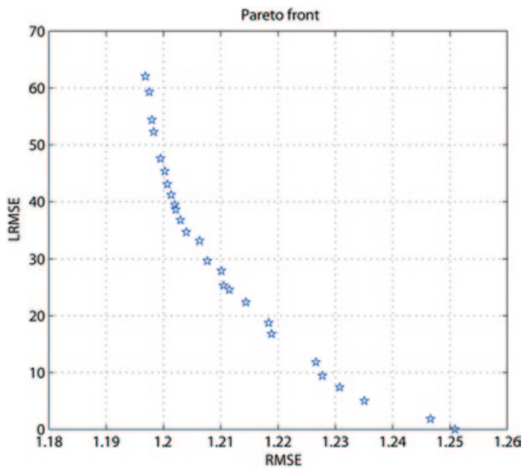


Fig. 3.10 Pareto front for the Leaf River model obtained by using two objective functions, RMSE and LRMSE

In general, GA returns solutions that are close to the global optimum, but are not exact solutions. Also, if there is not sufficient diversity, GA may even return local optimal solutions. Only the RMSE is used in this example for model calibration. In practice, it is common to consider multiple objectives, as discussed in Sect. 3.5. We will demonstrate such a case in the next example.

Example 3.8 *Use GA to solve an MOO problem*
In addition to RMSE, the mean square error of log-transformed flows (LRMSE) is added when calibrating the Leaf River HyMOD model described in Example 3.7. Unlike RMSE that is sensitive to high-flow events, the LRMSE puts more emphasis on minimizing the differences between simulated and observed low-flow events. MATLAB's `gamultiobj` function is used to solve the MOO problem. Figure 3.10 shows the Pareto front after 120 generations.

3.6.3 Multiobjective Evolutionary Algorithm

Both the BnB and traditional GA discussed previously can be used to solve MOO, in a way that a single Pareto-optimal solution is sought each time by solving an SOO problem. The process is then repeated to produce a set of Pareto-optimal solutions. Since the mid-1980s, evolutionary algorithms have been developed to find multiple Pareto-optimal solutions in a single run and, therefore, eliminate the need for running a sequence of SOO problems. These evolutionary algorithms are collectively referred to as the MOEA.

The vector evaluated genetic algorithm (VEGA) developed by Schaffer (1984) represents one of the earliest MOEA. VEGA creates subpopulations within a population, one for each objective. Therefore, if there are m objectives, then the size of each subpopulation is N/m for an N-member population. The individuals in each subpopulation are the fittest for the corresponding objective of the group. The subpopulations are then shuffled to create a new generation, on which the GA crossover and mutation operators are applied and the offspring of a generation are selected. The main limitation of the Schaffer's method is its inability to retain solutions with acceptable performance; also, the solutions in a given generation tend to cluster around the individual minima and the solutions are not evenly distributed along the Pareto front (Marler and Arora 2004).

Goldberg (1989) first introduced the idea of Pareto ranking to move population toward the Pareto front. Under this scheme, individuals in a population that are Pareto nondominated by the rest of the population are assigned the highest rank and are tentatively eliminated from further contention. Individuals in the highest ranking group have the maximum fitness value (usually assigned the same dummy value to avoid loss of diversity) so that they have more chance to be selected during breeding of the next generation. This process is continued for the rest of the population to create groups of ranked solutions; the less fit groups are either discarded or penalized. This implies that at any evolution step, two populations are present, the current population and a tentative set of nondominated solutions, with the latter representing the current approximation of the Pareto front. Solutions from the current population that are nondominated by any solution in the tentative set are added to that set. The Pareto ranking scheme is behind several so-called first-generation MOEA, including multiobjective genetic algorithm (MOGA; Fonseca and Fleming 1993), nondominated sorting genetic algorithm (NSGA; Srinivas and Deb 1994) and niched Pareto genetic algorithm (NPGA; Horn and Nafpliotis 1993).

More recent development of MOEA focuses on (i) explicitly incorporating some type of elitist mechanism in the algorithm, which is important for ensuring the non-degrading performance of MOEA; and (ii) properly and evenly distributing solutions along the Pareto front to maintain diversity. On the latter issue, the use of relaxed Pareto dominance (e.g., e-dominance) has become more popular, so that the user can specify the precision with which they want to quantify each objective in a multiobjective problem. Representative algorithms from this category include the elitist nondominated sorting genetic algorithm (NSGA-II; Deb et al. 2002), strength Pareto EA (SPEA and SPEA-II; Zitzler and Thiele 1999), Pareto archived evolution strategy (PAES; Knowles and Corne 2000), and the honey-bee mating optimization (HBMO; Haddad et al. 2006).

Although we do not attempt to give a detailed account of all MOEA algorithms here, it is important to recognize that the EWR is one of the first fields to adopt MOEA for parameter estimation. Notably, Yapo et al. (1998) proposed a multiobjective complex evolution (MOCOM) method for calibrating hydrological models. MOCOM uses the Pareto ranking scheme suggested by Goldberg (1989; see the above), and the population is evolved using an extension of the downhill simplex search strategy. Reed et al. (2003) developed an ε-NSGA-II method to improve the efficiency of the NSGA-II method by using ε-dominance archiving technique. Under the concept of ε-dominance archiving, the user specifies an ε-grid in the parameter space based on their precision goals. Larger ε values result in a coarser grid, while smaller ε values produce a finer grid. As a result, the algorithm promotes a more uniform search of the objective space. The ε-NSGA-II algorithm and its parallel version have been applied to long-term groundwater monitoring network design and hydrological model calibration (Tang et al. 2007; Hadka and Reed 2012; Kollat et al. 2011). An review of the MOEA applications in EWR is provided by Efstratiadis and Koutsoyiannis (2010).

3.7 Review Questions

1. "Using prior information can increase the accuracy of the inverse solution." Is that statement correct? If not, why should we use it?
2. Can we avoid the two problems, "overfitting the data" and "overbelieving the prior information" simultaneously? Why?
3. Do the weighted sum method and the ε-constraint method give the same set of Pareto optimal solutions?
4. Return to the review question 7 of Chap. 2, try to decrease the number of observation data and/or increase the level of observation error until the inverse solution becomes instable. Then, as in Example 3.1, solve the bicriterion inverse problem in (Eq. 3.2.4) to see how the values of the estimated R and D change with the weighting coefficient α.
5. Verify that the $\boldsymbol{\theta}_\alpha$ in (3.3.10) is indeed the solution of (3.3.9).
6. Can the parameter reduction method used in Example 3.4 be seen as a regularization method? Why?
7. What is the benefit of solving the coupled inverse problem with multistate observations? Do the estimated parameters become more accurate or more stable, or both?
8. The multistate inverse problem in Sect. 3.4 and the multiperformance inverse problem in Sect. 3.5 have the same mathematical form and share the same solution methods, but what is the essential difference between them in the concept?
9. Find a multistate model used in your study area, formulate its inverse problem, and solve the inverse problem with a multiobjective code, such as the MATLAB function `gamultiobj`.

Chapter 4
Statistical Methods for Parameter Estimation

In this chapter, the classical inverse problem (CIP) is formulated in a statistical framework. By using the Bayesian inference theory, we cast the CIP as an information transfer problem, in which the prior information and the information transferred from state observations are combined to reduce uncertainty in the estimated parameters. The prior information is modeled using a probability density function (PDF) called the prior PDF, and the inverse solution is also a PDF known as the posterior PDF. Because it is a PDF, the inverse solution is always existent and unique but with uncertainty. When the posterior PDF is in a relatively simple form, point estimates of the unknown parameters can be readily obtained by solving an optimization problem, just as we have done in the deterministic framework. Further, the statistical framework naturally lends itself to the estimation of confidence intervals. When the posterior PDF has a complex multimodal shape, however, the nonuniqueness and instability issues associated with the inverse solution arise again. For such cases, Monte Carlo sampling methods provide powerful tools for learning the posterior PDFs without requiring knowing their actual functional forms. Using Monte Carlo sampling methods, we can approximately find the posterior PDF (i.e., the inverse solution), evaluate the uncertainty of a point estimate, and assess the reliability of a model application.

In Sect. 4.1, we formulate the CIP using Bayesian inference. Different types of PDFs used for describing the observation error and prior information are introduced. In Sect. 4.2, various point estimators are derived for these different types of PDFs. Statistical properties used for assessing the quality of a point estimator are given. The CIP solutions formulated in Chaps. 2 and 3, including regularization, are explained as point estimates in the statistical framework. The Markov chain Monte Carlo (MCMC), which is a class of Monte Carlo sampling algorithms, is discussed in detail in Sect. 4.3. Two popular MCMC algorithms, the Metropolis–Hastings algorithm and the Gibbs algorithm, are introduced. The application of MCMC for inverse solution and global optimization is also discussed in that section.

N-Z. Sun, A. Sun, *Model Calibration and Parameter Estimation,*
DOI 10.1007/978-1-4939-2323-6_4

4.1 The Statistical Inverse Problem

4.1.1 Statement of the Statistical Inverse Problem

In Chap. 2, the CIP was formulated in a deterministic framework. Because of the uncertain prior knowledge and observation errors, the identified parameter can never be an exact one, even when the model error is not considered. In a statistical framework, we have the following different considerations:

- The parameter identification problem is not considered as the inverse problem of a deterministic forward model; instead, it is considered as an estimation problem of a random parameter (a random vector or a random function). This consideration is based on the stochastic nature of the unknown parameter when it is estimated with incomplete or sparse information.
- The prior knowledge and observation data are treated as information providers—the former reflects the subjective judgment of a modeler, whereas the latter are objective.
- The model serves the role of transferring information from the observation space to parameter space.
- The result of parameter estimation is a PDF of the unknown(s), conditioned on data and prior information.

A short review on probability and statistics is given in Appendix B. For detailed discussions, readers may refer to any textbook on these topics (e.g., Olofsson and Andersson 2012; Walpole 2013). In this chapter, it is assumed that the estimated parameter is an *m*-dimensional random vector denoted by $\boldsymbol{\theta} \in \mathbb{R}^m$. We will use the same notation to represent its realizations (a specific realization is denoted by a subscript or a superscript). The infinite-dimensional case will be considered in Chap. 6, where we will show how a stochastic process can be parameterized and represented by a finite-dimensional parameter vector.

In a statistical framework, the information used for parameter estimation is essentially the same as that used in a deterministic framework. Unlike the deterministic framework, a statistical method naturally gives some estimation of uncertainty. Essential components of the statistical inversion include the following:

- The prior information I, which is cast in a probabilistic framework
- A conditional PDF, $p(\boldsymbol{\theta} \mid I)$, which is called the *prior PDF* of $\boldsymbol{\theta}$
- A set of observations, $\mathbf{u}_D^{obs}$, and the associated random observation error $\mathbf{e}_D$ and the error distribution, $p(\mathbf{e}_D)$
- A set of model outputs $\mathbf{u}_D(\boldsymbol{\theta})$ corresponding to $\mathbf{u}_D^{obs}$

In this chapter, we shall limit ourselves to considering only the CIP (i.e., no model error), which leads to the following observation equation:

$$\mathbf{u}_D^{obs} = \mathbf{u}_D(\boldsymbol{\theta}) + \mathbf{e}_D. \tag{4.1.1}$$

The statistical inversion problem can then be stated as using prior information and the information transferred from observation data to reduce the uncertainty of a parameter being estimated. In what follows, we will proceed to describe:

- How to measure information quantitatively
- How to transfer information from the observation space to parameter space
- How to quantify the uncertainty of an estimated parameter

4.1.2 Information Content and Uncertainty

A continuous random vector $\boldsymbol{\theta}$ is completely characterized by its PDF, $p(\boldsymbol{\theta})$, using which the mean and the covariance of $\boldsymbol{\theta}$ can be calculated, respectively, by

$$\boldsymbol{\mu} = E(\boldsymbol{\theta}) = \int_{\mathbb{R}^m} \boldsymbol{\theta} p(\boldsymbol{\theta}) d\boldsymbol{\theta}, \tag{4.1.2}$$

and

$$\mathbf{C} = E[(\boldsymbol{\theta} - \boldsymbol{\mu})(\boldsymbol{\theta} - \boldsymbol{\mu})^T] = \int_{\mathbb{R}^m} (\boldsymbol{\theta} - \boldsymbol{\mu})(\boldsymbol{\theta} - \boldsymbol{\mu})^T p(\boldsymbol{\theta}) d\boldsymbol{\theta}, \tag{4.1.3}$$

in which $E(\cdot)$ denotes the mathematical expectation. In the classical information theory (Bard 1974; Gray 2011; Kullback 1997), the *uncertainty* associated with $p(\boldsymbol{\theta})$ of a continuous random variables is measured by its *differential entropy*, $\mathcal{H}(p)$, which is defined as

$$\mathcal{H}(p) = -E(\log p) = -\int_{\mathbb{R}^m} p(\boldsymbol{\theta}) \log p(\boldsymbol{\theta}) d\boldsymbol{\theta}, \tag{4.1.4}$$

where $E(\log p)$ is the mathematical expectation of $\log p(\boldsymbol{\theta})$ and the integral is taken over the whole parameter space $\mathbb{R}^m$. The negative value of $\mathcal{H}(p)$, $-\mathcal{H}(p) = E(\log p)$, is called the *information content* associated with $p(\boldsymbol{\theta})$.

Example 4.1 *The uncertainty of a uniform distribution*
The PDF of a uniform distribution over its definition region Ω is given by

$$p(\boldsymbol{\theta}) = \begin{cases} \dfrac{1}{V}, & \text{when } \boldsymbol{\theta} \in \Omega, \\ 0, & \text{otherwise}. \end{cases} \tag{4.1.5}$$

where V is the total volume of region Ω. From (4.1.4), we have

$$\mathcal{H}(p) = -\int_{\Omega} (1/V) \log(1/V) d\boldsymbol{\theta} = \log V. \tag{4.1.6}$$

Therefore, uniform distribution with a larger definition region has a larger uncertainty.

Example 4.2 *The uncertainty of a one-dimensional Gaussian distribution*
The PDF of a one-dimensional or univariate Gaussian distribution is given by

$$p(\theta) \doteq \mathcal{N}(\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2\sigma^2}(\theta-\mu)^2\right], \tag{4.1.7}$$

where μ and σ^2 are the mean and variance of the distribution, respectively, and the $\mathcal{N}$ symbol is used to denote a Gaussian PDF. According to (4.1.4), we have

$$\mathcal{H}(p) = -\int_{-\infty}^{\infty} p(\theta)\log p(\theta)d\theta = \log\sigma + \frac{1}{2}(1+\log 2\pi). \tag{4.1.8}$$

Therefore, a Gaussian distribution with a larger variance tends to have greater uncertainty.

Example 4.3 *The uncertainty of an m-dimensional (or multivariate) Gaussian distribution*
The PDF of an m-dimensional Gaussian distribution is given by

$$p(\boldsymbol{\theta}) \doteq \mathcal{N}(\boldsymbol{\mu},\mathbf{C}) = (2\pi)^{-m/2}(\det\mathbf{C})^{-1/2}\exp\left[-\frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\mu})^T\mathbf{C}^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu})\right], \tag{4.1.9}$$

where $\boldsymbol{\mu}$ and $\mathbf{C}$ are the mean and covariance matrix of $\boldsymbol{\theta}$ as defined in (4.1.2) and (4.1.3), respectively, and the determinant of $\mathbf{C}$, $\det\mathbf{C}$, is called the *generalized variance* of $\boldsymbol{\theta}$. Substituting (4.1.9) into (4.1.4), we get the uncertainty of an m-dimensional Gaussian distribution

$$\begin{aligned}\mathcal{H}(p) &= -E\left[-\frac{m}{2}\log 2\pi - \frac{1}{2}\log(\det\mathbf{C}) - \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\mu})^T\mathbf{C}^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu})\right] \\ &= \frac{m}{2}(1+\log 2\pi) + \frac{1}{2}\log(\det\mathbf{C}).\end{aligned} \tag{4.1.10}$$

Therefore, a multidimensional Gaussian distribution with a larger generalized variance has a larger uncertainty. ■

From Examples 4.2 and 4.3, we see that the uncertainty quantified by entropy measure (4.1.4) is consistent with our intuition—a PDF spanning over a wider region has a larger uncertainty (see Fig. 4.1).

Strictly speaking, the differential entropy defined in (4.1.4) is not a valid extension of the Shannon entropy to the case of continuous random variables. For example, it cannot remain invariant under variable transformation. A modified version of continuous entropy is given by (Jaynes and Bretthorst 2003)

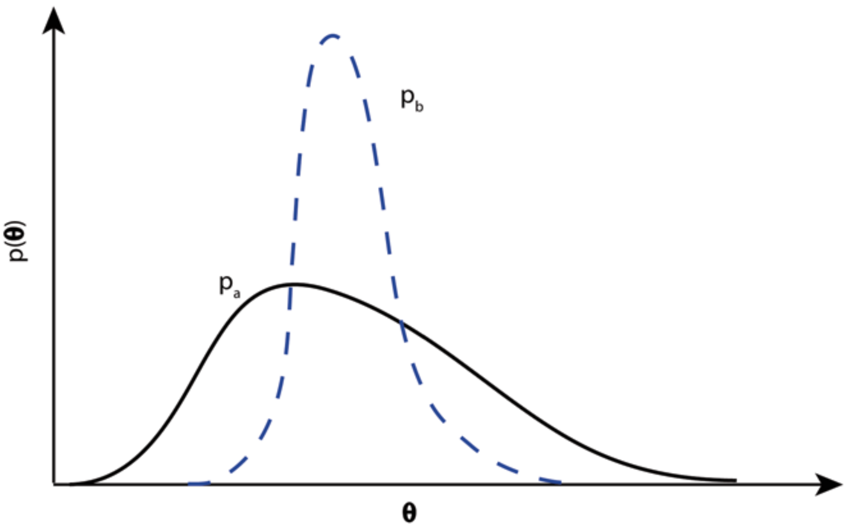


Fig. 4.1 Probability distribution $p_b(\theta)$ has less uncertainty than probability distribution $p_a(\theta)$ does

$$\mathcal{H}_r(p) = -\int_{\mathbb{R}^m} p(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\theta})}{p_M(\boldsymbol{\theta})} d\boldsymbol{\theta}, \tag{4.1.11}$$

where $p_M(\boldsymbol{\theta})$ is the density of a uniform distribution called the invariant measure. Because both $p_M(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta})$ must be transformed in the same way, $\mathcal{H}_r(p)$ is invariant under variable transformation. The entropy defined in (4.1.11) is also known as the *relative information entropy*. More discussions on the Shannon entropy measure will be given in Sect. 10.1, together with other uncertainty measures.

4.1.3 Bayesian Inference for Inverse Solution

Bayesian inference is a statistical inference method that uses prior information I and observation data D as evidence to find the probability of a hypothesis H being true. This can be done by calculating the conditional probability $p(H \mid D, I)$. Applying the product rule of conditional probability gives

$$p(H, D \mid I) = p(D \mid H, I)p(H \mid I) = p(H \mid D, I)p(D \mid I). \tag{4.1.12}$$

Solving $p(H \mid D, I)$ from (4.1.12), we obtain the well-known Bayes' theorem (see Appendix B)

$$p(H \mid D, I) = \frac{p(H \mid I)p(D \mid H, I)}{p(D \mid I)}. \tag{4.1.13}$$

The hypothesis H can be an estimate, a statistical model, or the conclusion from an experiment. This suggests that the Bayesian inference method can be applied to a wide array of inverse problems, not only for parameter estimation, but also for structure identification, model selection, and reliability analysis. In this chapter, H is regarded as the parameter $\boldsymbol{\theta}$ to be estimated and D as the observation data $\mathbf{u}_D^{obs}$. Thus, the Bayes' theorem (4.1.13) can be rewritten in the following form

$$p(\boldsymbol{\theta} \mid \mathbf{u}_D^{obs}, I) = \frac{p(\boldsymbol{\theta} \mid I) p(\mathbf{u}_D^{obs} \mid \boldsymbol{\theta}, I)}{p(\mathbf{u}_D^{obs} \mid I)}, \tag{4.1.14}$$

and the meanings of different quantities are

- $p(\boldsymbol{\theta} \mid I)$ is the prior PDF of $\boldsymbol{\theta}$ and will be denoted by $p_0(\boldsymbol{\theta})$ in the sequel
- $p(\mathbf{u}_D^{obs} \mid \boldsymbol{\theta}, I)$, which is the conditional probability that $\mathbf{u}_D^{obs}$ is observed for a given $\boldsymbol{\theta}$, is called the *likelihood* of $\boldsymbol{\theta}$ and will be denoted by $L(\boldsymbol{\theta})$
- $p(\boldsymbol{\theta} \mid \mathbf{u}_D^{obs}, I)$ is the posterior PDF of $\boldsymbol{\theta}$ conditioned on both prior information and the observed data and will be denoted by $p_*(\boldsymbol{\theta})$
- $p(\mathbf{u}_D^{obs} \mid I)$, the marginal probability distribution $\int_{\mathbb{R}^m} p_0(\boldsymbol{\theta}) L(\boldsymbol{\theta}) d\boldsymbol{\theta}$, plays the role of a normalization constant and its value will be denoted as $1/c$.

Using the above notations, we can simply write the Bayes' theorem (4.1.14) as

$$p_*(\boldsymbol{\theta}) = c p_0(\boldsymbol{\theta}) L(\boldsymbol{\theta}). \tag{4.1.15}$$

Equation (4.1.15) is called the *Bayesian inference* for inverse solution. It serves as the foundation for the statistical CIP and will be used repeatedly throughout this chapter.

Using (4.1.4) and (4.1.14), we can calculate the uncertainties associated with both prior and posterior distributions and the difference between them, which is given by

$$\mathcal{H}(p_0) - \mathcal{H}(p_*) = \int_{\mathbb{R}^m} [p_*(\boldsymbol{\theta}) \log p_*(\boldsymbol{\theta}) - p_0(\boldsymbol{\theta}) \log p_0(\boldsymbol{\theta})] d\boldsymbol{\theta}. \tag{4.1.16}$$

Equation (4.1.16) provides a measure of the reduction in uncertainty after transferring information from observation data; its opposite, $\mathcal{H}(p_*) - \mathcal{H}(p_0)$, measures the information content that is transferred.

The posterior PDF, $p_*(\boldsymbol{\theta})$, is the solution of Bayesian inference. It expresses our degree of belief regarding the true value of $\boldsymbol{\theta}$ in light of new observations or samples. With $p_*(\boldsymbol{\theta})$, we can calculate the probability of $\boldsymbol{\theta}$ falling in any subregion (Ω) of the parameter space by evaluating the following integral

$$P(\boldsymbol{\theta} \in \Omega) = \int_{\Omega} p_*(\boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{4.1.17}$$

Equation (4.1.17) is often used to quantify the confidence intervals associated with our estimate by specifying the appropriate subregion (Ω) for integration. Note that finding $p_*(\boldsymbol{\theta})$ by Bayesian inference is always a well-posed problem because it is given by (4.1.15) explicitly and uniquely, regardless of how much prior information and observation data we have. If the quantity and quality of information is sufficient, $p_*(\boldsymbol{\theta})$ will define a narrow neighborhood around the true parameter; otherwise, $p_*(\boldsymbol{\theta})$ will spread over a larger region. In the latter case, the result of parameter estimation has a large uncertainty associated with it (Fig. 4.1).

Example 4.4 *Bayesian inference for Gaussian prior and likelihood function*
Analytical expressions of the posterior PDF can only be derived for a few special distributions, and the Gaussian distribution is one of them. Let us consider a univariate case and assume that the prior PDF and the likelihood function are both Gaussian, namely,

$$p_0(\theta) = \mathcal{N}(\mu_0, \sigma_0),\ L(\theta) = \mathcal{N}(\mu_1, \sigma_1). \tag{4.1.18}$$

Because the product of two Gaussian distributions is also Gaussian, from the Bayes' theorem (4.1.15) we can deduce that the posterior is also a Gaussian, $p_*(\theta) = \mathcal{N}(\mu, \sigma)$, and its mean and variance are given, respectively, by

$$\mu = \frac{\sigma_0^2 \sigma_1^2}{\sigma_0^2 + \sigma_1^2}\left(\frac{\mu_0}{\sigma_0^2} + \frac{\mu_1}{\sigma_1^2}\right) = \frac{\sigma_1^2}{\sigma_0^2 + \sigma_1^2}\mu_0 + \frac{\sigma_0^2}{\sigma_0^2 + \sigma_1^2}\mu_1, \tag{4.1.19}$$

$$\sigma^2 = \frac{\sigma_0^2 \sigma_1^2}{\sigma_0^2 + \sigma_1^2}. \tag{4.1.20}$$

The second equality in (4.1.19) suggests the posterior mean is a weighted mean of the prior mean and the mean of observations, with the weights given by the variances.

As an illustration, suppose that we want to estimate the concentration of a pollutant in an aquifer. The prior PDF of the pollutant concentration is assumed to be Gaussian. Its mean and standard deviation are estimated to be 20 and 10 mg/L, respectively, based on studies of similar sites and expert opinions. Suppose that we then take a number of samples and conduct a statistical analysis of the samples, through which the mean and standard deviation are calculated to be 18.7 and 3.5 mg/L, respectively. The statistical analysis also suggests that it is appropriate to model the data distribution as Gaussian, which leads to Gaussian likelihood function $L(\theta) = \mathcal{N}(18.7, 3.5)$. The posterior mean and standard deviation can then be easily calculated using (4.1.19) and (4.1.20), and the results are 18.8 and 3.3 mg/L, respectively. In this case, the posterior is closer to the likelihood function than to the prior (see Fig. 4.2a).

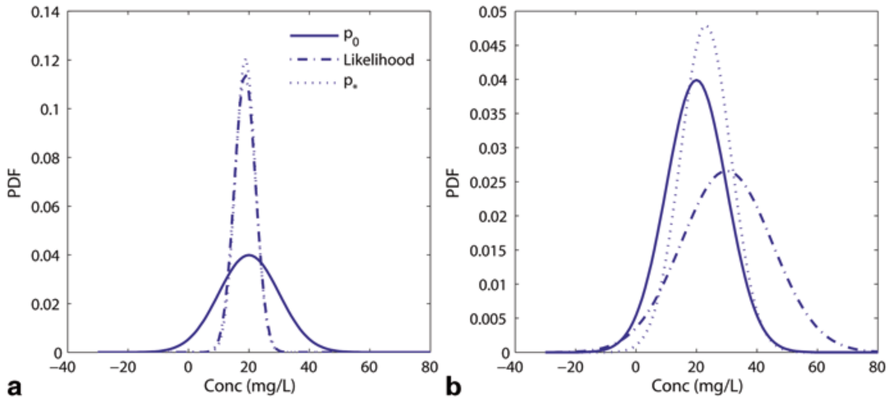


Fig. 4.2 Role of observations in Bayesian inference. **a** The posterior is closer to the likelihood function than to the prior, **b** the posterior is closer to the prior than to the likelihood function

If the likelihood function is $L(\theta) = \mathcal{N}(30,15)$ instead, the posterior mean and standard deviation become 23.1 and 8.3 mg/L, respectively, which are closer to the statistics of the prior (see Fig. 4.2b).

4.1.4 Probability Distribution of Observation Error

Observational error usually has two additive parts: system (or instrument) error and random error (or white noise). The former is caused by the drift of measurement instrument from its normal operating conditions, whereas the latter is caused by uncontrollable stochastic factors. In what follows, we assume that the observation error consists of only random error, while the system error has been eliminated by correction. According to the central limit theorem in statistics, when the same physical quantity is measured many times, the histogram should be close to a normal distribution. But the data collected for model calibration and parameter identification are not from repeated experiments or repeated measurements of a fixed physical quantity. Instead, they are taken at different locations and times. In this case, the randomness is actually used to represent uncertainty. Note that Gaussian distribution is certainly not always the best choice for characterizing uncertainty. For example, let c represent the number of precipitation events in a day; in this case, the observation error associated with the event count c is better modeled using a Poisson distribution. If we do not have extra knowledge, however, the Gaussian distribution can be used because it represents the least compromising choice for describing random observation error (Demoment and Goussard 2010).

Let $\mathbf{e}_D$ be the observation error associated with a set of n observation data. When the error distribution is chosen to be Gaussian with zero mean, we have

$$p(\mathbf{e}_D) = (2\pi)^{-n/2}(\det \mathbf{C}_D)^{-1/2} \exp\left[-\frac{1}{2}\mathbf{e}_D{}^T\mathbf{C}_D^{-1}\mathbf{e}_D\right], \tag{4.1.21}$$

where $\mathbf{C}_D$ is the covariance matrix of the joint PDF of observation errors.

Because the mean of Gaussian distribution (4.1.21) is assumed zero, only its covariance needs to be determined. Usually, $\mathbf{C}_D$ is characterized by a set of parameters $\boldsymbol{\psi}$ called *hyperparameters*, a term used to refer parameters of statistical distributions. Hyperparameters may be treated as additional unknowns in the case of insufficient prior information. For example, when the components of $\mathbf{e}_D$ are independent of each other, $\mathbf{C}_D$ becomes a diagonal matrix and the hyperparameter vector $\boldsymbol{\psi}$ consists of the variances of all n components,

$$\boldsymbol{\psi} = (\sigma_{D,1}^2, \sigma_{D,2}^2, \cdots, \sigma_{D,n}^2), \tag{4.1.22}$$

where $\sigma_{D,i}^2$ is the variance of the *i*-th component of $\mathbf{e}_D$. Further, when all components of $\mathbf{e}_D$ are independent and identically distributed (i.i.d.), $\boldsymbol{\psi}$ reduces to a single scalar, σ_D^2. When the components of $\mathbf{e}_D$ are not independent, however, more hyperparameters are needed for determining the nondiagonal elements of $\mathbf{C}_D$. In Chap. 7, we will show how to identify hyperparameters using the observation data. In this chapter, however, we assume that $p(\mathbf{e}_D)$ is specified.

Because there is no model error in CIP, we can establish the following equality based on the observation equation (4.1.1),

$$L(\boldsymbol{\theta}) = p(u_D^{obs} \mid \boldsymbol{\theta}, I) = p(\mathbf{e}_D).$$

Furthermore, when $\mathbf{e}_D$ follows a Gaussian distribution, we have the following likelihood function based on (4.1.21),

$$\begin{aligned} L(\boldsymbol{\theta}) &= (2\pi)^{-n/2} (\det \mathbf{C}_D)^{-1/2} \\ &\times \exp\left\{ -\frac{1}{2} [\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta})]^T \mathbf{C}_D^{-1} [\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta})] \right\}. \end{aligned} \tag{4.1.23}$$

If $p(\mathbf{e}_D)$ is not Gaussian, $L(\boldsymbol{\theta})$ will have different expressions. For example, if all components of $\mathbf{e}_D$ are independent and the mean deviation η_i $(i = 1, 2, \cdots, n)$ can be estimated for each component, we can use the following L_1-norm-based likelihood function

$$L(\boldsymbol{\theta}) = \left[\prod_{i=1}^{n} \frac{1}{2\eta_i} \right] \exp\left\{ -\sum_{i=1}^{n} \frac{\left| u_{D,i}^{obs} - u_{D,i}(\boldsymbol{\theta}) \right|}{\eta_i} \right\}. \tag{4.1.24}$$

Readers may refer to Pawitan (2001) and Sprott (2000) for more examples of the likelihood function.

4.1.5 Probability Distribution of Prior Information

Selecting a proper prior distribution is essential for Bayesian inference; however, it can also be a challenging problem because it usually requires translating qualitative information into quantitative forms. If an inappropriate prior distribution is used in the Bayesian inference, an incorrect inverse solution may be generated. Therefore, the choice of a prior distribution should be based on existing data and rely on as little subjective inputs as possible.

A common form of prior information is given as an admissible region (Ω) in the parameter space, for example, an m-dimensional hyperbox $[\mathbf{L}, \mathbf{U}]$ defined by upper and lower bounds of the estimated parameter vector $\boldsymbol{\theta}$. Because we have no other information in this case, we should use the principle of indifference to assign equal probability to all points in the admissible region. The prior distribution is thus a uniform distribution, as defined in Example 4.1:

$$p_0(\boldsymbol{\theta}) = \frac{1}{V}, \text{ when } \boldsymbol{\theta} \in \Omega;\ p_0(\boldsymbol{\theta}) = 0, \text{ otherwise}, \tag{4.1.25}$$

where V is the volume of (Ω). Uniform distribution in a region is not a noninformative prior because it contains strong information: the unknown parameters must be in a given range. Designating the range subjectively may produce biased estimation.

Another often used prior distribution is the Gaussian distribution (4.1.9), in which an initial guess $\boldsymbol{\theta}_0$ is used as the mean and its uncertainty is characterized by a prior covariance matrix $\mathbf{C}_P$

$$p_0(\boldsymbol{\theta}) = (2\pi)^{-m/2}(\det \mathbf{C}_P)^{-1/2} \exp\left\{-\frac{1}{2}[\boldsymbol{\theta} - \boldsymbol{\theta}_0]^T \mathbf{C}_P^{-1}[\boldsymbol{\theta} - \boldsymbol{\theta}_0]\right\}. \tag{4.1.26}$$

As yet another example, the L_1-norm-based prior distribution (also called the Laplace distribution) is given by

$$p_0(\boldsymbol{\theta}) = \left[\prod_{i=1}^{m} \frac{1}{2\alpha_i}\right] \exp\left\{-\sum_{i=1}^{m} \frac{|\theta_i - \theta_{0,i}|}{\alpha_i}\right\}, \tag{4.1.27}$$

where $\theta_{0,i}$ is a prior estimate of θ_i, and α_i is a scale parameter.

Gaussian prior is a special case of exponential conjugate priors. A *conjugate prior* of a likelihood function means the produced posterior distribution has the same functional structure as the prior distribution. As we have shown in Example 4.4, when the likelihood is Gaussian, the posterior will also be Gaussian, provided that the prior is chosen to be Gaussian. Other non-Gaussian prior distributions will be introduced in Chap. 6. For a detailed discussion on conjugate priors and noninformative priors, the readers may refer to Robert (2007). An in-depth analysis of prior distributions for Bayesian inference is recently given in Kitanidis (2012).

Like in the case of likelihood functions, various prior distributions have hyperparameters associated with them, such as the mean $\boldsymbol{\theta}_p$ and covariance matrix $\mathbf{C}_P$ in the Gaussian distribution (4.1.26) and the scale parameters $(\alpha_1, \alpha_2, \cdots, \alpha_m)$ in the L_1-norm prior distribution (4.1.27). In practice, these hyperparameters need to be estimated from observation data as well, because prior knowledge is generally insufficient to determine them. The topic of hyperparameter estimation will be covered in Chap. 7. In this chapter, we simply assume $p_0(\boldsymbol{\theta})$ is given.

4.2 Point Estimation

4.2.1 Maximum a Posteriori Estimate

In the previous section, we showed that the posterior distribution $p_*(\boldsymbol{\theta})$ is the inverse solution under the Bayesian inference framework, namely,

$$p_*(\boldsymbol{\theta}) = cp_0(\boldsymbol{\theta})L(\boldsymbol{\theta}). \tag{4.2.1}$$

We also discussed how to select the prior distribution $p_0(\boldsymbol{\theta})$ and the likelihood function $L(\boldsymbol{\theta})$. However, after $p_0(\boldsymbol{\theta})$ and $L(\boldsymbol{\theta})$ are selected, finding the posterior distribution can still be a challenging problem when dimensionality of the parameter space is high. We will return to this topic later in Sect. 4.3.

In the deterministic framework, the goal of inversion is to seek an approximate solution $\hat{\boldsymbol{\theta}}$ in the parameter space that is as close as possible to the true parameter and, thus, can be used to replace the true parameter for model application. In the statistical framework, ideally we would like to obtain a $p_*(\boldsymbol{\theta})$ that forms a narrow region around the true parameter and is unimodal. In this case, the *maximum a posteriori* (MAP) estimate is defined by

$$\boldsymbol{\theta}_{MAP} = \arg\max_{\boldsymbol{\theta}}\left\{p_*(\boldsymbol{\theta})\right\}, \tag{4.2.2}$$

where $\boldsymbol{\theta}_{MAP}$ is such a point in the parameter space that has the maximum probability to be the true parameter for given observations and prior information. In other words, we would like to find the mode of the posterior distribution. Finding MAP is an optimization problem that can be solved by a search process without the need to know the full posterior distribution. Toward this end, we substitute (4.2.1) into (4.2.2) and take the logarithm of $p_*(\boldsymbol{\theta})$ to obtain

$$\boldsymbol{\theta}_{MAP} = \arg\max_{\boldsymbol{\theta}}\left\{\log L(\boldsymbol{\theta}) + \log p_0(\boldsymbol{\theta})\right\}. \tag{4.2.3}$$

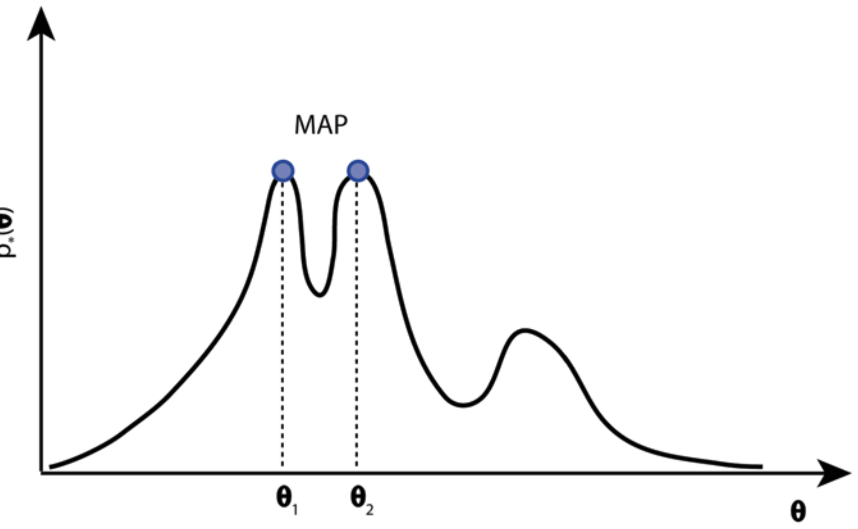


Fig. 4.3 A multimodal posterior distribution, where the MAP estimates are not unique

The MAP estimate $\boldsymbol{\theta}_{MAP}$ is a point in the parameter space and, therefore, is called a *point estimate* or a *point estimator* of the true parameter. In the following subsections, we will introduce more point estimators under different statistical assumptions and for different forms of $L(\boldsymbol{\theta})$ and $p_0(\boldsymbol{\theta})$.

In practice, however, finding the MAP by solving a problem as given in (4.2.3) is not always possible. As shown in Fig. 4.3, even though $p_*(\boldsymbol{\theta})$ is uniquely determined, it may not be convex, and its global maximum may not be unique, when the information content provided for inversion is insufficient. In this case, the ill-posedness of inversion can again be seen and the search for point estimates becomes meaningless. An in-depth discussion of this aspect will be provided in Sect. 4.3.

A point estimator $\hat{\boldsymbol{\theta}}$ always contains uncertainty because it is determined by incomplete information. Thus, $\hat{\boldsymbol{\theta}}$ may also be treated as a random variable. The PDF of $\hat{\boldsymbol{\theta}}$ is called a *sampling distribution* and is denoted by $p(\hat{\boldsymbol{\theta}})$. For a given $p(\hat{\boldsymbol{\theta}})$, we can calculate its sample mean $E(\hat{\boldsymbol{\theta}})$ and sample variance $Var(\hat{\boldsymbol{\theta}})$. Then, we can assess its quality according to the following statistical properties:

- The *bias* of an estimator $\hat{\boldsymbol{\theta}}$ is defined by $B(\hat{\boldsymbol{\theta}}) = E(\hat{\boldsymbol{\theta}}) - \boldsymbol{\theta}^t$, where $\boldsymbol{\theta}^t$ is the true parameter to be estimated. When the bias vanishes, $\hat{\boldsymbol{\theta}}$ is called an unbiased estimator.
- The *mean squared error* (MSE) of an estimator $\hat{\boldsymbol{\theta}}$ is defined by

$$MSE(\hat{\boldsymbol{\theta}}) = E[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^t)^2] = B^2(\hat{\boldsymbol{\theta}}) + Var(\hat{\boldsymbol{\theta}}). \tag{4.2.4}$$

For an unbiased estimator, MSE is the sampling variance. The square root of MSE (or RMSE) has the same unit as the estimated parameter. A small RMSE means the estimator, on average, is close to the true parameter.

- An ideal point estimator is one that is unbiased and has the minimum MSE. But such an estimator may not exist because a biased estimator may have a lower MSE than any unbiased estimator. An unbiased estimator is called a *minimum-variance unbiased estimator or the best unbiased estimator* if it has the lowest variance compared to other unbiased estimators. According to (4.2.4), such an estimator has the lowest MSE among all unbiased estimators.

More detailed discussion on these and other statistical properties of estimators, such as efficiency, sufficiency, consistency, and robustness, can be found, for example, in Beck and Arnold (1977), Anderson and Moore (1979), and Lehmann and Casella (1998).

Note that the sampling distribution of an estimator depends not only on the estimator itself, but also on the model structure and probability distribution of the observation error. It can be found only for some simple cases, for example, when the model $u(\boldsymbol{\theta})$ is linear and $p(\mathbf{e}_D)$ is Gaussian. When a model is nonlinear and the observation data are limited, the sampling distribution $p(\hat{\boldsymbol{\theta}})$ is very difficult to obtain and, as a result, the assessment of a statistical estimator can become a challenging task.

4.2.2 Estimators With Uniform Prior Distribution

In the case of uniform prior distributions, the prior knowledge can only be used to define an admissible region, but within the region it is not informative. The estimation problem is, thus, only based on the information provided by observation data, namely, criterion (C-1) of inversion. Because $\log p_0(\boldsymbol{\theta})$ is a constant in this case, the MAP estimator given in (4.2.3) turns into the following *maximum likelihood* estimator (MLE)

$$\boldsymbol{\theta}_{MLE} = \arg\max_{\boldsymbol{\theta}}\left\{\log L(\boldsymbol{\theta})\right\} = \arg\min_{\boldsymbol{\theta}}\left\{-\log L(\boldsymbol{\theta})\right\}, \boldsymbol{\theta} \in P_{ad}. \tag{4.2.5}$$

When the observation error follows a Gaussian distribution, MLE becomes the *generalized least squares* (GLS) estimator after substituting the Gaussian likelihood function (4.1.21) into (4.2.5),

$$\boldsymbol{\theta}_{GLS} = \arg\min_{\boldsymbol{\theta}}[\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta})]^T \mathbf{C}_D^{-1}[\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta})], \boldsymbol{\theta} \in P_{ad}. \tag{4.2.6}$$

When all observations are independent of each other, the GLS estimator further reduces to the *weighted least squares* (WLS) estimator

$$\boldsymbol{\theta}_{WLS} = \arg\min_{\boldsymbol{\theta}}[\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta})]^T \mathbf{W}[\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta})], \boldsymbol{\theta} \in P_{ad}, \tag{4.2.7}$$

which coincides with the quasi-solution defined in the last chapter when the weighted L_2-norm is used in the observation space, but the diagonal of the weight matrix

$\mathbf{W}$ is now specified as $\left\{\sigma_{D,1}^{-2}, \sigma_{D,2}^{-2}, \cdots, \sigma_{D,n}^{-2}\right\}$. Furthermore, when all error variances are the same (i.e., in the case of i.i.d. observation errors), the WLS estimator reduces to the *ordinary least squares* (OLS) estimator

$$\boldsymbol{\theta}_{OLS} = \arg\min_{\boldsymbol{\theta}} [\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta})]^T [\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta})], \boldsymbol{\theta} \in P_{ad}. \tag{4.2.8}$$

The OLS estimator is identical to the inverse solution of (2.3.1) given in Chap. 2 when L_2-norm is used. Now we know that using the OLS criterion for inversion needs to meet certain statistical assumptions:

- The observation error $\mathbf{e}_D$ should be normally distributed with zero mean and all components of $\mathbf{e}_D$ should be i.i.d.
- When the components of $\mathbf{e}_D$ have different variances, the WLS estimator should be used.
- When the components of $\mathbf{e}_D$ are correlated, the GLS estimator should be used.

From OLS to GLS, more and more hyperparameters in the covariance matrix $\mathbf{C}_D$ need to be determined. If we define $(\|\bullet\|_{\mathbf{C}_D}^2 = \bullet^T \mathbf{C}_D^{-1} \bullet)$ as the generalized L_2-norm in the observation space, then GLS, WLS, and OLS estimators can be expressed using a unified form

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \left\| \mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta}) \right\|_{\mathbf{C}_D}^2, \boldsymbol{\theta} \in P_{ad}. \tag{4.2.9}$$

When the observation error is not Gaussian, we have to derive other estimators according to the selected likelihood function. For example, using the L_1-norm-based likelihood function given in (4.1.24), we have the following weighted L_1-norm (WL1) estimator

$$\boldsymbol{\theta}_{WL1} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \left| u_{D,i}^{obs} - u_{D,i}(\boldsymbol{\theta}) \right| / \eta_i, \boldsymbol{\theta} \in P_{ad}. \tag{4.2.10}$$

This estimator is more robust than the L_2-norm-based estimators when the observation dataset has outliers.

4.2.2.1 Linear Regression

When the model is linear with respect to the estimated parameter, the parameter estimation problem is called *linear regression*. In this case, we have $\mathbf{u}_D(\boldsymbol{\theta}) = \mathbf{G}\boldsymbol{\theta}$, where $\mathbf{G}$ is an $n \times m$ matrix. Using the necessary condition of minimization, we obtain

$$\mathbf{G}^T \mathbf{C}_D^{-1} (\mathbf{u}_D^{obs} - \mathbf{G}\boldsymbol{\theta}) = 0. \tag{4.2.11}$$

Solving this equation, we get an explicit GLS estimator

$$\boldsymbol{\theta}_{GLS} = \mathbf{B}\mathbf{u}_D^{obs}, \text{ where } \mathbf{B} = (\mathbf{G}^T\mathbf{C}_D^{-1}\mathbf{G})^{-1}\mathbf{G}^T\mathbf{C}_D^{-1}. \tag{4.2.12}$$

From this explicit expression and (4.2.11), we can find the mean of $\boldsymbol{\theta}_{GLS}$

$$E(\boldsymbol{\theta}_{GLS}) = (\mathbf{G}^T\mathbf{C}_D^{-1}\mathbf{G})^{-1}\mathbf{G}^T\mathbf{C}_D^{-1}E(\mathbf{u}_D^{obs}) = \boldsymbol{\theta}, \tag{4.2.13}$$

and the covariance matrix of $\boldsymbol{\theta}_{GLS}$

$$\begin{aligned} \operatorname{Cov}(\boldsymbol{\theta}_{GLS}) &= E\left\{[\boldsymbol{\theta}_{GLS} - E(\boldsymbol{\theta}_{GLS})][\boldsymbol{\theta}_{GLS} - E(\boldsymbol{\theta}_{GLS})]^T\right\} \\ &= \mathbf{B}E\left\{[\mathbf{u}_D^{obs} - \mathbf{G}\boldsymbol{\theta}][\mathbf{u}_D^{obs} - \mathbf{G}\boldsymbol{\theta}]^T\right\}\mathbf{B}^T \\ &= \mathbf{B}\mathbf{C}_D\mathbf{B}^T = (\mathbf{G}^T\mathbf{C}_D^{-1}\mathbf{G})^{-1}. \end{aligned} \tag{4.2.14}$$

The final equality on the right hand of (4.2.14) is based on $\mathbf{BG} = \mathbf{I}$ derived from (4.2.12). Equation (4.2.13) means $\boldsymbol{\theta}_{GLS}$ is an unbiased estimator, while (4.2.14) means the variance of $\boldsymbol{\theta}_{GLS}$ reaches the lower bound of variances among all unbiased estimators (the Gauss–Markov theorem) (Golub and Van Loan 1996). Therefore, for a linear model, $\boldsymbol{\theta}_{GLS}$ is the best unbiased estimator.

4.2.3 Estimators With Gaussian Prior Distribution

When the information content of the prior distribution is more than a uniform distribution in the admissible region, a better MAP estimate can be obtained because both (C-1) and (C-2) criteria are used for inversion. As a commonly seen special case, we assume that the probability distributions of both observation error and prior information are Gaussian. After substituting (4.1.23) and (4.1.26) into (4.2.2), we obtain the following *Conjugate Gaussian* (CG) estimator

$$\begin{aligned} \boldsymbol{\theta}_{CG} = \arg\min_{\boldsymbol{\theta}} \Big\{ &[\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta})]^T\mathbf{C}_D^{-1}[\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta})] \\ &+ [\boldsymbol{\theta} - \boldsymbol{\theta}_0]^T\mathbf{C}_P^{-1}[\boldsymbol{\theta} - \boldsymbol{\theta}_0]\Big\}. \end{aligned} \tag{4.2.15}$$

When $\mathbf{C}_D = \sigma_D^2\mathbf{I}$ and $\mathbf{C}_P = \sigma_P^2\mathbf{I}$, the above equation reduces to

$$\boldsymbol{\theta}_{CG} = \arg\min_{\boldsymbol{\theta}}\left\{\left\|\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta})\right\|_2^2 + \alpha\left\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\right\|_2^2\right\}, \boldsymbol{\theta} \in P_{ad}. \tag{4.2.16}$$

Equation (4.2.16) is exactly identical to the Tikhonov regularization problem (3.3.7), if the coefficient α is set to $\alpha = \sigma_P^2 / \sigma_D^2$. More generally, when the observation error distribution and prior distribution are not Gaussian, but can be expressed by exponential functions as

$$L(\boldsymbol{\theta}) \propto \exp\left\{-\frac{1}{\lambda_D^2} G(\boldsymbol{\theta})\right\} \text{ and } p_0(\boldsymbol{\theta}) \propto \exp\left\{-\frac{1}{\lambda_P^2} R(\boldsymbol{\theta})\right\}, \tag{4.2.17}$$

the MAP estimate (4.29) becomes the general Tikhonov regularization (GTR) solution described under Sect. 3.3

$$\boldsymbol{\theta}_{GTR} = \arg\min_{\boldsymbol{\theta}} \left\{G(\boldsymbol{\theta}) + \alpha R(\boldsymbol{\theta})\right\}, \boldsymbol{\theta} \in P_{ad}, \tag{4.2.18}$$

where $\alpha = \lambda_D^2 / \lambda_P^2$, and $G(\boldsymbol{\theta})$ and $R(\boldsymbol{\theta})$ are norm terms. The discussion herein casts the Tikhonov regularization method in the statistical framework, and the problem of determining the optimal regularization coefficient is turned into estimating hyperparameters, σ_D^2 and σ_P^2, in the likelihood function and the prior distribution, respectively.

4.2.3.1 Linear Regression With Regularization

For a linear model $\mathbf{u}(\boldsymbol{\theta}) = \mathbf{G}\boldsymbol{\theta}$, an explicit expression for the mean of $\boldsymbol{\theta}_{CG}$ can be found by solving (4.2.15)

$$E(\boldsymbol{\theta}_{CG}) = (\mathbf{G}^T \mathbf{C}_D^{-1} \mathbf{G} + \mathbf{C}_P^{-1})^{-1} (\mathbf{G}^T \mathbf{C}_D^{-1} \mathbf{u}_D^{obs} + \mathbf{C}_P^{-1} \boldsymbol{\theta}_0), \tag{4.2.19}$$

and the corresponding covariance matrix is given by

$$\mathrm{Cov}(\boldsymbol{\theta}_{CG}) = (\mathbf{G}^T \mathbf{C}_D^{-1} \mathbf{G} + \mathbf{C}_P^{-1})^{-1}. \tag{4.2.20}$$

Applying the Woodbury identity for matrix inversion (Golub and Van Loan 1996), we can derive the following alternative expressions for the mean and variance of $\boldsymbol{\theta}_{CG}$

$$E(\boldsymbol{\theta}_{CG}) = \boldsymbol{\theta}_0 + \mathbf{C}_P \mathbf{G}^T (\mathbf{G}\mathbf{C}_P \mathbf{G}^T + \mathbf{C}_D)^{-1} (\mathbf{u}_D^{obs} - \mathbf{G}\boldsymbol{\theta}_0), \tag{4.2.21}$$

$$\mathrm{Cov}(\boldsymbol{\theta}_{CG}) = \mathbf{C}_P - \mathbf{C}_P \mathbf{G}^T (\mathbf{C}_D + \mathbf{G}\mathbf{C}_P \mathbf{G}^T)^{-1} \mathbf{G}\mathbf{C}_P. \tag{4.2.22}$$

From (4.2.21), it can be observed that $\boldsymbol{\theta}_{CG}$ is a biased estimator because of the use of prior information and regularization.

4.2.4 Minimum Relative Entropy Estimator

The Kullback–Leibler (K–L) *relative entropy* between one distribution $p(\boldsymbol{\theta})$ and another distribution $q(\boldsymbol{\theta})$ is defined by (Jaynes 1982; Kullback and Leibler 1951)

$$\mathcal{K}(p;q) = \int_{\mathbb{R}^m} p(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta}. \tag{4.2.23}$$

It has the similar form as the entropy measure defined in (4.1.11), which differ by a minus sign. However, they are different concepts. Minimizing $\mathcal{K}(p;q)$ means making $p(\boldsymbol{\theta})$ as close to a fixed distribution $q(\boldsymbol{\theta})$ as possible. Especially, when $q(\boldsymbol{\theta})$ is a given prior distribution, we have

$$\mathcal{K}(p;p_0) = \int_{\mathbb{R}^m} p(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\theta})}{p_0(\boldsymbol{\theta})} d\boldsymbol{\theta}. \tag{4.2.24}$$

Therefore, minimizing $\mathcal{K}(p;p_0)$ means honoring the prior information as much as possible, which is criterion (C-2) of inversion. From the point of view of multicriterion optimization, the fitting data criterion (C-1) can be used as a constraint for inverse solution. Assuming that the mean of observation error is zero, from the observation equation (4.1.1), we have n constraints:

$$u_{D,i}^{obs} = E[u_{D,i}(\boldsymbol{\theta})] = \int_{\mathbb{R}^m} p(\boldsymbol{\theta}) u_{D,i}(\boldsymbol{\theta}) d\boldsymbol{\theta},\ i = 1, 2, \cdots, n. \tag{4.2.25}$$

The *minimum relative entropy* (MRE) estimation processor proceeds in two steps. In the first step, the unknown probability distribution $p(\boldsymbol{\theta})$ is found by minimizing $\mathcal{K}(p;p_0)$, subject to the n constraints in (4.2.25) (and the higher-order moment constraints if used) and the normality constraint

$$\int_{\mathbb{R}^m} p(\boldsymbol{\theta}) d\boldsymbol{\theta} = 1. \tag{4.2.26}$$

In the second step, the MRE estimate is obtained by

$$\boldsymbol{\theta}_{MRE} = \int_{\mathbb{R}^m} \boldsymbol{\theta} p(\boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{4.2.27}$$

Because $p(\boldsymbol{\theta})$ contains the prior information and the information transferred from data, it is actually a posterior distribution and, therefore, the MRE estimator gives the posterior mean estimate. If there is no error in the data, the MRE estimates are identical to the Bayesian estimates.

The constrained optimization problem for finding the posterior $p(\boldsymbol{\theta})$ can be solved by using the method of Lagrange multipliers. The Lagrange function for this case is defined by

$$\begin{aligned} L(p,\eta,\lambda) &= \mathcal{K}(p,p_0) + \sum_{i=1}^{n} \lambda_i \left(\int_{\mathbb{R}^m} p(\boldsymbol{\theta}) u_{D,i}(\boldsymbol{\theta}) d\boldsymbol{\theta} - u_{D,i}^{obs} \right), \\ &+ \eta \left(\int_{\mathbb{R}^m} p(\boldsymbol{\theta}) d\boldsymbol{\theta} - 1 \right) \end{aligned} \qquad (4.2.28)$$

where λ_i and η are Lagrange multipliers. For a complete review on this approach, the reader is referred to Woodbury (2011). MRE has been used to identify release history of contaminant sources (Woodbury and Ulrych 1996), hydraulic conductivity (Hou and Rubin 2005), and the leakage pathways in geologic carbon repositories (Sun and Nicot 2012). A Matlab implementation of MRE is provided by Neupauer and Borchers (2001).

4.2.5 Bayesian Inversion for Multistate Models

In the coupled inverse problem considered in Sect. 3.4.2, we have k state variables, $\{u_1(\boldsymbol{\theta}), u_2(\boldsymbol{\theta}), \cdots, u_k(\boldsymbol{\theta})\}$, and k sets of observation data, $\{D_1, D_2, \cdots, D_k\}$, for inversion, where $D_i = \mathbf{u}_{D,i}^{obs}$ ($i = 1, 2, \cdots, k$). In this case, the Bayesian theorem in (4.1.14) has the following extended form

$$\begin{aligned} &p(\boldsymbol{\theta} \mid D_1, D_2, \ldots, D_k, I) \propto p(\boldsymbol{\theta} \mid I) p(D_1, D_2, \cdots, D_k \mid \boldsymbol{\theta}, I) \\ &= p(\boldsymbol{\theta}) p(D_1 \mid \boldsymbol{\theta}, I) p(D_2 \mid D_1, \boldsymbol{\theta}, I) p(D_3 \mid D_1, D_2, \boldsymbol{\theta}, I) \ldots \\ &\cdots p(D_k \mid D_1, D_2, \cdots, D_{k-1}, \boldsymbol{\theta}, I). \end{aligned} \qquad (4.2.29)$$

Then, all discussions in this section so far can be easily extended to the multistate inverse problems. For example, the CG estimator in (4.2.15) is extended to

$$\begin{aligned} \boldsymbol{\theta}_{CG} = \arg\min_{\boldsymbol{\theta}} \Bigg\{ &\sum_{i=1}^{k} \left[\mathbf{u}_{D,i}^{obs} - \mathbf{u}_{D,i}(\boldsymbol{\theta}) \right]^T \mathbf{C}_{D,i}^{-1} \left[\mathbf{u}_{D,i}^{obs} - \mathbf{u}_{D,i}(\boldsymbol{\theta}) \right] \\ &+ \left[\boldsymbol{\theta} - \boldsymbol{\theta}_p \right]^T \mathbf{C}_P^{-1} \left[\boldsymbol{\theta} - \boldsymbol{\theta}_p \right] \Bigg\}. \end{aligned} \qquad (4.2.30)$$

This solution is equivalent to the regularized multistate inversion in Sect. 3.4.2 when the covariance matrices are used as weights.

4.3 Monte Carlo Methods for Bayesian Inference

4.3.1 Exploring the Posterior Distribution

In Sect. 4.1, the posterior distribution is defined as the inverse solution in the statistical framework

$$p_*(\boldsymbol{\theta}) = cp_0(\boldsymbol{\theta})L(\boldsymbol{\theta}). \tag{4.3.1}$$

In Sect. 4.2, point estimators are derived based on different criteria. When $p_*(\boldsymbol{\theta})$ has a Gaussian-like shape (single modal and nearly symmetric), point estimates obtained via solving an optimization problem can be good approximations of the unknown parameter. Because Bayesian inference uses incomplete information for parameter estimation, the shape of $p_*(\boldsymbol{\theta})$ may be highly complex (e.g., asymmetric and with multiple modes and long tails). As a result, the following difficulties often arise in practice:

- Estimates obtained by a point estimator, such as the MAP, may be nonunique, although the posterior distribution is uniquely determined by (4.3.1). This is already illustrated in Fig. 4.3. If we are limited to considering only point estimates, we will face the same ill-posed problem as that in the deterministic framework.
- When the posterior distribution does not have a Gaussian-like shape, a point estimate does not make much sense. For example, Fig. 4.3 suggests that no single point estimate can be a good approximation of the true parameter: the mean of the distribution is not representative of the unknown parameter, and the variance of the distribution is not a useful measure of the uncertainty.

In principle, these difficulties can go away if we have access to the full posterior distribution, instead of merely point estimates. A complete picture of the posterior distribution helps determine whether a point estimate makes sense and how reliable it is. For example, we can obtain the MAP estimate directly based on the plot of posterior distribution. We can then accept the estimate if the shape of $p_*(\boldsymbol{\theta})$ is limited to a narrow region surrounding it; conversely, if we find out that the shape of $p_*(\boldsymbol{\theta})$ is too disperse, new data should be collected.

The posterior distribution can also be used to quantify the uncertainty. Let $\mathbf{g}(\boldsymbol{\theta})$ be a cost function consisting of, for example, a set of objectives of the model application, the mean of model estimated $\mathbf{g}(\boldsymbol{\theta})$ can be obtained by calculating the integral

$$E(\mathbf{g}) = \int_{\mathbb{R}^m} \mathbf{g}(\boldsymbol{\theta})p_*(\boldsymbol{\theta})d\boldsymbol{\theta}, \tag{4.3.2}$$

and the uncertainty of the estimation can be evaluated by calculating the covariance

$$\mathrm{Cov}(\mathbf{g}) = \int_{\mathbb{R}^m} [\mathbf{g}(\boldsymbol{\theta}) - E(\mathbf{g})][\mathbf{g}(\boldsymbol{\theta}) - E(\mathbf{g})]^T p_*(\boldsymbol{\theta})d\boldsymbol{\theta}. \tag{4.3.3}$$

The advantage of using Bayesian inference for inversion now becomes clear. If we could find out the shape of the full posterior distribution and calculate the integrals (4.3.2) and (4.3.3), then the parameter estimation problem, the uncertainty assessment problem, and the data sufficiency problem can all be addressed. Unfortunately, except for a few limited cases (e.g., Gaussian prior and likelihood), the two tasks are analytically intractable, especially when the model is nonlinear and the dimension of $\boldsymbol{\theta}$ is high. More often than not, numerical methods have to be used.

4.3.2 Markov Chain Monte Carlo Sampling Techniques

4.3.2.1 Monte Carlo Integration

As the mean of a random variable, (4.3.2) can be calculated approximately by its sample mean

$$E(\mathbf{g}) \approx \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}(\boldsymbol{\theta}_i), \tag{4.3.4}$$

where $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_N\}$ in $\mathbb{R}^m$ is a set of points (samples) drawn from the PDF $p_*(\boldsymbol{\theta})$. According to the law of large numbers, (4.3.4) becomes accurate when N approaches to infinity. In fact, such a random sampling method provides a general tool for numerical integration. For an integrand function $h(\boldsymbol{\theta})$, if it can be decomposed into the product of a function $f(\boldsymbol{\theta})$ and a PDF $p(\boldsymbol{\theta})$ defined over a region (Ω), then we have

$$\int_{\Omega} h(\boldsymbol{\theta})d\boldsymbol{\theta} = \int_{\Omega} f(\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} = E[f] \approx \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{\theta}_i), \tag{4.3.5}$$

where $p(\boldsymbol{\theta})$ is called a *target distribution* and (4.3.5) is referred to as *Monte Carlo integration*. The problem is how we can strategically draw the samples to make the Monte Carlo integration more efficient or converge faster. Both homogeneous sampling and i.i.d. random sampling methods can be inefficient because they may take a lot of samples from those regions where the target distribution $p(\boldsymbol{\theta})$ is close to zero. In what follows, we introduce the MCMC methods that can guide sampling to regions of significant probability.

Although the original concept of MCMC was introduced in the middle of the last century, it gained significant momentum in the last two decades because of the advance of computing power. MCMC algorithms now play an influential role in statistical inference, machine learning, and decision analysis. For a detailed account of the history of MCMC, interested readers may refer to the review papers and textbooks by Liu (2001), Robert and Casella (2004), Berge et al. (1995), and Andrieu et al. (2005).

To explore a target distribution $p(\boldsymbol{\theta})$ over its definition region, an MCMC algorithm constructs a series of samples or a *Markov chain*

$$\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}, \cdots \tag{4.3.6}$$

where the sample $\boldsymbol{\theta}_{i+1}$ is generated by a random walk rule based on the current sample $\boldsymbol{\theta}_i$, but is independent of all other samples before $\boldsymbol{\theta}_i$. This is equivalent to assuming a first-order Markov process. A Markov process is uniquely defined by the probability of moving to $\boldsymbol{\theta}_{i+1}$ from $\boldsymbol{\theta}_i$ which is called the *transition kernel* and often denoted by $P(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1})$, $P(\boldsymbol{\theta}_i \to \boldsymbol{\theta}_{i+1})$, or the conditional probability, $P(\boldsymbol{\theta}_{i+1} \mid \boldsymbol{\theta}_i)$, in the literature. The main idea behind MCMC is to construct a Markov chain (4.3.6) that converges to samples drawn from the target distribution after a large number of steps. This is guaranteed by the following *detailed balance condition*

$$p(\boldsymbol{\theta}_i)P(\boldsymbol{\theta}_i \to \boldsymbol{\theta}_{i+1}) = p(\boldsymbol{\theta}_{i+1})P(\boldsymbol{\theta}_{i+1} \to \boldsymbol{\theta}_i), \tag{4.3.7}$$

which requires that each transition $\boldsymbol{\theta}_i \to \boldsymbol{\theta}_{i+1}$ is reversible (Robert and Casella 2004). Integrating (4.3.7) over the definition region, we have

$$\begin{aligned} &\int_\Omega p(\boldsymbol{\theta}_i)P(\boldsymbol{\theta}_i \to \boldsymbol{\theta}_{i+1})d\boldsymbol{\theta}_i = \int_\Omega p(\boldsymbol{\theta}_{i+1})P(\boldsymbol{\theta}_{i+1} \to \boldsymbol{\theta}_i)d\boldsymbol{\theta}_i \\ &= p(\boldsymbol{\theta}_{i+1})\int_\Omega P(\boldsymbol{\theta}_{i+1} \to \boldsymbol{\theta}_i)d\boldsymbol{\theta}_i = p(\boldsymbol{\theta}_{i+1}). \end{aligned} \tag{4.3.8}$$

Equation (4.3.8) indicates that once a sample $\boldsymbol{\theta}_i$ from the target distribution is found, all subsequent samples will be distributed according to the target distribution. As a result, the Monte Carlo integration can be calculated efficiently.

The Metropolis–Hastings algorithm (Hastings 1970; Metropolis et al. 1953) and the Gibbs algorithm (Geman and Geman 1984), to be introduced below, are the two best known MCMC algorithms.

4.3.2.2 Metropolis–Hastings Algorithm

The Metropolis–Hastings algorithm is a simple but powerful MCMC sampler that serves as the basis of many practical MCMC algorithms. To generate a new sample $\boldsymbol{\theta}_{i+1}$ based on the current sample $\boldsymbol{\theta}_i$, the Metropolis–Hastings algorithm uses a *proposal distribution*, $q(\boldsymbol{\theta}_i \to \boldsymbol{\theta})$, to draw a candidate $\boldsymbol{\theta}'$. This is especially useful when the target distribution cannot be directly simulated, such as the posterior distribution $p_*(\boldsymbol{\theta})$ used in the calculation of (4.3.2) and (4.3.3) for Bayesian inversion. A Gaussian distribution with mean $\boldsymbol{\theta}_i$ and covariance $\mathbf{C}$, for example, may serve as a proposal distribution. In this case, the randomly drawn $\boldsymbol{\theta}'$ will be around the current $\boldsymbol{\theta}_i$ with a dispersed distance depending on the value of $(\det \mathbf{C})^{1/2}$. Based on the value of an acceptance probability, the Metropolis–Hastings algorithm then determines whether to move to $\boldsymbol{\theta}'$ or stay with the current value. The sample series generated in this way will converge to the target distribution after a *burn-in* period. More specifically, a generic Metropolis–Hastings algorithm starts from an arbitrary initial sample point $\boldsymbol{\theta}_1$ satisfying $p(\boldsymbol{\theta}_1) > 0$, and finds $\boldsymbol{\theta}_{i+1}$ from $\boldsymbol{\theta}_i$ by implementing the following steps:

1. Use the selected proposal distribution to draw a candidate $\boldsymbol{\theta}'$ and calculate the *Metropolis ratio* r given by

$$r = \frac{p(\boldsymbol{\theta}')}{p(\boldsymbol{\theta}_i)} \cdot \frac{q(\boldsymbol{\theta}' \to \boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i \to \boldsymbol{\theta}')}. \tag{4.3.9}$$

   Note that for symmetric proposal distributions (e.g., Gaussian), $q(\boldsymbol{\theta}_i \to \boldsymbol{\theta}')$ is equal to $q(\boldsymbol{\theta}' \to \boldsymbol{\theta}_i)$.

2. Calculate the acceptance probability

$$\alpha(\boldsymbol{\theta}_i \to \boldsymbol{\theta}') = \min(1, r). \tag{4.3.10}$$

3. Accept $\boldsymbol{\theta}'$ with probability $\alpha(\boldsymbol{\theta}_i \to \boldsymbol{\theta}')$ or reject $\boldsymbol{\theta}'$ with probability $1 - \alpha(\boldsymbol{\theta}_i \to \boldsymbol{\theta}')$. If $\boldsymbol{\theta}'$ is accepted, let $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}'$; if $\boldsymbol{\theta}'$ is rejected, let $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i$ and return to step 1 to generate another candidate.

From the above algorithm, we see that the transition kernel for the Metropolis–Hastings algorithm is given by

$$P(\boldsymbol{\theta}_i \to \boldsymbol{\theta}_{i+1}) = q(\boldsymbol{\theta}_i \to \boldsymbol{\theta}_{i+1})\alpha(\boldsymbol{\theta}_i \to \boldsymbol{\theta}_{i+1}). \tag{4.3.11}$$

Using (4.3.10), it is easy to show that

$$p(\boldsymbol{\theta}_i)q(\boldsymbol{\theta}_i \to \boldsymbol{\theta}_{i+1})\alpha(\boldsymbol{\theta}_i \to \boldsymbol{\theta}_{i+1}) = p(\boldsymbol{\theta}_{i+1})q(\boldsymbol{\theta}_{i+1} \to \boldsymbol{\theta}_i)\alpha(\boldsymbol{\theta}_{i+1} \to \boldsymbol{\theta}_i). \tag{4.3.12}$$

By combining the above two equations, the detailed balance equation (4.3.7) is obtained. Therefore, after discarding the samples in a "burn-in" period to eliminate the effect of the initial sample, the Markov chain constructed by the Metropolis–Hastings algorithm converges to the target distribution.

Besides its simplicity, another major advantage of the Metropolis–Hastings algorithm is that the target distribution $p(\boldsymbol{\theta})$ can be replaced by function $cp(\boldsymbol{\theta})$, where c is any constant. The difficulty of normalization is thus avoided. Unfortunately, there is no general theoretical result for determining the number of samples required. The effectiveness of the algorithm is also a problem. Using larger step size for exploring may require fewer samples but result in lower acceptance rate, while using smaller step size may require more samples but fewer rejections. Roberts and Rosenthal (1998) recommended that an appropriate acceptance rate should be in the range 25–50 %. When the proposal distribution is a Gaussian distribution, we can use this recommendation to adjust the value of the spread hyperparameter (i.e., standard deviation) adaptively during the sampling process.

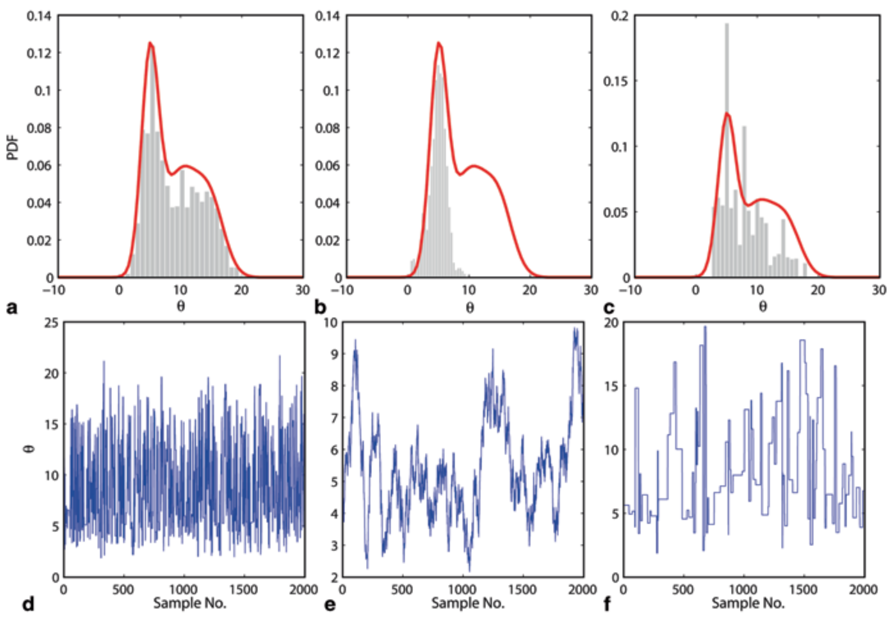


Fig. 4.4 Effect of the spread of proposal distribution on Metropolis–Hastings sampling: (**a**) $\sigma = 10$, (**b**) $\sigma = 0.25$, and (**c**) $\sigma = 80$. The corresponding trace plots are given in (**d**)–(**f**).

Example 4.5 *Using the Metropolis–Hastings algorithm to approximate a PDF*
To illustrate how Metropolis–Hastings algorithm works, let us assume that we know a univariate PDF $p(\theta)$ up to some normalizing constant

$$p(\theta) \propto \exp\left(-\frac{(\theta-5)^2}{4}\right) + 0.3\exp\left(-\frac{(\theta-15)^2}{10}\right) \\ +0.5\exp\left(-\frac{(\theta-10)^2}{18}\right). \tag{4.3.13}$$

The Gaussian distribution is used as the proposal distribution $q(\theta_i \to \theta) = \mathcal{N}(\theta_i, \sigma)$. Let us start from an initial guess $\theta_0 = 5$ and assume the standard deviation of the Gaussian distribution σ is 10. With this information, we draw a random sample from the proposal distribution $\mathcal{N}(5,10)$, say, 13.6. The corresponding Metropolis–Hastings ratio is then calculated using (4.3.9)

$$r = \frac{p(13.6)}{p(5)} = \frac{0.49}{1.12} = 0.44. \tag{4.3.14}$$

Thus, the acceptance probability α for this sample is 0.44 according to (4.3.10). Then, we draw a random sample from the uniform distribution, $\mathcal{U}(0,1)$, say, $u = 0.63$. In this case, the candidate is rejected because $\alpha < u$. We then draw an-

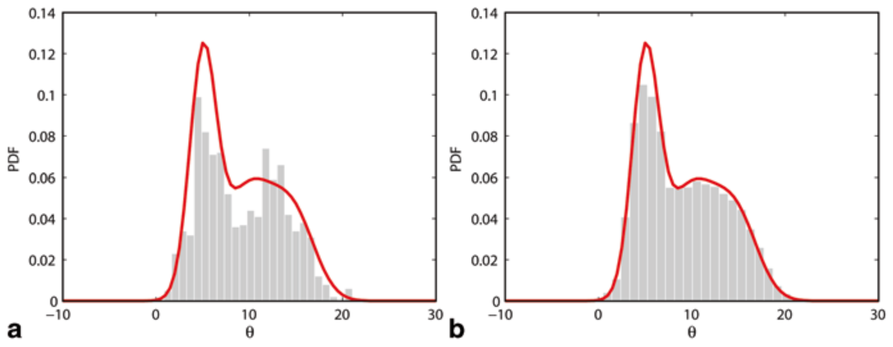


Fig. 4.5 Effect of sample sizes. **a** 1000, **b** 20000

other sample from the proposal distribution and repeat the above process many times.

In this example, the first 100 samples were discarded (i.e., burn-in period), after which 2000 more random samples were generated. Figure 4.4a–c shows the results for σ equal to 10, 0.25, and 80, respectively, where the solid line is the actual PDF and the bar plots are the frequency histograms of random samples.

The hyperparameter σ can have a significant effect. When σ is too small compared to the actual spread (Fig. 4.4b), the samples tend to cluster in a narrow region and fail to explore the other significant modes, although the acceptance rate is high (90 %). On the other hand, when σ is too large (Fig. 4.4c), the samples are too dispersed and the acceptance rate is also low (10 %). The acceptance rate corresponding to Fig. 4.4a is 44 %.

Trace plots (i.e., a plot of sample value vs. sample number) can be used to check the convergence of MCMC. In this case, the trace plot (Fig. 4.4d) corresponding to Fig. 4.4a provides an example of a well-mixed Markov chain, while the trace plot corresponding to Fig. 4.4b is an example of a poorly mixed chain. The trace plot corresponding to Fig. 4.4c gives a typical example of low acceptance rates, in which we see a stair-function-like behavior because of contiguous sample rejections.

As discussed in the text, the sample size also plays a significant role on the goodness of fit. Figs. 4.5a–b show that when the sample size increases from 1000 to 20000, the resulting histogram becomes significantly smoother. As an exercise, the reader can also test the effects of initial guess and the length of burn-in period.

4.3.2.3 Gibbs Algorithm

The Gibbs algorithm devises a random walk from the *i*-th sample $\boldsymbol{\theta}_i$ to the next sample $\boldsymbol{\theta}_{i+1}$ in a sequence of movements along the coordinate directions. Let $\boldsymbol{\theta}$ be an *m*-dimensional variable, $\boldsymbol{\theta} = (\theta_1, \theta_2, \cdots, \theta_m)$ and let us denote the i-th sample as $\boldsymbol{\theta}_i = (\theta_1^{(i)}, \theta_2^{(i)}, \cdots, \theta_m^{(i)})$, where the superscripts in parentheses are used to indicate the sample number.

The Gibbs sampling uses the following steps to move from $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}_{i+1}$:

1. Draw a sample for each component of $\boldsymbol{\theta}_i$ from the full conditional distribution

$$\theta_k^{(i+1)} \sim p(\theta_k \mid \theta_1^{(i+1)}, \theta_2^{(i+1)}, \cdots, \theta_{k-1}^{(i+1)}, \theta_{k+1}^{(i)}, \ldots, \theta_m^{(i)}).$$

Thus, $\theta_1^{(i+1)}$ is drawn from $p(\theta_1 \mid \theta_2^{(i)}, \theta_3^{(i)}, \cdots, \theta_m^{(i)})$, $\theta_2^{(i+1)}$ is drawn from $p(\theta_2 \mid \theta_1^{(i+1)}, \theta_3^{(i)}, \cdots, \theta_m^{(i)})$, and so on. Note that

- The full conditional distribution is assumed known and is used as the proposal distribution; otherwise, Metropolis–Hastings steps can be embedded using the procedure described in the last subsection.
- The acceptance probability of the Gibbs sampler is always one.
- Each new component sample is used immediately to condition the sampling of the next component.

2. Set the new sample as $\boldsymbol{\theta}_{i+1} = (\theta_1^{(i+1)}, \theta_2^{(i+1)}, \cdots, \theta_m^{(i+1)})$.

From the above process, we see that the transition kernel of the Gibbs algorithm is given by

$$P(\boldsymbol{\theta}_i \to \boldsymbol{\theta}_{i+1}) = \prod_{k=1}^{m} p(\theta_k \mid \theta_1^{(i+1)}, \theta_2^{(i+1)}, \cdots, \theta_{k-1}^{(i+1)}, \theta_{k+1}^{(i)}, \cdots, \theta_m^{(i)}). \qquad (4.3.15)$$

With this transition kernel, the same Eqs. (4.3.12) and (4.3.8) can be derived. Therefore, the distribution of Gibbs samples approaches the target distribution after a burn-in period. Although there is no rejection in the Gibbs sampling process, its computational effort is high when the dimensionality m is large, because m movements are needed to generate a new sample of $\boldsymbol{\theta}$.

Example 4.6 *Gibbs sampling*
A popular application of the Gibbs sampler is to estimate parameters of a PDF. In this example, we would like to estimate the unknown mean μ and standard deviation σ of a Gaussian distribution,

$$\mathcal{N}(\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right).$$

For reasons to be clear soon, we define a new variable, $\lambda \doteq \sigma^{-2}$, which is often known as the precision. Thus, the unknown variable is $\boldsymbol{\theta} = (\mu, \lambda)$. To use the Gibbs sampler, we first need to define the full conditional PDFs, which are used as proposal distributions for μ and λ

$$p(\mu \mid \lambda, \mathbf{x}_o) \text{ and } p(\lambda \mid \mu, \mathbf{x}_o), \qquad (4.3.16)$$

where $\mathbf{x}_o = \{x_{o,j}\}_{j=1}^{n}$ represents a sample dataset. Derivation of (4.3.16) requires knowledge of prior PDFs. The commonly used priors for μ and λ are Gaussian and gamma distribution, respectively (Besag et al. 1995):

$$\begin{aligned} &\mu \propto \mathcal{N}(m_\mu, s), \\ &\lambda \propto Gamma(a,b) = \frac{b^a}{\Gamma(a)} \lambda^{a-1} \exp(-b\lambda), \end{aligned} \tag{4.3.17}$$

where m_μ and s are the mean and standard deviation of μ; a and b are shape and rate parameters of the gamma distribution, and $\Gamma(\cdot)$ is the gamma function. For the selected priors and assuming Gaussian likelihood, we can derive the full conditional distributions as

$$p(\mu|\sigma, \mathbf{x}_o) \propto \mathcal{N}(m_1, s_1), \tag{4.3.18}$$

$$p(\lambda|\mu, \mathbf{x}_o) \propto Gamma(a_1, b_1), \tag{4.3.19}$$

where

$$\begin{aligned} m_1 &= \frac{m_\mu / s^2 + (n/\sigma^2)\bar{x}}{1/s^2 + n/\sigma^2}, \\ s_1^2 &= \frac{1}{1/s^2 + n/\sigma^2}, \\ a_1 &= a + n/2, \\ b_1 &= \frac{1}{2}\sum_{i=1}^{n} (x_i - \mu)^2 + b. \end{aligned} \tag{4.3.20}$$

and $\bar{x}$ is the sample mean of $\mathbf{x}_o$.

As a solid example, let us assume that the "true" parameters are $\mu = 11, \sigma = 2$. We generated $n = 50$ samples from $\mathcal{N}(\mu = 11, \sigma = 2)$ as our data. The corresponding sample mean is calculated as $\bar{x} = 10.87$. In addition, the parameters of the priors are given as

$$m_\mu = 9,\ s = 3,\ a = 3,\ b = 0.7.$$

Starting with an initial value of $\sigma = 1.5$ (or $\lambda = 0.44$), we used the formulae in (4.3.18)–(4.3.20) to perform Gibbs sampling of μ and λ repeatedly. The length of the burn-in period is 500, and the total number of Gibbs samples is 5000. The results of λ are converted to σ at the end.

Figure 4.6a and b show the resulting histograms for μ and σ, respectively. After incorporating the observation data, the posterior mean of μ moved from its

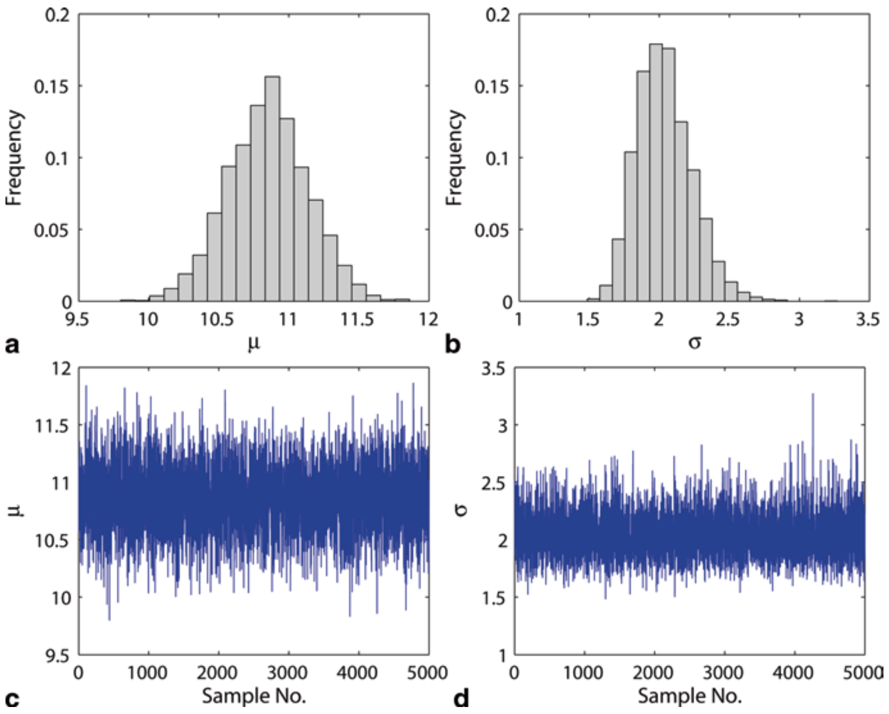


Fig. 4.6 Relative frequency histogram of the Markov chains generated using the Gibbs sampler for **a** mean, μ, **b** standard deviation, σ, **c**–**d** corresponding trace plots

prior mean of 9 to 10.85 and the posterior mean of σ moved from its prior mean 3 to 2.03. Figure 4.6c and d show the corresponding trace plots for both parameters. In practice, the Gibbs sampling procedure delineated here is often combined with the Metropolis–Hastings sampler in a hierarchical Bayesian framework, in which the latter is used to sample PDFs of physical parameters and the former is used to sample PDFs of hyperparameters. A full exposition of the formulation and use of hierarchical Bayesian framework for inversion will be provided in Chap. 7. ■

Over the years, many variants of Metropolis–Hastings and Gibbs samplers have been proposed to improve the performance or for special applications. Readers may refer to Liu (2001), Robert and Casella (2004), Berg (2004), and Bolstad (2010) for further readings.

Implementations of Metropolis–Hastings sampler, Gibbs sampler, and their variants can be found in many packages, including Matlab, R, and Python. Matlab's own Statistical Toolbox, for example, includes a Metropolis–Hastings sampler and random number generators for different distributions. The Bayesian inference using Gibbs sampling (BUGS) project (openBUGS) is a widely used open-source MCMC toolbox (http://www.openbugs.info/w/); in particular, it can be used to perform Gibbs sampling even when closed forms of the full conditional distributions are not available.

4.3.3 Sampling for Inverse Solution

MCMC is an effective method for exploring any target distribution with or without an analytical expression. Therefore, it is widely used to explore the posterior distribution for Bayesian inference. Especially, when the model is nonlinear and/or the posterior distribution is non-Gaussian, MCMC provides one of the few approaches for performing Bayesian inversion.

Now let us go back to our discussion of the Metropolis–Hastings sampler. Suppose that the target distribution represents the posterior distribution, namely, $p_*(\boldsymbol{\theta}) = cp_0(\boldsymbol{\theta})L(\boldsymbol{\theta})$, the ratio r in (4.3.9) can then be written as

$$r = \frac{p_0(\boldsymbol{\theta}')}{p_0(\boldsymbol{\theta}_i)} \cdot \frac{L(\boldsymbol{\theta}')}{L(\boldsymbol{\theta}_i)} \cdot \frac{q(\boldsymbol{\theta}' \to \boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i \to \boldsymbol{\theta}')}. \tag{4.3.21}$$

When the prior distribution is uniform in a region P_{ad} and the proposal distribution is symmetric, both p_0 and $q(\cdot)$ cancel out and the ratio r becomes the ratio of likelihood functions

$$r = \frac{L(\boldsymbol{\theta}')}{L(\boldsymbol{\theta}_i)}, \ \boldsymbol{\theta}' \in P_{ad}. \tag{4.3.22}$$

In this case, we only need to explore the likelihood function. In general, of course, we have to explore both the prior distribution and the likelihood function. There are case-dependent techniques to increase the effectiveness of exploration and decrease the dependence between samples. For example, after taking one sample of the prior distribution, we take many samples of the likelihood function (Tarantola 2005); when a prior guess of the unknown parameter is available, we can use it as the initial sample to shorten the burn-in period. We can also use multiple Markov chains starting from different initial samples to avoid getting trapped at a local mode and missing out significant unexplored regions (Gilks et al. 1998).

After exploring the posterior distribution using MCMC, we obtain a set of samples or a Markov chain

$$\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_i, \cdots, \boldsymbol{\theta}_N, \tag{4.3.23}$$

where N is the total number of samples (excluding the burn-in period). An important question that we must answer is whether the Markov chain has converged or when the sampling process can be terminated. There are more than ten convergence diagnostic tools presented in the references, but none of them can give us a definitive answer (Brooks and Roberts 1998; Cowles and Carlin 1996). In Examples 4.5 and 4.6, we showed that simple trace plots (sample value vs. sample index) can be very useful in visually assessing convergence. Through examining the trace plots, we can see whether a chain has converged to a stationary distribution and whether a longer burn-in period is necessary. The Gelman–Rubin test (Gelman and Rubin 1992) uses multiple chains to test whether they all converge to the same target distribution; failure could indicate the presence of a multimodal posterior distribution or the need to take more samples. Yet another example is the Geweke test (Geweke 1992) that compares the sample mean of the first n_F samples (e.g., first 10 % of the samples) to that of the last n_L samples (e.g., last 50 % of all samples); significant difference between the two may indicate that the sampling chain has not reached a stationary distribution.

The samples generated by an MCMC method are not independent of each other. When the samples of a chain are poor mixing, the chain may explore only a small region of the parameter space. The independence between samples can be measured by a simple autocovariance defined for each component $k = 1, 2, \cdots, m$ of the parameter vector $\boldsymbol{\theta}$. For a lag h in sample indices, the sample autocovariance function is given by

$$\gamma_k(h) = \frac{1}{N-h}\sum_{i=1}^{N-h}(\theta_k^{(i)} - \bar{\theta}_k)(\theta_k^{(i+h)} - \bar{\theta}_k),\ 0 \le h < N, \tag{4.3.24}$$

where $\bar{\theta}_k = \frac{1}{N}\sum_{i=1}^{N}\theta_k^{(i)}$ is the sample mean of component k, and the sample autocorrelation is defined by $\rho_k(h) = \gamma_k(h) / \gamma_k(0)$. If $\rho_k(h)$ drops quickly with the increase in lag h for all components, we can conclude that the chain is good mixing and the convergence would be fast. Note that the convergence of different components may not be isochronous. Thus, we have to check the autocorrelations for all components.

An MCMC-based inverse problem solver can give us the following results:

- Once we have the MCMC samples (4.3.23), we can use them to get additional statistics. For example, we can calculate the sample mean, an estimate of the *posterior mean*, by

$$\boldsymbol{\theta}_{PM} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\theta}_i. \tag{4.3.25}$$

Equation (4.3.25) gives us a smoothed estimation of the unknown parameter. If we do not have these MCMC samples, as we have mentioned before, the sample mean and sample covariance would be very difficult to calculate for nonlinear models.

- From these MCMC samples, we can find all modes, the tail, and high probability regions of the posterior distribution. Once we find from the samples that a point estimate makes sense, we can easily output this point estimate and calculate its uncertainty.

Moreover, after the cost function of model application $\mathbf{g}(\boldsymbol{\theta})$ is given, we can use the MCMC samples to approximate $E(\mathbf{g})$ in (4.3.2) by

$$E(\mathbf{g}) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}(\boldsymbol{\theta}_i), \tag{4.3.26}$$

and $\mathrm{Cov}(\mathbf{g})$ in (4.3.3) by

$$\mathrm{Cov}(\mathbf{g}) = \frac{1}{N-1}\sum_{i=1}^{N}[\mathbf{g}(\boldsymbol{\theta}_i) - E(\mathbf{g})][\mathbf{g}(\boldsymbol{\theta}_i) - E(\mathbf{g})]^T. \tag{4.3.27}$$

In summary, the MCMC sampling process is simple and very useful, but can be computationally expensive, especially when the dimension of the unknown parameter is high. In the Metropolis–Hastings algorithm, testing each candidate sample needs to solve the forward problem once, regardless of whether the candidate can be accepted or rejected. In the Gibbs algorithm, the forward problem needs to be solved m times for generating a new sample. Another difficulty of using MCMC is the formulation of prior distribution. In practice, we can use the original posterior distribution as the prior distribution and then incorporate new data to obtain an updated posterior distribution.

Bayesian inference using MCMC sampling is now widely used in EWR modeling. Modified versions of the standard MCMC algorithms and their applications in hydrological and geo-hydrological modeling are given by Oliver et al. (1997), Vrugt et al. (2003), Feyen et al. (2007), Blasone et al. (2008), Fu and Gómez-Hernández (2009), Vrugt et al. (2009), Liu et al. (2010), among others. Review papers on this topic are given by Mosegaad and Sambridge (2002), Hendricks Franssen et al. (2009), and Yustres et al. (2012).

The differential evolution adaptive metropolis (DREAM) toolbox (available in Matlab and Python) developed by J. Vrugt and his coworkers runs multiple different chains simultaneously for global exploration, and automatically tunes the proposal distribution using differential evolution (Vrugt et al. 2009). Another Matlab-based MCMC toolbox is MCMCStat developed by Haario et al. (2006), which implements an adaptive Metropolis–Hastings algorithm and can be easily adopted for parameter estimation.

Example 4.7 *Estimate parameters of a HyMOD model*
In this example, the five parameters of the HyMOD model introduced in Example 3.7 are calibrated using MCMC. Here the unknown parameter vector is denoted as $\boldsymbol{\theta} \doteq (C_{\max}, b, \alpha, R_q, R_s)$ and the lower and upper bounds of the parameters are the same as those given before in Table 3.3 for the Leaf River watershed. The initial guesses of the five parameters are $\boldsymbol{\theta}_0$=(220, 0.5, 0.9, 0.02, 0.6). The priors of all parameters are assumed as Gaussian with statistics

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}=(220, 0.5, 0.9, 0.02, 0.6) \text{ and } \boldsymbol{\sigma}_{\boldsymbol{\theta}}=(100, 1.0, 0.1, 0.1, 0.3).$$

For this example, the Metropolis–Hastings algorithm in MCMCStat toolbox (Haario et al. 2006) is used and the likelihood function is set to the sum-of-squares between observed and simulated streamflows. The length of burn-in period is 500 and the number of samples is 5000.

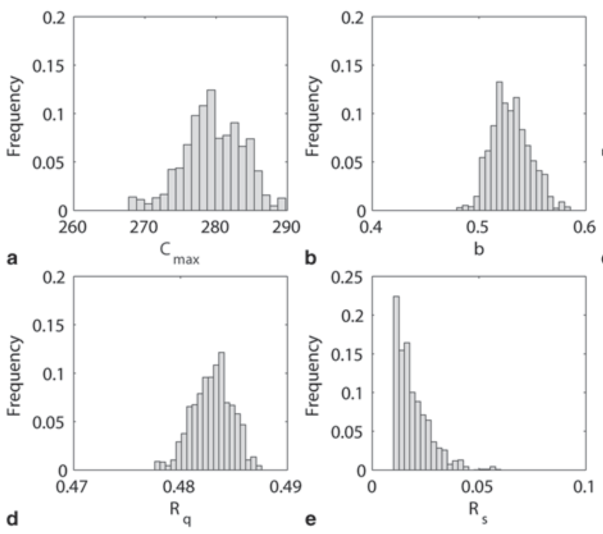


Fig. 4.7 Posterior distributions of the HyMOD parameters obtained using MCMC sampling, **a** $C_{\max}$, **b** b, **c** α, **d** R_q, **e** R_s

Figure 4.7a–e shows the resulting (marginal) posterior distributions of the HyMOD parameters. The shapes of the PDFs for $C_{\max}$, b, and R_q are Gaussian like, whereas those for α and R_s are more skewed.

4.3.4 Simulated Annealing

Previously, we showed how MCMC can be used to sample a complex shaped PDF. For the same token, the MCMC can also be used to find the global optimum of a complex multimodal function by forming a chain that gradually approaches the global optimum. This is the basic principle behind simulated annealing (SA), a class of stochastic global optimization methods.

In general, stochastic global optimization methods proceed in a similar way as many of the deterministic global optimization methods do (see Chap. 3.6). Given an optimization problem,

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}), \ \boldsymbol{\theta} \in P_{ad} \text{ subject to constraints,}$$

stochastic global optimization methods explore many points in the feasible space in the hope that the function value approaches the global minimum as the number of trials increases. The main difference is that in stochastic optimization the points are chosen according to some random rule. For any point $\boldsymbol{\theta}_i$, the objective function $f(\boldsymbol{\theta}_i)$ is evaluated and the transition to the next point can either depend on the current state or be purely random. Like in MCMC methods, the efficiency of the stochastic global optimization methods relies on the rules they use to search the parameter space. In addition, the efficiency of the global optimization methods depends on how the promising solutions survive different iterations and how the algorithms avoid trapping at local minima.

SA is closely related to the Metropolis–Hastings algorithm. As its name implies, SA is inspired by the annealing process in metallurgy. The main idea is to "melt" the system being optimized at a high temperature and then gradually lower the "energy" of the system following a cooling schedule until the system reaches a minimum energy state and no further changes occur. At each temperature, the simulation must proceed long enough to reach equilibrium before the temperature is lowered again. This is because like in metal annealing, the cooling should not happen too fast so that the system is trapped at local minima.

The standard SA includes several basic elements: the initial guess, a cooling schedule, an acceptance criterion, and a scheme for transitioning to different state or for visiting different neighbors of the current solutions. More specifically, the standard SA includes the following steps:

1. Start from a high temperature, T_0, and an initial guess, $\boldsymbol{\theta}_0$, and evaluate the value of user-specified objective function, $f(\boldsymbol{\theta}_0)$.
2. At a temperature T_n, generate a new point, $\boldsymbol{\theta}_{i+1}$, from the current point $\boldsymbol{\theta}_i$:
   - First generate a candidate point $\boldsymbol{\theta}'$ based on the current point $\boldsymbol{\theta}_i$ using transition function $q(\boldsymbol{\theta}_i \to \boldsymbol{\theta}')$. This is essentially the same as what the proposal distribution does in MCMC, through which a random variate is obtained
   - Evaluate the change in function value when migrating from $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}'$, $\Delta = f(\boldsymbol{\theta}') - f(\boldsymbol{\theta}_i)$. The new point $\boldsymbol{\theta}'$ is accepted with a probability α according to

$$\alpha = \begin{cases} 1, & \text{if } \Delta \le 0 \\ \exp(-\Delta / T_n), & \text{if } \Delta > 0 \end{cases}, \tag{4.3.28}$$

     where T_n is the current temperature. Equation (4.3.28) says that if the new state has a lower objective function value, accept it unconditionally; otherwise, the new state is accepted probabilistically and the probability is analogous to the metal's energy state when reaching equilibrium at a temperature during annealing. Therefore, an uphill climbing move (i.e., positive Δ) is more likely to be accepted at higher temperatures, but as the temperature approaches zero, most uphill moves will be rejected. The above equation can also be compared to the Metropolis–Hastings ratio given in (4.3.9).

   - If $\boldsymbol{\theta}'$ is accepted, let $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}'$; if $\boldsymbol{\theta}'$ is rejected, let $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i$.
3. Iterate step $2L$ times (e.g., $L = 10$) without changing the temperature. This is aimed to avoid trapping at a local equilibrium.
4. Reduce the temperature to T_{n+1}, which is usually done according to a user-specified cooling schedule. For example, a simple cooling schedule can be

$$T_{n+1} = \lambda T_n, \ 0 < \lambda < 1.$$

Steps 2–4 in the above are repeated until a certain stopping criterion is satisfied, such as when the minimum temperature is reached or when the objective function is below a user-specified tolerance. In the above SA algorithm, tunable parameters include number of iteration times (L) at each temperature and cooling rate λ.

Like genetic algorithms (GA) introduced in Chap. 3, SA is also derived based on analogy to a natural process. Unlike GA which evolves generations of individuals, SA evolves a single chain. If the transition PDF is symmetric and is bounded away from zero (i.e., $q(\boldsymbol{\theta}_i \to \boldsymbol{\theta}') > c$, where c is a positive real number), it can be shown that SA converges to a global minimizer (Zhigljavsky 2007).

In the literature, SA has been applied to solve both SOO and MOO problems. Suman and Kumar (2005) provided a review of different SA algorithms. In EWR, SA has been used to find contaminant source locations (Jha and Datta 2012; Sun and Nicot 2012).

4.4 Review Question

1. Compare the deterministic inverse problem to the statistical inverse problem in all of the following aspects: (a) concept, (b) well-posedness, (c) data used, (d) results of solution, (e) methods of solution, and (f) uncertainty of solution.
2. Assume the prior distribution of the unknown parameter θ is a uniform distribution over interval $[0,1]$. After estimation, its posterior distribution is given by $p(\theta) = 2,$ when $0 \le \theta < 1/4$; and $p(\theta) = 2/3,$ when $1/4 \le \theta < 1$. Calculate the information content transferred from data to the estimated parameter.
3. Discuss the statement, "The statistical inversion method is superior to the deterministic inversion method."
4. Discuss the statement, "The statistical inversion method cannot be used if we do not have a prior distribution and/or we do not know the distribution of observation error."
5. Discuss the statement, "When the probability distribution of observation error is inaccurate, the estimated parameters must be inaccurate too. In this case, the deterministic inversion method gives more accurate results."
6. Derive Eq. (4.2.19) and Eq. (4.2.20), find the Woodbury identity, and obtain Eq. (4.2.22) and Eq. (4.2.21).
7. Formulate the statistical inverse problem for two-state coupled models.
8. Compared with point estimators, what kinds of result can be obtained from sampling the posterior distribution by an MCMC solver?
9. Use an MCMC code, such as the one mentioned in Sect. 4.3.2, to resolve the Review Question 9 of Chap. 2.

Chapter 5
Model Differentiation

In the previous chapters, we showed that an inverse problem ultimately becomes an optimization problem, regardless the type of framework (deterministic or statistical) used to formulate it. Gradient-based algorithms are efficient for solving optimization problems, but require derivatives of the objective function $S(\boldsymbol{\theta})$ as inputs. In this chapter, we will consider methods for obtaining derivatives of a generic function defined by a model or by a computer code.

Section 5.1 introduces the *perturbation method*, which uses the first-order finite difference to approximate derivatives and is the simplest method for model differentiation. However, more accurate and effective alternatives to the perturbation method are available. In Sect. 5.2, we will introduce the *sensitivity equation method*, in which model derivatives are the solution of a system of sensitivity equations obtained by differentiating the governing equations of the model. With the same computational effort, this method can produce more accurate results than that produced by the simpler perturbation method. Nevertheless, additional derivation and programming are required.

Section 5.3 introduces *the adjoint-state method*, in which an adjoint problem is derived from the original forward problem. Although this method typically incurs more derivation and programming overhead, its computational effectiveness is attractive. By solving the adjoint problem and the original forward problem once, all components of the gradient vector $\nabla S(\boldsymbol{\theta})$ can be obtained, regardless of the dimension of the unknown parameter vector $\boldsymbol{\theta}$.

Section 5.4 gives a short introduction to the method of *automatic differentiation* (AD). As its name suggests, AD attempts to differentiate an arbitrary numerical code with input $\boldsymbol{\theta}$ and output $S(\boldsymbol{\theta})$ to generate a new code with input $\boldsymbol{\theta}$ and output $\nabla S(\boldsymbol{\theta})$. AD has two modes: a forward mode that "differentiates" a code from top to bottom and a reverse mode that "differentiates" a code from bottom to top. The two modes correspond to the sensitivity equation method and the adjoint-state method, respectively. With the advent of AD, model differentiation could eventually become straightforward. Unfortunately, the currently available AD tools, especially those based on the reverse mode, have not reached full automation for complicated models.

N-Z. Sun, A. Sun, *Model Calibration and Parameter Estimation,*
DOI 10.1007/978-1-4939-2323-6_5

Section 5.5 shows some basic applications of model differentiation in inverse modeling and local sensitivity analysis. As we shall see later in this book, model differentiation plays an important role not only for numerical optimization but also for sensitivity analysis, data assimilation, reliability assessment, and experimental design.

5.1 Perturbation Method

Whenever a gradient-based optimization algorithm is used for inversion, we need to calculate the gradient vector of the objective function $S(\boldsymbol{\theta})$

$$\mathbf{g}(\boldsymbol{\theta}) = \nabla S(\boldsymbol{\theta}) = \left(\frac{\partial S}{\partial \theta_1}, \frac{\partial S}{\partial \theta_2}, \cdots, \frac{\partial S}{\partial \theta_m}\right)^T, \tag{5.1.1}$$

where $\boldsymbol{\theta} \in P_{ad}$ is an m-dimensional input parameter vector, $\theta = (\theta_1, \theta_2, \cdots, \theta_m)^T$. For example, the bicriterion objective function given in Sect. 3.2 is

$$S(\boldsymbol{\theta}) = \left\|\mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{u}_D^{obs}\right\|_2^2 + \alpha \left\|\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathbf{0}}\right\|_2^2, \tag{5.1.2}$$

in which $\mathbf{u}_D^{obs}$ is an n-dimensional observation vector, $\mathbf{u}_D(\boldsymbol{\theta})$ is the corresponding model outputs, α is weight, and $\boldsymbol{\theta}_0$ is prior guess of the unknown $\boldsymbol{\theta}$. When the Gauss–Newton method is used to solve for the optimization problem, the following $n \times m$ Jacobian matrix needs to be calculated

$$\mathbf{J}_D(\boldsymbol{\theta}) = \frac{\partial \mathbf{u}_D}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial u_{D,1}}{\partial \theta_1} & \frac{\partial u_{D,1}}{\partial \theta_2} & \cdots & \frac{\partial u_{D,1}}{\partial \theta_m} \\ \frac{\partial u_{D,2}}{\partial \theta_1} & \frac{\partial u_{D,2}}{\partial \theta_2} & \cdots & \frac{\partial u_{D,2}}{\partial \theta_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial u_{D,n}}{\partial \theta_1} & \frac{\partial u_{D,n}}{\partial \theta_2} & \cdots & \frac{\partial u_{D,n}}{\partial \theta_m} \end{bmatrix}, \tag{5.1.3}$$

in which $\mathbf{J}_D$ is also called the *sensitivity matrix* of model outputs $\mathbf{u}_D$ to input parameters, and its elements are called *sensitivity coefficients*. When the forward solution is obtained by a numerical model, closed-form expressions of $\mathbf{u}_D(\boldsymbol{\theta})$ are not available for differentiation; instead, we need to evaluate $S(\boldsymbol{\theta})$ numerically by running the forward model.

The simplest method for calculating the gradient $\nabla S(\boldsymbol{\theta})$ is the perturbation method, which evaluates derivatives $\partial S / \partial \theta_j$ by taking the first-order finite difference approximation of $S(\boldsymbol{\theta})$ with respect to θ_j,

$$\frac{\partial S}{\partial \theta_j} \approx \frac{S(\boldsymbol{\theta} + \Delta_j \mathbf{e}_j) - S(\boldsymbol{\theta})}{\Delta_j}, \ (j = 1, 2, \cdots, m), \tag{5.1.4}$$

where $\mathbf{e}_j$ is a unit vector along the j-th coordinate direction and Δ_j is an increment or magnitude of perturbation applied to θ_j. To calculate the gradient $S(\boldsymbol{\theta})$ using the perturbation method, we need to run the forward solution code $m+1$ times, in which an extra run is needed for calculating $S(\boldsymbol{\theta})$. With the perturbation method, all elements of the sensitivity matrix $\mathbf{J}_D(\boldsymbol{\theta})$ can be evaluated numerically by

$$\frac{\partial u_{D,i}}{\partial \theta_j} \approx \frac{u_{D,i}(\boldsymbol{\theta} + \Delta_j \mathbf{e}_j) - u_{D,i}(\boldsymbol{\theta})}{\Delta_j}, (i = 1, 2, \cdots, n; \ j = 1, 2, \cdots, m). \tag{5.1.5}$$

A major difficulty related to the perturbation method is the determination of an appropriate increment Δ_j to be used in (5.1.4) or (5.1.5). Too large an increment may produce inaccurate results because of the truncation error, while a too small increment may produce unstable results because of the computational error of the forward solution. A rule of thumb is to set

$$\left|\Delta_j\right| = \beta \left|\theta_j\right|, \ \beta = 1\% \sim 10\%$$

such that the difference $\left|S(\boldsymbol{\theta} + \mathit{\Delta}_j \mathbf{e}_j) - S(\boldsymbol{\theta})\right|$ is significantly larger than the computational error of the forward solution. Thus, the perturbation method has certain subjectivity in it which may cause inaccuracy. In the rest of this chapter, we will introduce more accurate methods for obtaining derivatives.

5.2 Sensitivity Equation Method

5.2.1 Sensitivity Equation and Its Solution

Let us derive the sensitivity equation for the following general model equation

$$\boldsymbol{\mathcal{L}}(\mathbf{u}, \boldsymbol{\theta}) = \mathbf{0}. \tag{5.2.1}$$

For a single-state model, (5.2.1) consists of a single equation and the model state variable u is a scalar function. For a multistate model, (5.2.1) consists of a set of equations, where $\mathbf{u} = (u_1, u_2, \ldots, u_k)^T$ is a *k*-dimensional vector function, $\boldsymbol{\mathcal{L}} = (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k)^T$ is a vector operator, and $\boldsymbol{\theta}$ is an $m \times 1$ vector with components $(\theta_1, \theta_2, \cdots, \theta_m)^T$. Differentiating (5.2.1) with respect to θ_j yields

$$\nabla_{\mathbf{u}} \boldsymbol{\mathcal{L}} \frac{\partial \mathbf{u}}{\partial \theta_j} + \nabla_{\boldsymbol{\theta}} \boldsymbol{\mathcal{L}} \frac{\partial \boldsymbol{\theta}}{\partial \theta_j} = \mathbf{0}, \; (j = 1, 2, \cdots, m). \tag{5.2.2}$$

Equation (5.2.2) is called the *sensitivity equation* of model equation (5.2.1) because its solution yields the sensitivity coefficients. In (5.2.2), operators $\nabla_{\mathbf{u}} \boldsymbol{\mathcal{L}}$ and $\nabla_{\boldsymbol{\theta}} \boldsymbol{\mathcal{L}}$ are defined respectively by

$$\nabla_{\mathbf{u}} \boldsymbol{\mathcal{L}} = \left[\frac{\partial \mathcal{L}_i}{\partial u_j} \right]_{k \times k} \text{ and } \nabla_{\boldsymbol{\theta}} \boldsymbol{\mathcal{L}} = \left[\frac{\partial \mathcal{L}_i}{\partial \theta_j} \right]_{k \times m}. \tag{5.2.3}$$

Operators $\nabla_{\mathbf{u}} \boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{L}}$ are the same when $\boldsymbol{\mathcal{L}}$ is linear or similar (see the examples below). The term $\nabla_{\boldsymbol{\theta}} \boldsymbol{\mathcal{L}}(\partial \boldsymbol{\theta} / \partial \theta_j)$ is known after the primary problem is solved, and $\partial \boldsymbol{\theta} / \partial \theta_j$ has components $\partial \theta_i / \partial \theta_j = 0$ when $i \neq j$ and $\partial \theta_i / \partial \theta_j = 1$ when $i = j$. The solution of sensitivity equations (5.2.2) gives distributed derivatives $[\partial u_i / \partial \theta_j]$ for all state variables with respect to all components of the input parameter, using which the gradient $\nabla S(\boldsymbol{\theta})$ and Jacobian $\mathbf{J}_D(\boldsymbol{\theta})$ can be calculated.

Applying the sensitivity method to calculate the Jacobian has two major advantages: (1) the total computational effort is independent of the number of observations and (2) the method circumvents the difficulty of determining the perturbation increments as required by the perturbation method.

Example 5.1 *Derivation of the sensitivity equation for a linear model*
Assuming the model equation is given by a linear system

$$\mathbf{A}\mathbf{u} = \mathbf{b}, \tag{5.2.4}$$

where the response matrix $\mathbf{A}$ and the right-hand-side data vector $\mathbf{b}$ may depend on the parameter vector $\boldsymbol{\theta}$. Differentiating (5.2.4) with respect to component θ_j of $\boldsymbol{\theta}$, we obtain the following sensitivity equation

$$\mathbf{A}\mathbf{u}'_j = \mathbf{b}'_j, \; (j = 1, 2, \ldots, m), \tag{5.2.5}$$

where $\mathbf{u}'_j = \partial \mathbf{u} / \partial \theta_j$ and $\mathbf{b}'_j = (\partial \mathbf{b} / \partial \theta_j) - (\partial \mathbf{A} / \partial \theta_j)\mathbf{u}$. Equation (5.2.5) implies that the same code used for solving the model equation (5.2.4) can also be used to solve the sensitivity equation (5.2.5) after modifying the right-hand-side term appropriately. The solution process consists of two major steps:

1. Solve the primary problem (5.2.4) using a selected linear system solver.
2. For $j = 1, 2, \cdots, m$
   - Calculate $\mathbf{b}'_j$,
   - Use $\mathbf{b}'_j$ as the right-hand-side term of (5.2.5), and
   - Solve the equation to obtain $\mathbf{u}'_j$.

The gradient $\nabla S(\boldsymbol{\theta})$ or Jacobian $\mathbf{J}_D(\boldsymbol{\theta})$ can then be assembled using $\mathbf{u}'_j$.

Example 5.2 *Derivation of the sensitivity equation for an* ODE
Assuming the modeling equation is a second-order ODE given by

$$a\frac{d^2u}{dt^2} + b\frac{du}{dt} + cu = d \tag{5.2.6}$$

subject to initial conditions

$$u(x,t)\Big|_{t=0} = f_0 \text{ and } \frac{du}{dt}\Big|_{t=0} = f_1,$$

where coefficients a, b, c, and the right-hand-side term d may depend on the parameter vector $\boldsymbol{\theta}$ and so does model state u when the model is nonlinear. Differentiation of (5.2.6) with respect to component θ_j of $\boldsymbol{\theta}$ yields

$$a\frac{d^2u'_j}{dt^2} + b\frac{du'_j}{dt} + cu'_j = d'_j, \tag{5.2.7}$$

subject to

$$u'_j(x,t)\Big|_{t=0} = 0 \text{ and } \frac{du'_j}{dt}\Big|_{t=0} = 0,$$

where $u'_j = \partial u / \partial \theta_j$ and

$$d'_j = \frac{\partial d}{\partial \theta_j} - \frac{\partial a}{\partial \theta_j}\frac{d^2u}{dt^2} - \frac{\partial b}{\partial \theta_j}\frac{du}{dt} - \frac{\partial c}{\partial \theta_j}u. \tag{5.2.8}$$

Again, the sensitivity equation (5.2.7) and model equation (5.2.6) have the same form. Thus, they can be solved using the same code after replacing d with d'_j and setting f_0 and f_1 to zero. Solving (5.2.7) for all $j = 1, 2, \cdots, m$, the gradient $\nabla S(\boldsymbol{\theta})$ or Jacobian $\mathbf{J}_D(\boldsymbol{\theta})$ can then be obtained for any number of observations.

Example 5.3 *Derivation of the sensitivity equation for a partial differential equation (PDE)*
Consider a model governed by the following PDE

$$\begin{aligned}\frac{\partial(au)}{\partial t} &= \frac{\partial}{\partial x}\left(b_{11}\frac{\partial u}{\partial x} + b_{12}\frac{\partial u}{\partial y} - c_{11}u\right)\\ &\quad+\frac{\partial}{\partial y}\left(b_{21}\frac{\partial u}{\partial x} + b_{22}\frac{\partial u}{\partial y} - c_{22}u\right), \qquad (x,y)\in\boldsymbol{\Omega},\ t\geq 0 \\ &\quad+du+e\end{aligned} \tag{5.2.9}$$

subject to initial and boundary conditions

$$\begin{aligned}&u\,|_{t=0} = f_0,\\ &u\,|_{\Gamma_1} = f_1,\\ &-\left[\left(b_{11}\frac{\partial u}{\partial x} + b_{12}\frac{\partial u}{\partial y} - c_{11}u\right)n_x\right.\\ &\quad\left.+\frac{\partial}{\partial y}\left(b_{21}\frac{\partial u}{\partial x} + b_{22}\frac{\partial u}{\partial y} - c_{22}u\right)n_y\right]_{\Gamma_2} = f_2,\end{aligned} \tag{5.2.10}$$

where f_0, f_1, and f_2 are prescribed functions, Γ_1 and Γ_2 consist of boundaries of the domain $\boldsymbol{\Omega}$, and (n_x, n_y) is the normal direction of Γ_2. All coefficients of the PDE may depend on parameter $\boldsymbol{\theta}$ and also on the state variable *u* if the model is nonlinear. Note that (5.2.9) (or its three-dimensional counterpart) can be used to model many flow and mass/heat transport processes typically encountered in the EWR modeling (see examples in Chap. 1).

Differentiation of (5.2.9) and (5.2.10) with respect to component θ_j of $\boldsymbol{\theta}$ yields

$$\begin{aligned}\frac{\partial(au'_j)}{\partial t} &= \frac{\partial}{\partial x}\left(b_{11}\frac{\partial u'_j}{\partial x} + b_{12}\frac{\partial u'_j}{\partial y} - c_{11}u'_j\right)\\ &\quad+\frac{\partial}{\partial y}\left(b_{21}\frac{\partial u'_j}{\partial x} + b_{22}\frac{\partial u'_j}{\partial y} - c_{22}u'_j\right)\\ &\quad+du'_j + e'_j,\\ &\quad(x,y)\in\boldsymbol{\Omega},\ t\geq 0\end{aligned} \tag{5.2.11}$$

and

$$\begin{aligned}
&u'_j\,|_{t=0} = 0,\\
&u'_j\,|_{\Gamma_1} = 0,\\
&-\left[\left(b_{11}\frac{\partial u'_j}{\partial x} + b_{12}\frac{\partial u'_j}{\partial y} - c_{11}u'_j\right)n_x\right.\\
&\left.+\frac{\partial}{\partial y}\left(b_{21}\frac{\partial u'_j}{\partial x} + b_{22}\frac{\partial u'_j}{\partial y} - c_{22}u'_j\right)n_y\right]_{\Gamma_2} = f'_{2,j},
\end{aligned} \tag{5.2.12}$$

where $u'_j = \partial u / \partial \theta_j$ and

$$\begin{aligned}
e'_j =& \frac{\partial}{\partial x}\left(\frac{\partial b_{11}}{\partial \theta_j}\frac{\partial u}{\partial x} + \frac{\partial b_{12}}{\partial \theta_j}\frac{\partial u}{\partial y} - \frac{\partial c_{11}}{\partial \theta_j}u\right)\\
&+\frac{\partial}{\partial y}\left(\frac{\partial b_{21}}{\partial \theta_j}\frac{\partial u}{\partial x} + \frac{\partial b_{22}}{\partial \theta_j}\frac{\partial u}{\partial y} - \frac{\partial c_{22}}{\partial \theta_j}u\right)\\
&+\frac{\partial d}{\partial \theta_j}u + \frac{\partial e}{\partial \theta_j} - \frac{\partial}{\partial t}\left(\frac{\partial a}{\partial \theta_j}u\right),
\end{aligned} \tag{5.2.13}$$

$$\begin{aligned}
f'_{2,j} =& \left[\left(\frac{\partial b_{11}}{\partial \theta_j}\frac{\partial u}{\partial x} + \frac{\partial b_{12}}{\partial \theta_j}\frac{\partial u}{\partial y} - \frac{\partial c_{11}}{\partial \theta_j}u\right)n_x\right.\\
&\left.+\left(\frac{\partial b_{21}}{\partial \theta_j}\frac{\partial u}{\partial x} + \frac{\partial b_{22}}{\partial \theta_j}\frac{\partial u}{\partial y} - \frac{\partial c_{22}}{\partial \theta_j}u\right)n_y\right]_{\Gamma_2}.
\end{aligned} \tag{5.2.14}$$

After solving the primary problem defined by (5.2.9) and (5.2.10), we can calculate e'_j and $f'_{2,j}$, substitute them into sensitivity equations (5.2.11) and (5.2.12), and then solve the sensitivity problem by using the same code to obtain $u'_j = \partial u / \partial \theta_j$.

Example 5.4 *Derivation of the sensitivity equation for a multistate model*
Let us consider a classical leaky aquifer problem in hydrogeology, in which an unconfined aquifer is separated from the underlying confined aquifer by a leaky confining layer. Groundwater flow in such a leaky aquifer system is governed by the following two coupled PDEs (Bear 1979):

$$\begin{cases}
S_y\dfrac{\partial h}{\partial t} - \nabla\cdot(Kh\nabla h) - (h-\phi)/b_l = \lambda\\
S\dfrac{\partial \phi}{\partial t} - \nabla\cdot(T\nabla \phi) + (h-\phi)/b_l = -Q
\end{cases}, \tag{5.2.15}$$

where h is the water level in the unconfined aquifer; ϕ is the head of the confined aquifer; S_y and S are the specific yield and storativity, respectively; K and T are hydraulic conductivity and transmissivity; b_l is the leakage coefficient; λ is the infiltration rate; and Q is the pumping rate. The model in (5.2.15) consists of two state variables $\mathbf{u} = (h, \phi)$ and six parameters $\boldsymbol{\theta} = (S_y, S, K, T, b_l, \lambda)^T$. For simplification, we assume that these parameters are constant. Note that the first equation in (5.2.15) is nonlinear. To find distributed derivatives $h'_\lambda = \partial h / \partial \lambda$ and $\phi'_\lambda = \partial \phi / \partial \lambda$, we differentiate the primary model equations with respect to λ, which yields

$$\begin{cases} S_y \dfrac{\partial h'_\lambda}{\partial t} - \nabla\cdot(Kh\nabla h'_\lambda) - \nabla\cdot(Kh'_\lambda \nabla h) - (h'_\lambda - \phi'_\lambda)/b_l = 1 \\ S\dfrac{\partial \phi'_\lambda}{\partial t} - \nabla\cdot(T\nabla\phi'_\lambda) + (h'_\lambda - \phi'_\lambda)/b_l = 0 \end{cases}. \tag{5.2.16}$$

The model equation (5.2.15) and its sensitivity equation (5.2.16) have almost identical forms, except that an extra divergence term now appears in the first equation of (5.2.16) because of its nonlinearity. After solving the state variables h and ϕ from the primary model equation, we can substitute them into the sensitivity equation (5.2.16) and solve for both h'_λ and ϕ'_λ from the equation. The derivatives of h and ϕ with respect to other parameters can be obtained similarly.

5.2.2 Discrete Form

When a numerical method is used to solve a time-independent PDE, the forward solution can be obtained by solving a set of linear equations after spatial discretization. For nonlinear problems, both linearization and iteration will be needed. As shown in Example 5.1, the sensitivity equation of a linear model can be solved by the same numerical code after changing the right-hand-side terms. For a time-dependent PDE (such as the one shown in Example 5.3), the primary PDE is reduced to a set of algebraic equations after discretizing both temporal and spatial variables. We need to solve this system of algebraic equations when propagating the model states from time t_{k-1} to t_k (see also Example 1.10 in Chap. 1)

$$\mathbf{H}_k \mathbf{u}_k = \mathbf{b}_k. \tag{5.2.17}$$

In (5.2.17), $\mathbf{H}_k$ is an $N \times N$ coefficient matrix that may depend on parameter $\boldsymbol{\theta}$, N is the number of nodal points resulting from the spatial discretization of the state variable, $\mathbf{u}_k$; and the right-hand-side term $\mathbf{b}_k$ depends on the solution $\mathbf{u}_{k-1}$ from the previous time step, the forcing term, and the boundary conditions. Differentiation of (5.2.17) with respect to θ_j yields the following sensitivity equations

$$\mathbf{H}_k \mathbf{u}'_{k,j} = \mathbf{b}'_{k,j}, \quad (j = 1, 2, \cdots, m), \tag{5.2.18}$$

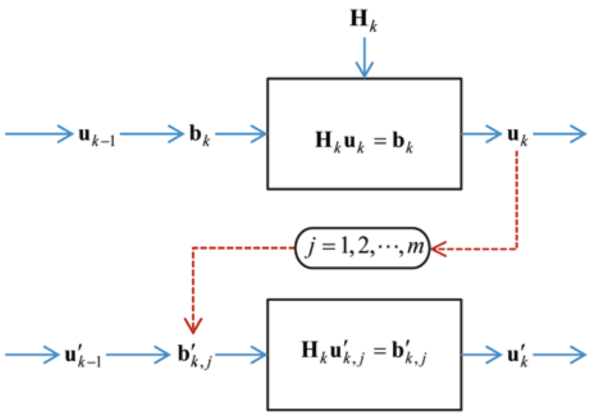


Fig. 5.1 Flow diagram for solving a discretized time-dependent sensitivity equation, where *k* and *k*-1 are time step indices, *j* is parameter index, and *m* is dimension of $\boldsymbol{\theta}$

where $\mathbf{u}'_{k,j} = \partial\mathbf{u}_k / \partial\theta_j$ and $\mathbf{b}'_{k,j} = (\partial\mathbf{b}_k / \partial\theta_j) - (\partial\mathbf{H}_k / \partial\theta_j)\mathbf{u}_k$.

The code for solving the sensitivity equations (5.2.18) can be developed by making minor modifications to the original code for solving the primary problem. At the *k*th time step, the following statements are executed after the forward solution $\mathbf{u}_k$ is calculated (see also Fig. 5.1):

For $j = 1, 2, \cdots, m$, do

1. Calculate $\mathbf{b}'_{k,j}$,
2. Use it to replace $\mathbf{b}_k$, and
3. Use the same solver to obtain all derivatives $\mathbf{u}'_{k,j}$.

Note that in the above process, we do not need to change $\mathbf{H}_k$, neither do we need to calculate e'_j and $f'_{2,j}$ as we have done in (5.2.13) and (5.2.14). Therefore, the discrete form of the sensitivity equation method is easier to implement than its continuous-form counterpart.

5.3 The Adjoint-State Method

5.3.1 Single-State Models

In this section, we introduce the adjoint-state method for evaluating the gradient and Jacobian, which is more effective than the sensitivity equation method when the dimension of parameter is high. For ease of understanding, we start off by considering a general single-state model represented by

$$\mathcal{L}(u, \boldsymbol{\theta}) = 0, \tag{5.3.1}$$

where the state variable $u = u(\boldsymbol{\theta})$ is defined in terms of spatial variables $\mathbf{x} \in \Omega$ and/or temporal variables $t \in [0, t_f]$, and $\boldsymbol{\theta}$ is the unknown parameter vector or functions. We assume that both the state space and parameter space are Hilbert space ($\mathbb{H}$) for which the inner product is defined by (see Appendix A)

$$(w, v) = \int_R wv dR \text{ for any } w, v \in \mathbb{H}, \tag{5.3.2}$$

where $R \doteq [0, t_f] \times \Omega$ denotes the spatial and temporal domain for integration. In mathematics, it can be shown that for any continuous linear operator $\mathcal{A} : \mathbb{H} \to \mathbb{H}$, there is an operator $\mathcal{A}^*$, called the *adjoint operator* of $\mathcal{A}$ (see Appendix A), such that

$$(\mathcal{A}w, v) = (w, \mathcal{A}^* v) \quad \text{for any } w, v \in \mathbb{H}. \tag{5.3.3}$$

The first step of the adjoint-state method is to define a *performance function* in the following general form

$$E(u, \boldsymbol{\theta}) = \int_R f(u, \boldsymbol{\theta}) dR, \tag{5.3.4}$$

where $f(u, \boldsymbol{\theta})$ is a function to be chosen by the user and examples of which will be given later in this section.

The second step is to derive the variational problem of the original model. Taking the first-order variation of the performance function (5.3.4), we have

$$\delta E = \int_R \left(\frac{\partial f}{\partial u} \delta u + \frac{\partial f}{\partial \boldsymbol{\theta}} \delta \boldsymbol{\theta} \right) dR. \tag{5.3.5}$$

Note that the two variations $\delta\boldsymbol{\theta}$ and δu in the above equation may be functions of both space and time. Taking the first-order variation of model equation (5.3.1), we have

$$\nabla_u \mathcal{L} \delta u + \nabla_{\boldsymbol{\theta}} \mathcal{L} \delta \boldsymbol{\theta} = 0, \tag{5.3.6}$$

where $\nabla_u \mathcal{L} = \partial \mathcal{L} / \partial u$ and $\nabla_{\boldsymbol{\theta}} \mathcal{L} = \partial \mathcal{L} / \partial \boldsymbol{\theta}$ are gradient operators. Equation (5.3.6) is known as the variation equation of model equation (5.3.1). It gives the relationship between $\delta\boldsymbol{\theta}$ and δu.

The third step is to use (5.3.6) to eliminate δu from (5.3.5). This is the key step of the adjoint-state method. Multiplying (5.3.6) by a differentiable function $\psi(\mathbf{x}, t)$, integrating the equation over R, and letting $\mathcal{A} \doteq \nabla_u \mathcal{L}$, $w \doteq \delta u$, $v \doteq \psi$, and $\mathcal{A} \doteq \nabla_{\boldsymbol{\theta}} \mathcal{L}$, $w \doteq \delta\boldsymbol{\theta}$, $v \doteq \psi$, respectively, in (5.3.3), we obtain

$$\int_R [\delta u \nabla_u^* \mathcal{L} \psi + \delta \boldsymbol{\theta} \nabla_{\boldsymbol{\theta}}^* \mathcal{L} \psi] dR = 0, \tag{5.3.7}$$

where the adjoint operators $(\nabla_u \mathcal{L})^*$ and $(\nabla_{\boldsymbol{\theta}} \mathcal{L})^*$ are denoted by $\nabla_u^* \mathcal{L}$ and $\nabla_{\boldsymbol{\theta}}^* \mathcal{L}$, respectively. The summation of (5.3.5) and (5.3.7) yields

$$\delta E = \int_R \left\{ \left[\frac{\partial f}{\partial u} + \nabla_u^* \mathcal{L} \psi \right] \delta u + \left[\frac{\partial f}{\partial \boldsymbol{\theta}} + \nabla_{\boldsymbol{\theta}}^* \mathcal{L} \psi \right] \delta \boldsymbol{\theta} \right\} dR. \tag{5.3.8}$$

The term associated with δu on the right-hand-side of the above equation can be eliminated if ψ is the solution of the following equation

$$\frac{\partial f}{\partial u} + \nabla_u^* \mathcal{L} \psi = 0. \tag{5.3.9}$$

Equation (5.3.9) is called the adjoint equation of the primary model, and its solution ψ is called the adjoint state. Depending on the type of the equation, certain subsidiary conditions may be needed to solve this equation (see examples given later in this section). After using the adjoint equation (5.3.9) to eliminate the term associated with δu in (5.3.8), we obtain the following variational identity.

$$\delta E = \int_R \left[\frac{\partial f}{\partial \boldsymbol{\theta}} + \nabla_{\boldsymbol{\theta}}^* \mathcal{L} \psi \right] \delta \boldsymbol{\theta} \, dR. \tag{5.3.10}$$

The fourth step is to calculate partial derivatives of the performance function. For each component θ_j, let $\delta \boldsymbol{\theta} = \left\{ 0, \cdots, 0, \Delta \theta_j, 0, \cdots, 0 \right\}^T$, we have

$$\frac{\partial E}{\partial \theta_j} = \int_R \left[\frac{\partial f}{\partial \theta_j} + \nabla_{\theta_j}^* \mathcal{L} \psi \right] dR, \; (j = 1, 2, \cdots, m). \tag{5.3.11}$$

The above four steps describe the adjoint-state method. The selection of the performance function $E(u, \boldsymbol{\theta})$ in the first step is problem dependent. For example, if we choose $f(u, \boldsymbol{\theta})$ in (5.3.4) to be

$$f(u, \boldsymbol{\theta}; \mathbf{x}, t) = \sum_{i=1}^{n} w_i^2 [u(\boldsymbol{\theta}; \mathbf{x}, t) - \mathbf{u}^{\mathbf{obs}}(\mathbf{x}, t)]^2 \delta(\mathbf{x} - \mathbf{x}_i) \delta(t - t_i), \tag{5.3.12}$$

the performance function $E(u, \boldsymbol{\theta})$ then becomes the weighted least squares misfit function,

$$E(\boldsymbol{\theta}) = S(\boldsymbol{\theta}) = \sum_{i=1}^{n} w_i^2 [u_{D,i}(\boldsymbol{\theta}) - u_{D,i}^{obs}]^2. \tag{5.3.13}$$

In (5.3.13), $\{u_{D,i}(\boldsymbol{\theta})\}_{i=1}^{n}$ is a set of model predictions corresponding to the observation locations and times $\{(\mathbf{x}_i, t_i)\}_{i=1}^{n}$ of the observation design, $\delta(\cdot)$ is the Dirac delta function, and $\{w_i\}_{i=1}^{n}$ are weights. Therefore, when a gradient-based optimization method is used for inversion, (5.3.11) can be used to calculate all components of $\nabla S(\boldsymbol{\theta})$. When a Gauss–Newton method is used for inversion, we need to calculate the Jacobian (5.1.3). In this case, we can choose $f(u,\boldsymbol{\theta})$ in (5.3.4) as

$$f(u,\boldsymbol{\theta};\mathbf{x},t) = \mathbf{u}(\boldsymbol{\theta};\mathbf{x},t)\delta(\mathbf{x}-\mathbf{x}_i)\delta(t-t_i), \tag{5.3.14}$$

and the performance function $E(u,\boldsymbol{\theta})$ becomes

$$E(\boldsymbol{\theta}) = \mathbf{u}(\boldsymbol{\theta};\mathrm{x}_i,\mathrm{t}_i) = u_{D,i}(\boldsymbol{\theta}). \tag{5.3.15}$$

Equation (5.3.11) can now be used to calculate $\partial u_{D,i} / \partial \theta_j$ for $j = 1,2,\cdots,m$. To obtain the Jacobian, this calculation is repeated for all $u_{D,i}(\boldsymbol{\theta})$ ($i = 1,2,\cdots,n$).

As an effective model differentiation tool, the adjoint-state method plays an important role in sensitivity analysis and experimental design when the number of parameters is large. In what follows, we will describe procedures for finding the adjoint operators, deriving the adjoint equation, finding the subsidiary conditions, and solving the adjoint equation to obtain the adjoint state.

5.3.2 Rules of Adjoint Operation

In this section, we introduce rules and formulae for deriving adjoint equations. Let us start with a simple example of applying the adjoint-state method.

Example 5.5 *The adjoint-state method for a lumped parameter model*
Assume that the model equation is given by

$$\frac{du}{dt} + a(u,\boldsymbol{\theta}) = 0,\ 0 \le t \le t_f;\ \text{s. t. } u(t)\big|_{t=0} = u_0. \tag{5.3.16}$$

Taking the first-order variation of the model, we obtain the following variational problem

$$\frac{d\delta u}{dt} + \frac{\partial a}{\partial u}\delta u + \frac{\partial a}{\partial \boldsymbol{\theta}}\delta\boldsymbol{\theta} = 0,\ 0 \le t \le t_f;\ \text{s. t. } \delta \mathrm{u}\big|_{t=0} = 0. \tag{5.3.17}$$

Multiplying (5.3.17) by a differentiable function $\psi(t)$ and then integrating it over the interval $[0,t_f]$, we obtain

$$\int_0^{t_f}\left(\psi\frac{d\delta u}{dt} + \psi\frac{\partial a}{\partial u}\delta u + \psi\frac{\partial a}{\partial \boldsymbol{\theta}}\delta\boldsymbol{\theta}\right)dt = 0. \tag{5.3.18}$$

Applying integration by parts to the first term in (5.3.18), the operator (d / dt) is moved from δu to ψ, and we obtain

$$(\psi\delta u)\Big|_0^{t_f} + \int_0^{t_f}\left(-\frac{d\psi}{dt} + \frac{\partial a}{\partial u}\psi\right)\delta u dt + \int_0^{t_f}\frac{\partial a}{\partial \boldsymbol{\theta}}\,\psi\,\delta\boldsymbol{\theta}\mathrm{dt} = 0. \tag{5.3.19}$$

Combining (5.3.19) with the variational equation of the performance function (5.3.5) and noting that the integration region *R* is now the interval $[0, t_f]$, we have

$$\begin{aligned}(\psi\delta u)\Big|_0^{t_f} &+ \int_0^{t_f}\left(-\frac{d\psi}{dt} + \frac{\partial a}{\partial u}\psi + \frac{\partial f}{\partial u}\right)\delta u dt \\ &+ \int_0^{t_f}\left(\frac{\partial a}{\partial \boldsymbol{\theta}}\psi + \frac{\partial f}{\partial \boldsymbol{\theta}}\right)\delta\boldsymbol{\theta} dt = 0.\end{aligned} \tag{5.3.20}$$

Letting $\psi(t)$ be the solution of the following adjoint problem

$$-\frac{d\psi}{dt} + \frac{\partial a}{\partial u}\psi + \frac{\partial f}{\partial u} = 0,\ 0 \le t \le t_f;\ \text{s. t. } \psi(t)\Big|_{t=t_f} = 0, \tag{5.3.21}$$

the second term in (5.3.20) is now eliminated and we obtain

$$\delta E = \int_0^{t_f}\left(\frac{\partial f}{\partial \boldsymbol{\theta}} + \frac{\partial a}{\partial \boldsymbol{\theta}}\psi\right)\delta\boldsymbol{\theta} dt. \tag{5.3.22}$$

Note that the first term $(\psi\delta u)\Big|_0^{t_f}$ in (5.3.20) also drops because of the imposed final condition $\psi(t)\Big|_{t=t_f} = 0$ in (5.3.21) and the initial condition $\delta u\Big|_{t=0} = 0$ in (5.3.17). The adjoint problem (5.3.21) is a backward-in-time problem but can be transferred into a forward-in-time problem via the transformation $\tau = t_f - t$. After the adjoint-state $\psi(t)$ is solved, the gradient of *E* can be calculated directly from (5.3.22).

In this example, the adjoint equation was obtained by integration by parts. In order to find the adjoint operators for distributed parameter models, however, we need to use the Green's theorem, which is a generalized form of integration by parts.

5.3.2.1 Green's Theorem and Its Extensions

Recall the Gauss' divergence theorem from multivariable calculus

$$\int_\Omega \nabla\cdot\mathbf{F}d\Omega = \int_\Gamma \mathbf{F}\cdot\mathbf{n}d\Gamma, \tag{5.3.23}$$

where $\mathbf{F}$ is a continuously differentiable vector function defined on a spatial region $\boldsymbol{\Omega}$, ∇ is the gradient operator, Γ is the bounding surface of $\boldsymbol{\Omega}$, $\nabla\cdot\mathbf{F}$ is the divergence of **F**, **n** is outward pointing unit normal vector, and $\mathbf{F}\cdot\mathbf{n}$ is the flux. In a Cartesian coordinate system, the divergence of **F** can be expressed by

$$\nabla\cdot\mathbf{F} = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}. \tag{5.3.24}$$

Letting $\mathbf{F} = \psi\nabla\phi$ in (5.3.23), where ϕ and ψ are second-order differentiable functions, we obtain *Green's first theorem*

$$\int_\Omega (\nabla\psi\cdot\nabla\phi + \psi\nabla^2\phi)d\Omega = \int_\Gamma \psi\nabla\phi\cdot\mathbf{n}d\Gamma. \tag{5.3.25}$$

Multiplying the integrands on both sides of (5.3.25) by a differentiable function u, we arrive at a useful identity

$$\int_\Omega \psi\nabla\cdot(u\nabla\phi)d\Omega = -\int_\Omega u(\nabla\psi\cdot\nabla\phi)d\Omega + \int_\Gamma u\psi\nabla\phi\cdot\mathbf{n}d\Gamma. \tag{5.3.26}$$

Letting $\mathbf{F} = \psi\nabla\phi - \phi\nabla\psi$ in (5.3.23), we obtain *Green's second theorem*

$$\int_\Omega (\psi\nabla^2\phi - \phi\nabla^2\psi)d\Omega = \int_\Gamma (\psi\nabla\phi - \phi\nabla\psi)\cdot\mathbf{n}d\Gamma. \tag{5.3.27}$$

Multiplying the integrands on both sides of (5.3.27) by any differentiable function K, we arrive at another useful identity

$$\begin{aligned}\int_\Omega \psi\nabla\cdot(K\nabla\phi)d\Omega &= \int_\Omega \phi\nabla\cdot(K\nabla\psi)d\Omega \\ &+\int_\Gamma K(\psi\nabla\phi - \phi\nabla\psi)\cdot\mathbf{n}d\Gamma.\end{aligned} \tag{5.3.28}$$

Finally, letting $\mathbf{F} = \phi\psi\mathbf{v}$ in (5.3.23), where $\mathbf{v}$ is a differentiable vector function, we obtain

$$\int_\Omega \psi\nabla\cdot(\mathbf{v}\phi)d\Omega = -\int_\Omega \phi\mathbf{v}\cdot\nabla\psi d\Omega + \int_\Gamma \psi\phi\mathbf{v}\cdot\mathbf{n}d\Gamma. \tag{5.3.29}$$

The above identities will be used to derive the rules of adjoint operation.

Example 5.6 *The adjoint problem of groundwater flow in a confined aquifer*
Two-dimensional groundwater flow in confined aquifer is modeled by

$$S\frac{\partial\phi}{\partial t} - \nabla\cdot(T\nabla\phi) - Q = 0,\ (x,y)\in\Omega,\ 0\le t\le t_f \tag{5.3.30}$$

subject to initial and boundary conditions

$$\phi\big|_{t=0} = f_0,\ \phi\big|_{\Gamma_1} = f_1,\ T\nabla\phi\cdot\mathbf{n}\big|_{\Gamma_2} = f_2,$$

where $\phi(x,y,t)$ is the hydraulic head; $S(x,y)$ and $T(x,y)$ are the aquifer storativity and transmissivity, respectively; the flow region Ω is bounded by $\Gamma = \Gamma_1 \cup \Gamma_2$; and f_0, f_1, and f_2 are known functions. The variational problem of this model is

$$S\frac{\partial \delta\phi}{\partial t} - \nabla\cdot(T\nabla\delta\phi) - \nabla\cdot(\delta T\nabla\phi) = 0,\ (x,y)\in\Omega,\ 0 \le t \le t_f \tag{5.3.31}$$

subject to initial and boundary conditions

$$\delta\phi\Big|_{t=0} = 0,\ \ \delta\phi\Big|_{\Gamma_1} = 0,\ \ (\delta T\nabla\phi + T\nabla\delta\phi)\cdot\mathbf{n}\Big|_{\Gamma_2} = 0. \tag{5.3.32}$$

Multiplying the variational equation by $\psi(x,y,t)$ and integrating it over the region $[0,\ t_f]\times(\Omega)$ yield

$$\begin{aligned}&\int_0^{t_f}\int_\Omega\left[S\delta\phi\frac{\delta\psi}{\delta t} + \psi\nabla\cdot(T\nabla\delta\phi) + \psi\nabla\cdot(\delta T\nabla\phi)\right]d\Omega dt\\ &-\int_\Omega (S\psi\delta\phi)\Big|_0^{t_f}\, d\Omega = 0.\end{aligned} \tag{5.3.33}$$

The second term in the above equation can be eliminated by applying the zero initial condition on $\delta\phi$ and imposing the following zero final condition on ψ

$$\psi(x,y,t)\Big|_{t=t_f} = 0,\ (x,y)\in\Omega. \tag{5.3.34}$$

Furthermore, using the two-dimensional forms of identities (5.3.26) and (5.3.28), (5.3.33) can be rewritten as

$$\begin{aligned}&\int_0^{t_f}\left\{\int_\Omega\left[S\frac{\partial\psi}{\partial t} + \nabla\cdot(T\nabla\psi)\right]\delta\phi d\Omega - \int_\Omega(\nabla\psi\cdot\nabla\phi)\delta T d\Omega\right.\\ &-\int_\Gamma \delta\phi T\nabla\psi\cdot\mathbf{n}d\Gamma + \int_\Gamma \psi T\nabla\delta\phi\cdot\mathbf{n}d\Gamma\\ &\left.+\int_\Gamma \delta T\psi\nabla\phi\cdot\mathbf{n}d\Gamma\right\}dt = 0.\end{aligned} \tag{5.3.35}$$

Setting the following zero-boundary conditions on ψ

$$\psi\Big|_{\Gamma_1} = 0 \text{ and } T\nabla\psi\cdot\mathbf{n}\Big|_{\Gamma_2} = 0, \tag{5.3.36}$$

and using the zero-boundary condition on $\delta\phi$, we have

$$\int_0^{t_f}\int_\Omega\left\{\left[S\frac{\partial\psi}{\partial t} + \nabla\cdot(T\nabla\psi)\right]\delta\phi - [\nabla\psi\cdot\nabla\phi]\delta T\right\}d\Omega dt = 0, \tag{5.3.37}$$

in which all integrals along Γ in (5.3.35) are eliminated because of the zero-boundary conditions. To calculate the gradient $\partial E/\partial T$, we rewrite (5.3.5) as

$$\delta E = \int_0^{t_f}\int_\Omega\left(\frac{\partial f}{\partial\phi}\delta\phi + \frac{\partial f}{\partial T}\delta T\right)d\Omega dt. \tag{5.3.38}$$

Subtracting (5.3.37) from (5.3.38), the term associated with $\delta\phi$ is eliminated, provided that ψ is the solution of the following adjoint problem

$$-S\frac{\partial \psi}{\partial t} - \nabla\cdot(T\nabla\psi) + \frac{\partial f}{\partial \phi} = 0,\ (x,y) \in \Omega,\ 0 \le t \le t_f, \tag{5.3.39}$$

subject to initial and boundary conditions

$$\psi\big|_{t=t_f} = 0,\ \psi\big|_{\Gamma_1} = 0,\ T\nabla\psi\cdot\mathbf{n}\big|_{\Gamma_2} = 0.$$

As a result, we obtain the functional partial derivative

$$\delta E = \int_0^{t_f}\int_\Omega \left(\frac{\partial f}{\partial T} + \nabla\psi\cdot\nabla\phi\right)\delta T d\Omega dt. \tag{5.3.40}$$

Because of the final condition, problem (5.3.39) is a backward-in-time problem. For convenience, we can use transformation $\tau = t_f - t$ to transfer it into a forward-in-time problem

$$S\frac{\partial \psi_f}{\partial \tau} - \nabla\cdot(T\nabla\psi_f) + \frac{\partial f}{\partial \phi} = 0,\ (x,y) \in \Omega,\ 0 \le t \le t_f, \tag{5.3.41}$$

subject to initial and boundary conditions

$$\psi_f\big|_{\tau=0} = 0,\ \psi_f\big|_{\Gamma_1} = 0,\ T\nabla\psi_f\cdot\mathbf{n}\big|_{\Gamma_2} = 0.$$

Comparing problem (5.3.41) with the primary problem (5.3.30), we see that the two problems are different only in the sink/source term and subsidiary conditions. Therefore, the code that is used to solve the primary problem can be adapted for solving the adjoint problem by making minor modifications to it. After $\psi_f(x,y,\tau)$ is obtained, we have the adjoint-state $\psi(x,y,t) = \psi_f(x,y,t_f - t)$.

We can use (5.3.40) to calculate δE for any increment δT after solving for ϕ and ψ. In a numerical model, a distributed parameter like $T(x,y)$ is approximated by a finite-dimensional vector (parameterization)

$$T(x,y) \approx \sum_{j=1}^{m} T_j\beta_j(x,y), \tag{5.3.42}$$

where $(\beta_1,\beta_2,\cdots,\beta_m)$ are basis functions. Using (5.3.40) and noting that $\partial T / \partial T_j = \beta_j$, we obtain

$$\frac{\partial E}{\partial T_j} = \int_0^{t_f}\int_\Omega \left(\frac{\partial f}{\partial T} + \nabla\phi\cdot\nabla\psi\right)\beta_j d\Omega dt,\quad (j = 1,2,\cdots,m). \tag{5.3.43}$$

In a special case, when the flow region is divided into m zones and $T(x,y) = T_j$ in the *j*-th zone (Ω_j), we have

$$\frac{\partial E}{\partial T_j} = \int_0^{t_f} \int_{\Omega_j} \left(\frac{\partial f}{\partial T} + \nabla\phi \cdot \nabla\psi \right) d\Omega dt, \quad (j = 1, 2, \cdots, m). \tag{5.3.44}$$

5.3.2.2 Some Rules of Adjoint Operation

Using the extended forms of Green's theorem, we can derive rules of adjoint operation as summarized in Table 5.1 (Sun and Yeh 1990), in which the symbol • represents a function that an operator is active on it, *f* and *g* are scalar functions, $\mathbf{v}$ is a vector function, $\mathbf{K}$ is a symmetric tensor function, and $\mathbf{n}$ is the unit normal vector of boundary (Γ). The table covers almost all operators often encountered in EWR models. Using these rules, derivation of the adjoint problem for a given model becomes straightforward.

As an example, comparing (5.3.17) with (5.3.21) in Example 5.5, we see that the term $(d/dt)\delta u$ in the variational equation is replaced by $(-d/dt)\psi$ in the adjoint equation. This uses the rule in the second row (the first two columns) of Table 5.1.

Similarly, in Example 5.6 terms $-\nabla\cdot(T\nabla\delta\phi)$ and $-\nabla\cdot(\delta T\nabla\phi)$ in the variational equation (5.3.31) are replaced by terms $-\nabla\cdot(T\nabla\psi)$ and $\nabla\phi\cdot\nabla\psi$, respectively, in the adjoint equation and for the derivative calculation. This applies the rule listed in the fourth row and the eighth row of Table 5.1, respectively. Let us consider one more application in the following example.

Example 5.7 *Application of adjoint operation rules*
Consider the following nonlinear heat transport model

$$\mathcal{L} \doteq \rho c \frac{\partial T}{\partial t} - \nabla\cdot[k(T)\nabla T] - Q = 0, \mathbf{x} \in \Omega, 0 \le t \le t_f, \tag{5.3.45}$$

subject to initial and boundary conditions

$$T\big|_{t=0} = f_0, \ T\big|_{\Gamma_1} = f_1, \text{ and } -k(T)\nabla T\cdot\mathbf{n}\big|_{\Gamma_2} = f_2,$$

where T is temperature, ρ is density, c is heat capacity, $k(T)$ is thermal conductivity, and Q is heat source density. Taking variation of the model equation, we obtain the following gradient operators

$$\begin{aligned} \nabla_T\mathcal{L} &= \rho c(T)\frac{\partial}{\partial t}\bullet - \nabla\cdot[k'(T)\nabla T\bullet] - \nabla\cdot[k(T)\nabla\bullet] \\ \nabla_k\mathcal{L} &= -\nabla\cdot[\nabla T\bullet] \\ \nabla_c\mathcal{L} &= \rho\frac{\partial T}{\partial t}\bullet \end{aligned} \tag{5.3.46}$$

Using the adjoint operation rules in Table 5.1, we can obtain the following adjoint operators,

$$\begin{aligned} &\nabla_T^* \mathcal{L} = -\rho c \frac{\partial}{\partial t} \bullet + k'(T) \nabla T \cdot \nabla \bullet - \nabla \cdot [k(T) \nabla \bullet] \\ &\nabla_k^* \mathcal{L} = \nabla T \cdot \nabla \bullet \\ &\nabla_c^* \mathcal{L} = \rho \frac{\partial T}{\partial t} \bullet . \end{aligned} \tag{5.3.47}$$

Therefore, the adjoint problem of the model can be obtained directly as

$$\begin{aligned} &\rho c \frac{\partial \psi}{\partial t} + \nabla \cdot [k(T) \nabla \psi] - k'(T) \nabla T \cdot \nabla \psi - \frac{\partial f}{\partial T} = 0, \\ &\mathbf{x} \in \Omega, \ 0 \le t \le t_f \end{aligned} \tag{5.3.48}$$

subject to zero final and boundary conditions

$$\psi\Big|_{t=t_f} = 0, \ \ \psi\Big|_{\Gamma_1} = 0, \text{ and } k(T) \nabla \psi \cdot \mathbf{n}\Big|_{\Gamma_2} = 0.$$

Finally, the functional derivatives with respect to k and c are given by

$$\delta E = \int_0^{t_f} \int_{(\Omega)} \left(\frac{\partial f}{\partial k} + \nabla T \cdot \nabla \psi \right) \delta k d\Omega dt, \tag{5.3.49}$$

$$\delta E = \int_0^{t_f} \int_{(\Omega)} \left(\frac{\partial f}{\partial c} + \rho \frac{\partial T}{\partial t} \psi \right) \delta c d\Omega dt. \tag{5.3.50}$$

■

Table 5.1 Rules of adjoint operation

Term in $\nabla \mathcal{L}$	Term in $(\nabla \mathcal{L})^*$	Term in $\nabla \mathcal{L}_B$	Term in $(\nabla \mathcal{L}_B)^*$
$f\bullet$	$f\bullet$	$\mathbf{v} \cdot \mathbf{n}$	$\mathbf{0}$
$f(\partial / \partial t)\bullet$	$-(\partial / \partial t)(f\bullet)$	$f\nabla \bullet \cdot \mathbf{n}$	$f\nabla \bullet \cdot \mathbf{n}$
$-(\partial / \partial t)(f\bullet)$	$f(\partial / \partial t)\bullet$	$\mathbf{K}\nabla \bullet \cdot \mathbf{n}$	$\mathbf{K}\nabla \bullet \cdot \mathbf{n}$
$\nabla \cdot (f\nabla \bullet)$	$\nabla \cdot (f\nabla \bullet)$	$\mathbf{K}\nabla(f\bullet) \cdot \mathbf{n}$	$f\mathbf{K}\nabla \bullet \cdot \mathbf{n}$
$\nabla \cdot (\mathbf{K}\nabla \bullet)$	$\nabla \cdot (\mathbf{K}\nabla \bullet)$		
$\mathbf{v} \cdot (\nabla \bullet)$	$-\nabla \cdot (\mathbf{v}\bullet)$		
$\nabla \cdot (\mathbf{v}\bullet)$	$-\mathbf{v} \cdot (\nabla \bullet)$		
$\nabla \cdot (f\nabla g\bullet)$	$-f\nabla g \cdot \nabla \bullet$		
$\nabla \cdot (\mathbf{K}\nabla g\bullet)$	$-\mathbf{K}\nabla g \cdot \nabla \bullet$		
$\nabla \cdot [\mathbf{K}\nabla(f\bullet)]$	$f\nabla \cdot (\mathbf{K}\nabla \bullet)$		

5.3.3 Multistate Models

In many EWR real case studies, we need multistate models governed by coupled PDEs. Extending the adjoint-state method from the single-state case to multistate case is straightforward. Using the same notations given in Sect. 5.2.1, we can write the variational problem of a general multistate model $\mathcal{L}(\mathbf{u}, \boldsymbol{\theta}) = \mathbf{0}$ as

$$\nabla_{\mathbf{u}}\mathcal{L}\delta\mathbf{u} + \nabla_{\boldsymbol{\theta}}\mathcal{L}\delta\boldsymbol{\theta} = \mathbf{0}, \tag{5.3.51}$$

where $\nabla_{\mathbf{u}}\mathcal{L} = [\partial\mathcal{L}_i / \partial u_j]_{k\times k}$ and $\nabla_{\boldsymbol{\theta}}\mathcal{L} = [\partial\mathcal{L}_i / \partial\theta_j]_{k\times m}$ are gradient operators with respect to $\mathbf{u} = [u_i]_{k\times 1}$ and $\boldsymbol{\theta} = [\theta_j]_{m\times 1}$, respectively; $\delta\mathbf{u} = [\delta u_i]_{k\times 1}$ and $\delta\boldsymbol{\theta} = [\delta\theta_j]_{m\times 1}$ denote the corresponding variations. The subscripts outside square brackets in the above indicate dimensions of either a matrix or vector, and subscripts inside square brackets denote element indices. All elements may be functions defined on a time-space domain R. The adjoint model of a multistate model can be derived in the same way as in Sect. 5.3.1 for a single-state model.

5.3.3.1 Deriving the Adjoint Model

The inner product of two elements $\mathbf{w}$ and $\mathbf{v}$ in the multistate space is defined by

$$(\mathbf{w}, \mathbf{v})_R = \int_R (\mathbf{w}^T\mathbf{v})dR. \tag{5.3.52}$$

For any matrix operator $\mathbf{A}$, we can find its adjoint $\mathbf{A}^+ = (\mathbf{A}^*)^T$ such that

$$(\mathbf{w}, \mathbf{A}\mathbf{v})_R = (\mathbf{v}, \mathbf{A}^+\mathbf{w})_R. \tag{5.3.53}$$

Let $\mathbf{w}$ be a vector function $\boldsymbol{\psi}$ having continuous second-order derivatives, the matrix operator $\mathbf{A}$ be $\nabla_{\mathbf{u}}\mathcal{L}$, and $\mathbf{v}$ be $\delta\mathbf{u}$ in the above equation, we have

$$(\boldsymbol{\psi}, \nabla_{\mathbf{u}}\mathcal{L}\delta\mathbf{u})_R = (\delta\mathbf{u}, \nabla_{\mathbf{u}}^+\mathcal{L}\boldsymbol{\psi})_R, \tag{5.3.54}$$

where $\nabla_{\mathbf{u}}^+\mathcal{L} = (\nabla_{\mathbf{u}}\mathcal{L})^+$. Similarly, we have also

$$(\boldsymbol{\psi}, \nabla_{\boldsymbol{\theta}}\mathcal{L}\delta\boldsymbol{\theta})_R = (\delta\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}}^+\mathcal{L}\boldsymbol{\psi})_R, \tag{5.3.55}$$

where $\nabla_{\boldsymbol{\theta}}^+\mathcal{L} = (\nabla_{\boldsymbol{\theta}}\mathcal{L})^+$. Using (5.3.54) and (5.3.55) to the inner production of $\boldsymbol{\psi}$ and the variation equation (5.3.51) gives

$$(\delta\mathbf{u}, \nabla_{\mathbf{u}}^+\mathcal{L}\boldsymbol{\psi})_R + (\delta\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}}^+\mathcal{L}\boldsymbol{\psi})_R = 0. \tag{5.3.56}$$

Next, we define the following performance function

$$E(\mathbf{u}, \boldsymbol{\theta}) = \int_R f(\mathbf{u}, \boldsymbol{\theta})dR, \tag{5.3.57}$$

for which the first-order variation is given by

$$\delta E = \int_R \left(\frac{\partial f}{\partial \mathbf{u}} \delta \mathbf{u} + \frac{\partial f}{\partial \boldsymbol{\theta}} \delta \boldsymbol{\theta} \right) dR = \left(\delta \mathbf{u}, \left[\frac{\partial f}{\partial \mathbf{u}} \right]^T \right)_R + \left(\delta \boldsymbol{\theta}, \left[\frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} \right]^T \right)_R, \tag{5.3.58}$$

where $\partial f / \partial \mathbf{u} = [\partial f / \partial u_i]_{1\times k}$ and $\partial f / \partial \boldsymbol{\theta} = [\partial \mathbf{f} / \partial \theta_i]_{1\times m}$. Adding (5.3.56) to (5.3.58) yields

$$\delta E = \left(\delta \mathbf{u}, \nabla_{\mathbf{u}}^{+} \boldsymbol{\mathcal{L}} \boldsymbol{\psi} + \left[\frac{\partial f}{\partial \mathbf{u}} \right]^T \right)_R + \left(\delta \boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}}^{+} \boldsymbol{\mathcal{L}} \boldsymbol{\psi} + \left[\frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} \right]^{\mathbf{T}} \right)_R. \tag{5.3.59}$$

When ψ is the solution to the following equation

$$(\nabla_{\mathbf{u}}^{+} \boldsymbol{\mathcal{L}} \psi) + \left(\frac{\partial f}{\partial \mathbf{u}} \right)^T = \mathbf{0}, \tag{5.3.60}$$

then the term associated with $\delta \mathbf{u}$ in (5.3.59) vanishes and we have

$$\delta E = \int_R \left[\frac{\partial f}{\partial \boldsymbol{\theta}} + (\nabla_{\boldsymbol{\theta}}^{+} \boldsymbol{\mathcal{L}} \boldsymbol{\psi})^T \right] \delta \boldsymbol{\theta} dR. \tag{5.3.61}$$

Equation (5.3.60) is called the adjoint equation and $\boldsymbol{\psi}$ the adjoint-state of the multi-state model. With the adjoint state, the gradient of the performance function E with respect to each component of $\boldsymbol{\theta}$ can be calculated from

$$\frac{\partial E}{\partial \theta_j} = \int_R \left[\frac{\partial f}{\partial \theta_j} + (\nabla_{\theta_j}^{+} \boldsymbol{\mathcal{L}} \boldsymbol{\psi})^T \right] dR, \ (j = 1, 2, \cdots, m). \tag{5.3.62}$$

Because the same adjoint-state $\boldsymbol{\psi}$ is used in (5.3.62) for all components, all gradients can be calculated by solving the adjoint equation (5.3.60) only once.

Note that when the primary model equation $\boldsymbol{\mathcal{L}}(\mathbf{u}, \boldsymbol{\theta}) = \mathbf{0}$ contains ODEs and/or PDEs, initial and boundary conditions must be supplied to Eq. (5.3.60) to form a complete adjoint problem. If the boundary conditions of the primary model are represented in a general form by $\boldsymbol{\mathcal{L}}_B(\mathbf{u}, \boldsymbol{\theta}) = \mathbf{0}$, where $\boldsymbol{\mathcal{L}}_B$ is a set of operators defined on the boundary of the time-space domain R, the variational equation (5.3.51) has boundary conditions

$$\nabla_{\mathbf{u}} \boldsymbol{\mathcal{L}}_B \delta \mathbf{u} + \nabla_{\boldsymbol{\theta}} \boldsymbol{\mathcal{L}}_B \delta \boldsymbol{\theta} = \mathbf{0}. \tag{5.3.63}$$

Boundary conditions for the adjoint-state equation (5.3.60) can then be obtained by finding the adjoint operators $\nabla_{\mathbf{u}}^{+} \boldsymbol{\mathcal{L}}_B$ and $\nabla_{\boldsymbol{\theta}}^{+} \boldsymbol{\mathcal{L}}_B$.

5.3.3.2 Finding the Adjoint Operators

The adjoint operators used above for PDEs models can be derived by the Green formulae (Sun and Yeh 1990) as described in the last section for the single-state case. For example, we can use Table 5.1 to find all elements of the adjoint matrix $\nabla_{\mathbf{u}}^{*}\boldsymbol{\mathcal{L}}$, its transpose matrix $(\nabla_{\mathbf{u}}^{*}\boldsymbol{\mathcal{L}})^{T}$ then is $\nabla_{\mathbf{u}}^{+}\boldsymbol{\mathcal{L}}$. And, $\nabla_{\boldsymbol{\theta}}^{+}\boldsymbol{\mathcal{L}}$, $\nabla_{\mathbf{u}}^{+}\boldsymbol{\mathcal{L}}_{B}$ and $\nabla_{\boldsymbol{\theta}}^{+}\boldsymbol{\mathcal{L}}_{B}$ can be found by the same way as shown in the following examples.

Example 5.8 *Derive the adjoint equations for the Saint-Venant model (see also Example 1.4 in Chap. 1)*

$$\begin{cases} \mathcal{L}_1 \equiv \dfrac{\partial h}{\partial t} + u\dfrac{\partial h}{\partial x} + h\dfrac{\partial u}{\partial x} - q = 0 \\ \mathcal{L}_2 \equiv \dfrac{\partial u}{\partial t} + g\dfrac{\partial h}{\partial x} + u\dfrac{\partial u}{\partial x} - F(u,h) = 0 \end{cases}. \tag{5.3.64}$$

Taking the first-order variation of these equations gives

$$\begin{aligned} &\nabla_h\mathcal{L}_1 = \frac{\partial}{\partial t}\bullet + u\frac{\partial}{\partial x}\bullet + \frac{\partial u}{\partial x}\bullet,\ \nabla_u\mathcal{L}_1 = \frac{\partial h}{\partial x}\bullet + h\frac{\partial}{\partial x}\bullet, \\ &\nabla_h\mathcal{L}_2 = g\frac{\partial}{\partial x}\bullet - \frac{\partial F}{\partial h}\bullet,\ \nabla_u\mathcal{L}_2 = \frac{\partial}{\partial t}\bullet + \frac{\partial u}{\partial x}\bullet + u\frac{\partial}{\partial x}\bullet - \frac{\partial F}{\partial u}\bullet. \end{aligned} \tag{5.3.65}$$

Using Table 5.1, we can find

$$\begin{aligned} &\nabla_h^*\mathcal{L}_1 = -\frac{\partial}{\partial t}\bullet + u\frac{\partial}{\partial x}\bullet + \frac{\partial u}{\partial x}\bullet,\ \nabla_u^*\mathcal{L}_1 = \frac{\partial h}{\partial x}\bullet + h\frac{\partial}{\partial x}\bullet \\ &\nabla_h^*\mathcal{L}_2 = g\frac{\partial}{\partial x}\bullet - \frac{\partial F}{\partial h}\bullet,\ \nabla_u^*\mathcal{L}_2 = -\frac{\partial}{\partial t}\bullet + \frac{\partial u}{\partial x}\bullet + u\frac{\partial}{\partial x}\bullet - \frac{\partial F}{\partial u}\bullet. \end{aligned} \tag{5.3.66}$$

Applying (5.3.60), we have

$$\begin{pmatrix} \nabla_h^*\mathcal{L}_1 & \nabla_h^*\mathcal{L}_2 \\ \nabla_u^*\mathcal{L}_1 & \nabla_u^*\mathcal{L}_2 \end{pmatrix}\begin{pmatrix} \psi_1 \\ \psi_2 \end{pmatrix} + \begin{pmatrix} \partial f / \partial h \\ \partial f / \partial u \end{pmatrix} = \mathbf{0}, \tag{5.3.67}$$

which, after substituting the gradient operators from (5.3.66), leads to the following adjoint-state equations

$$\begin{cases} \dfrac{\partial \psi_1}{\partial t} - u\dfrac{\partial \psi_1}{\partial x} - \dfrac{\partial u}{\partial x}\psi_1 - g\dfrac{\partial \psi_2}{\partial x} + \dfrac{\partial F}{\partial h}\psi_2 = \dfrac{\partial f}{\partial h} \\ \dfrac{\partial \psi_2}{\partial t} - u\dfrac{\partial \psi_2}{\partial x} - \dfrac{\partial u}{\partial x}\psi_2 + \dfrac{\partial F}{\partial u}\psi_2 - h\dfrac{\partial \psi_1}{\partial x} - \dfrac{\partial h}{\partial x}\psi_1 = \dfrac{\partial f}{\partial u} \end{cases}. \tag{5.3.68}$$

These equations are solved with zero final and zero-boundary conditions.

Example 5.9 *The adjoint problem for a coupled flow and mass transport model*
Let us consider the mass transport problem in a transient groundwater flow field and assume that the aquifer is isotropic, heterogeneous, and unconfined. The coupled model is given by (see Examples 1.9 and 1.10 in Chap. 1)

$$\begin{cases} \mathcal{L}_1 \equiv S_y \dfrac{\partial h}{\partial t} - \nabla\cdot[K(h-b)\nabla h] - Q = 0 \\ \mathcal{L}_2 \equiv \dfrac{\partial \theta_e C}{\partial t} - \nabla\cdot[\theta_e \mathbf{D}\nabla C] + \nabla\cdot(\theta_e \mathbf{V} C) - S(C) = 0 \\ \qquad\qquad 0 \le t \le t_f, \mathbf{x} \in (\Omega) \end{cases} \tag{5.3.69}$$

subject to initial and boundary condition

$$\begin{array}{l} h\big|_{t=0} = f_0,\ h\big|_{\Gamma_1} = f_1, \text{ and } K(h-b)\nabla h\cdot\mathbf{n}\big|_{\Gamma_2} = f_2 \\ C\big|_{t=0} = g_0,\ C\big|_{\Gamma_1} = g_1, \text{ and } (\theta_e \mathbf{D}\nabla C - \theta_e \mathbf{V} C)\cdot\mathbf{n}\big|_{\Gamma_2} = g_2. \end{array}$$

In the above equation, h is the groundwater level, S_y is specific yield, b is the elevation of aquifer bottom, Q is the flow source term, C is concentration, θ_e is effective porosity (note the subscript e is used here to avoid confusion with generic parameter notation θ), $\mathbf{V}$ is flow velocity, $S(C)$ is the contaminant source term, and $\mathbf{D}$ is the hydrodynamic dispersion coefficient depending on flow velocity $\mathbf{V}$, longitudinal dispersivity α_L, and transverse dispersivity α_T (see Example 1.7). Note that the state variables h and C are implicitly coupled because C depends on $\mathbf{V}$ which, in turn, depends on h through Darcy's law.

The following gradient operators can be obtained directly by taking the first-order variation of the governing equations

$$\begin{aligned} \nabla_h \mathcal{L}_1 &= S_y \frac{\partial}{\partial t}\bullet - \nabla\cdot[K(h-b)\nabla\bullet] - \nabla\cdot[K\nabla h\ \bullet] \\ \nabla_C \mathcal{L}_1 &= 0\ \bullet \\ \nabla_h \mathcal{L}_2 &= \nabla\cdot[K\mathbf{E}_C\nabla\bullet] - \nabla\cdot(KC\nabla\bullet) \\ \nabla_C \mathcal{L}_2 &= \frac{\partial}{\partial t}(\theta_e\bullet) - \nabla\cdot[\theta_e \mathbf{D}\nabla\bullet] + \nabla\cdot(\theta_e \mathbf{V}\bullet) - S'(C)\bullet \end{aligned} \tag{5.3.70}$$

where $\mathbf{E}_C = (\partial\mathbf{D}/\partial\mathbf{V})\nabla C$ and $\partial\mathbf{D}/\partial\mathbf{V}$ is defined as

$$\frac{\partial \mathbf{D}}{\partial \mathbf{V}} = \left(\frac{\partial \mathbf{D}}{\partial V_1}, \frac{\partial \mathbf{D}}{\partial V_2}, \frac{\partial \mathbf{D}}{\partial V_3}\right).$$

Darcy's law was used in the derivation of $\nabla_h \mathcal{L}_2$ in (5.3.70). A variation δh causes a variation $\delta \mathbf{V} = -(K/\theta_e)\nabla \delta h$. Using Table 5.1, we can find their adjoint operators

$$\begin{aligned}
\nabla_h^* \mathcal{L}_1 &= -S_y \frac{\partial}{\partial t} \bullet - \nabla\cdot[K(h-b)\nabla\bullet] + K\nabla h\cdot\nabla\bullet \\
\nabla_C^* \mathcal{L}_1 &= 0\bullet \\
\nabla_h^* \mathcal{L}_2 &= \nabla\cdot[K\mathbf{E}_C\nabla\bullet] - \nabla\cdot(KC\nabla\bullet) \\
\nabla_C^* \mathcal{L}_2 &= -\theta_e \frac{\partial}{\partial t}\bullet - \nabla\cdot[\theta_e \mathbf{D}\nabla\bullet] - \theta_e \mathbf{V}\cdot\nabla\bullet - S'(C)\bullet.
\end{aligned} \tag{5.3.71}$$

Let $\boldsymbol{\psi} = (\psi_1, \psi_2)^T$ be the adjoint states. The coupled adjoint equation (5.3.60) is

$$\begin{pmatrix} \nabla_h^* \mathcal{L}_1 & \nabla_h^* \mathcal{L}_2 \\ \nabla_C^* \mathcal{L}_1 & \nabla_C^* \mathcal{L}_2 \end{pmatrix} \begin{pmatrix} \psi_1 \\ \psi_2 \end{pmatrix} = \begin{pmatrix} -\partial f/\partial h \\ -\partial f/\partial C \end{pmatrix}. \tag{5.3.72}$$

Or in the scalar form

$$\begin{aligned}
&S_y \frac{\partial \psi_1}{\partial t} + \nabla\cdot[K(h-b)\nabla\psi_1] - K\nabla h\cdot\nabla\psi_1 \\
&= \frac{\partial f}{\partial h} + \nabla\cdot[K\mathbf{E}_C\nabla\psi_2] - \nabla\cdot(KC\nabla\psi_2)
\end{aligned} \tag{5.3.73}$$

$$\theta_e \frac{\partial \psi_2}{\partial t} + \nabla\cdot[\theta_e \mathbf{D}\nabla\psi_2] + \theta_e \mathbf{V}\cdot\nabla\psi_2 + S'(C)\psi_2 = \frac{\partial f}{\partial C}. \tag{5.3.74}$$

For this problem, the boundary operators on Γ_2 are

$$\begin{aligned}
\nabla_h \mathcal{L}_{B1} &= [K(h-b)\nabla\bullet + K\nabla h\bullet]\cdot\mathbf{n} \\
\nabla_C \mathcal{L}_{B1} &= [0\bullet]\cdot\mathbf{n} \\
\nabla_h \mathcal{L}_{B2} &= [-K\mathbf{E}_C\nabla\bullet + KC\nabla\bullet]\cdot\mathbf{n} \\
\nabla_C \mathcal{L}_{B2} &= [\theta_e \mathbf{D}\nabla\bullet - \theta_e \mathbf{V}\bullet]\cdot\mathbf{n}
\end{aligned} \tag{5.3.75}$$

From the right two columns of Table 5.1, we can find their adjoint operators

$$\begin{aligned}
\nabla_h^* \mathcal{L}_{B1} &= [K(h-b)\nabla\bullet]\cdot\mathbf{n} \\
\nabla_C^* \mathcal{L}_{B1} &= [0\bullet]\cdot\mathbf{n} \\
\nabla_h^* \mathcal{L}_{B2} &= [-K\mathbf{E}_C\nabla\bullet + KC\nabla\bullet]\cdot\mathbf{n} \\
\nabla_C^* \mathcal{L}_{B2} &= [\theta_e \mathbf{D}\nabla\bullet]\cdot\mathbf{n}
\end{aligned} \tag{5.3.76}$$

Therefore, the subsidiary conditions for ψ_1 in (5.3.73) are given by

$$\begin{aligned}&\psi_1\big|_{t=t_f} = 0, \psi_1\big|_{\Gamma_1} = 0,\\ &\text{and}\,[K(h-b)\nabla\psi_1 - K\mathbf{E}_C\nabla\psi_2 + KC\nabla\psi_2]\cdot\mathbf{n}\big|_{\Gamma_2} = 0\end{aligned} \tag{5.3.77}$$

and the subsidiary conditions for ψ_2 in (5.3.74) are given by

$$\psi_2\big|_{t=t_f} = 0,\ \psi_2\big|_{\Gamma_1} = 0, \text{ and } (\theta_e\mathbf{D}\nabla\psi_2)\cdot\mathbf{n}\big|_{\Gamma_2} = 0. \tag{5.3.78}$$

After the adjoint states ψ_1 and ψ_2 are obtained by solving the coupled adjoint problem derived above, all derivatives of the performance function E with respect to θ_j can be calculated by substituting operator $\nabla^\dagger_{\theta_j}\mathcal{L}$ into (5.3.62). For example, when θ_j is hydraulic conductivity $K(x,y,z)$, we have

$$\begin{aligned}\nabla_K\mathcal{L}_1 &= -\nabla\cdot\left[(h-b)\nabla h\bullet\right]\\ \nabla_K\mathcal{L}_2 &= \nabla\cdot[\mathbf{F}_h\nabla C\bullet] - \nabla\cdot[C\nabla h\bullet],\end{aligned} \tag{5.3.79}$$

where $\mathbf{F}_h = (\partial\mathbf{D}/\partial\mathbf{V})\nabla h$. The second equation in (5.3.79) was derived using the fact that a variation δK causes a variation $\delta\mathbf{V} = -(\nabla h/\theta_e)\delta K$, according to Darcy's law. Thus, applying the rules given in Table 5.1, we find

$$\begin{aligned}\nabla^*_K\mathcal{L}_1 &= (h-b)\nabla h\cdot\nabla\bullet\\ \nabla^*_K\mathcal{L}_2 &= -\mathbf{F}_h\nabla C\cdot\nabla\bullet + C\nabla h\cdot\nabla\bullet\end{aligned} \tag{5.3.80}$$

using which we can calculate

$$\nabla^+_K\boldsymbol{\mathcal{L}}\boldsymbol{\psi} = \begin{pmatrix}\nabla^*_K\mathcal{L}_1 & \nabla^*_K\mathcal{L}_2\end{pmatrix}\begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix}. \tag{5.3.81}$$

Substituting (5.3.81) into (5.3.62) and replacing θ_j by K, we get

$$\delta E = \int_0^{t_f}\int_\Omega\left[\frac{\partial f}{\partial K} + (h-b)\nabla h\cdot\nabla\psi_1 - \mathbf{F}_h\nabla C\cdot\nabla\psi_2 + C\nabla h\cdot\nabla\psi_2\right]\delta K\,d\Omega\,dt. \tag{5.3.82}$$

Similarly, we can obtain the gradients of performance function E with respect to other parameters (i.e., specific yield S_y, porosity θ_e, dispersivities α_L and α_T)

$$\begin{aligned}
\delta E &= \int_0^{t_f}\int_\Omega \frac{\partial h}{\partial t}\psi_1 \delta S_y d\Omega dt \\
\delta E &= \int_0^{t_f}\int_\Omega \left[\frac{\partial C}{\partial t}\psi_2 + \mathbf{D}\nabla C\cdot\nabla\psi_2 - C\mathbf{V}\cdot\nabla\psi_2\right]\delta\theta_e d\Omega dt \\
\delta E &= \int_0^{t_f}\int_\Omega \frac{\partial D}{\partial \alpha_L}\nabla C\cdot\nabla\psi_2 \delta\alpha_L d\Omega dt \\
\delta E &= \int_0^{t_f}\int_{(\Omega)} \frac{\partial D}{\partial \alpha_T}\nabla C\cdot\nabla\psi_2 \delta\alpha_T d\Omega dt.
\end{aligned} \tag{5.3.83}$$

5.3.3.3 Solution of the Multistate Adjoint Problem

The adjoint problem of the coupled groundwater flow and mass transport model can be solved by the following steps:

1. Solve the coupled flow and mass transport problem (5.3.69) by a numerical method. In each time step, do
   - Solve the flow problem to obtain the head distribution $h(\mathbf{x},t)$
   - Calculate the velocity distribution $\mathbf{V}(\mathbf{x},t)$ by Darcy's law
   - Calculate the dispersion coefficient $\mathbf{D}(\mathbf{V},\alpha_L,\alpha_T)$, and
   - Solve the mass transport problem to obtain the concentration distribution $C(\mathbf{x},t)$.
2. Solve the adjoint equation (5.3.74) with subsidiary conditions (5.3.78) to obtain the adjoint-state $\psi_2(\mathbf{x},t)$;
3. Calculate the sink/source terms $\nabla\cdot[K\mathbf{E}_C\cdot\nabla\psi_2]$ and $\nabla\cdot(KC\nabla\psi_2)$ for the adjoint equation (5.3.73);
4. Substitute the terms into (5.3.73) and solve the equation with subsidiary conditions (5.3.77) to obtain the adjoint-state $\psi_1(\mathbf{x},t)$.

Structures of the primary forward problem and the adjoint problem are basically the same except that (i) the initial condition is replaced by the final condition, (ii) a new term $K\nabla h\cdot\nabla\psi_1$ is added to (5.3.73), and (iii) the right-hand-side terms of the adjoint equations need to be calculated. Therefore, the code for solving the adjoint problem can be developed by making minor changes to the code for solving the primary forward problem. The computational accuracy of $\psi_2(\mathbf{x},t)$ is same as that of $C(\mathbf{x},t)$, but the computational accuracy of $\psi_1(\mathbf{x},t)$ is a tricky problem because it depends on $\nabla h, \nabla C$, and $\nabla\psi_2$. When the first-order finite difference method (FDM) or linear finite element method (FEM) are used, the computational accuracy of these gradients is usually poor, and moreover, overshooting and numerical dispersion errors associated with the numerical solutions of C and ψ_2 can propagate to estimates of ∇C and $\nabla \psi_2$. Therefore, use of a high-order FEM for solving these PDEs is strongly recommended.

More examples, including derivation of the adjoint problem for multiphase flow problems, can be found in Sun (1994). In practice, when the forward model is solved numerically, the discrete form of adjoint-state equations is more general and convenient.

5.3.4 Discrete Form

The discrete adjoint-state method derives adjoint equations for numerical models. It follows the same steps as those involved in the continuous adjoint method, but needs less mathematical derivations and programming effort. After temporal and/or spatial discretization, the solution of one or a set of ODEs or PDEs is reduced to the solution of a set of algebraic equations in each time step. The collection of equations from all time steps can be written as

$$\begin{cases} \mathbf{G}_1\mathbf{u}_1 = \mathbf{H}_1\mathbf{u}_0 + \mathbf{q}_1 \\ \mathbf{G}_2\mathbf{u}_2 = \mathbf{H}_2\mathbf{u}_1 + \mathbf{q}_2 \\ \cdots \quad \cdots \quad \cdots \quad \cdots \\ \mathbf{G}_{K-1}\mathbf{u}_{K-1} = \mathbf{H}_{K-1}\mathbf{u}_{K-2} + \mathbf{q}_{K-1} \\ \mathbf{G}_K\mathbf{u}_K = \mathbf{H}_K\mathbf{u}_{K-1} + \mathbf{q}_K \end{cases}, \tag{5.3.84}$$

where $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_K$ are the forward solutions at K time steps and $\mathbf{u}_0$ is given by the initial condition. For linear models, coefficient matrices $\{\mathbf{G}_k\}$, $k = 1, 2, \cdots, K$, depend only on the parameter vector $\boldsymbol{\theta}$, and the dimensions of each matrix $\mathbf{G}_k$ are $N \times N$, with N the number of nodes resulting from the spatial discretization of the state variable $\mathbf{u}$. The right-hand-side matrices $\mathbf{H}_k$ (also have dimensions $N \times N$) and forcing terms $\mathbf{q}_k$ may depend also on $\boldsymbol{\theta}$. This set of equations is solved from the first time step to the last time step. Let the objective function for differentiation be

$$E = f(\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_K; \boldsymbol{\theta}). \tag{5.3.85}$$

A variation $\delta\boldsymbol{\theta}$ will cause variations $\delta\mathbf{u}_1, \delta\mathbf{u}_2, \cdots, \delta\mathbf{u}_K$ and then a variation δE. The first-order variation of (5.3.85) results

$$\delta E = \sum_{k=1}^{K} \frac{\partial f}{\partial \mathbf{u}_k} \delta\mathbf{u}_k + \frac{\partial f}{\partial \boldsymbol{\theta}} \delta\boldsymbol{\theta}. \tag{5.3.86}$$

The dimensions of $\delta \mathbf{u}_k$ and $\delta\boldsymbol{\theta}$ are $N \times 1$ and $m \times 1$, respectively. Our purpose is to eliminate all unknown variations of the state variable from (5.3.86) so that an explicit relationship between δE and $\delta\boldsymbol{\theta}$ can be found. Taking the first-order variation of the discretized model equation (5.3.84) yields

$$\mathbf{G}_k \delta\mathbf{u}_k - \mathbf{H}_k \delta\mathbf{u}_{k-1} + \left(\frac{\partial \mathbf{G}_k}{\partial \boldsymbol{\theta}} \mathbf{u}_k - \frac{\partial \mathbf{H}_k}{\partial \boldsymbol{\theta}} \mathbf{u}_{k-1} - \frac{\partial \mathbf{q}_k}{\partial \boldsymbol{\theta}} \right) \delta\boldsymbol{\theta} = \mathbf{0}, \quad k = 1, 2, \cdots, K. \tag{5.3.87}$$

In deriving the above equation, we used the following shorthand notation

$$\frac{\partial \mathbf{G}_k}{\partial \boldsymbol{\theta}} = \left(\frac{\partial \mathbf{G}_k}{\partial \theta_1}, \frac{\partial \mathbf{G}_k}{\partial \theta_2}, \cdots, \frac{\partial \mathbf{G}_k}{\partial \theta_m} \right). \tag{5.3.88}$$

Therefore,

$$\begin{aligned}
(\delta\mathbf{G}_k)\mathbf{u}_k &= \left(\frac{\partial \mathbf{G}_k}{\partial \boldsymbol{\theta}} \delta\boldsymbol{\theta} \right) \mathbf{u}_k \\
&= \left(\frac{\partial \mathbf{G}_k}{\partial \theta_1} \delta\theta_1 + \frac{\partial \mathbf{G}_k}{\partial \theta_2} \delta\theta_2 + \cdots + \frac{\partial \mathbf{G}_k}{\partial \theta_m} \delta\theta_m \right) \mathbf{u}_k \\
&= \left(\frac{\partial \mathbf{G}_k}{\partial \theta_1} \mathbf{u}_k \delta\theta_1 + \frac{\partial \mathbf{G}_k}{\partial \theta_2} \mathbf{u}_k \delta\theta_2 + \cdots + \frac{\partial \mathbf{G}_k}{\partial \theta_m} \mathbf{u}_k \delta\theta_m \right) \\
&= \left(\frac{\partial \mathbf{G}_k}{\partial \theta_1} \mathbf{u}_k \;\; \frac{\partial \mathbf{G}_k}{\partial \theta_2} \mathbf{u}_k \;\; \cdots \;\; \frac{\partial \mathbf{G}_k}{\partial \theta_m} \mathbf{u}_k \right) \begin{pmatrix} \delta\theta_1 \\ \delta\theta_2 \\ \vdots \\ \delta\theta_m \end{pmatrix} \\
&= \left(\frac{\partial \mathbf{G}}{\partial \boldsymbol{\theta}} \mathbf{u}_k \right) \delta\boldsymbol{\theta}.
\end{aligned}$$

The term $(\partial \mathbf{H}_k / \partial \boldsymbol{\theta})\mathbf{u}_{k-1}$ was derived similarly. Multiplying an arbitrary vector $\boldsymbol{\psi}_k^T$ to the kth equation in (5.3.87), adding all K equations together, and rearranging the order of summation, we arrive at

$$\begin{aligned}
&\left\{ \sum_{k=1}^{K-1} (\boldsymbol{\psi}_k^T \mathbf{G}_k - \boldsymbol{\psi}_{k+1}^T \mathbf{H}_{k+1}) \delta\mathbf{u}_k + \boldsymbol{\psi}_K^T \mathbf{G}_K \delta\mathbf{u}_K \right\} \\
&+ \left\{ \sum_{k=1}^{K} \boldsymbol{\psi}_k^T \left(\frac{\partial \mathbf{G}_k}{\partial \boldsymbol{\theta}} \mathbf{u}_k - \frac{\partial \mathbf{H}_k}{\partial \boldsymbol{\theta}} \mathbf{u}_{k-1} - \frac{\partial \mathbf{q}_k}{\partial \boldsymbol{\theta}} \right) \right\} \delta\boldsymbol{\theta} = \mathbf{0}.
\end{aligned} \tag{5.3.89}$$

Just as in the case of continuous form we can obtain the following equation after adding (5.3.89) to (5.3.86),

$$\frac{\partial E}{\partial \boldsymbol{\theta}} = \sum_{k=1}^{K} \boldsymbol{\psi}_k^T \left(\frac{\partial \mathbf{G}_k}{\partial \boldsymbol{\theta}} \mathbf{u}_k - \frac{\partial \mathbf{H}_k}{\partial \boldsymbol{\theta}} \mathbf{u}_{k-1} - \frac{\partial \mathbf{q}_k}{\partial \boldsymbol{\theta}} \right) + \frac{\partial f}{\partial \boldsymbol{\theta}}, \tag{5.3.90}$$

provided that $\{\psi_k\}_{k=1}^K$ are the solutions to the following set of adjoint equations:

$$\begin{cases} \mathbf{G}_1^T \boldsymbol{\psi}_1 = \mathbf{H}_2^T \boldsymbol{\psi}_2 - (\partial f / \partial \mathbf{u}_1)^T \\ \mathbf{G}_2^T \boldsymbol{\psi}_2 = \mathbf{H}_3^T \boldsymbol{\psi}_3 - (\partial f / \partial \mathbf{u}_2)^T \\ \cdots\cdots\cdots\cdots \\ \mathbf{G}_{K-1}^T \boldsymbol{\psi}_{K-1} = \mathbf{H}_K^T \boldsymbol{\psi}_K - (\partial f / \partial \mathbf{u}_{K-1})^T \\ \mathbf{G}_K^T \boldsymbol{\psi}_K = -(\partial f / \partial \mathbf{u}_K)^T \end{cases}. \tag{5.3.91}$$

Comparing the set of discrete adjoint equations (5.3.91) to the original equations (5.3.84), we see that the two sets have identical structures, except that the solution of the former must be in a reverse direction (i.e., from the final time step to the first time step). Once the discrete adjoint states are obtained for all time steps, the gradient of the objective function can be calculated directly from (5.3.90).

The discrete adjoint method can be extended to nonlinear models, in which case the coefficient matrices $\mathbf{G}_k$ of (5.3.84) would depend on the unknown state $\mathbf{u}_k (k = 1, 2, \cdots, K)$. For this case, the first-order variational equation (5.3.87) becomes

$$\mathbf{G}_k' \delta \mathbf{u}_k - \mathbf{H}_k \delta \mathbf{u}_{k-1} + \left(\frac{\partial \mathbf{G}_k}{\partial \boldsymbol{\theta}} \mathbf{u}_k - \frac{\partial \mathbf{H}_k}{\partial \boldsymbol{\theta}} \mathbf{u}_{k-1} - \frac{\partial \mathbf{q}_k}{\partial \boldsymbol{\theta}} \right) \delta \boldsymbol{\theta} = \mathbf{0} \tag{5.3.92}$$
$$(k = 1, 2, \cdots, K)$$

where $\mathbf{G}_k' = \mathbf{G}_k + (\partial \mathbf{G}_k / \partial \mathbf{u}_k) \mathbf{u}_k$ and the shorthand notation used in (5.3.88) is implied for $(\partial \mathbf{G}_k / \partial \mathbf{u}_k)$. Following the above derivation, we can obtain the same results as in the linear case, except that all $\mathbf{G}_k$ are replaced by $\mathbf{G}_k'$ in (5.3.91).

Example 5.10 *The adjoint method for one-dimensional advection–dispersion–reaction Model*
Let us consider the following one-dimensional, advection–dispersion–reaction equation

$$\frac{\partial C}{\partial t} - \frac{\partial}{\partial x}\left(D \frac{\partial C}{\partial x} \right) + V \frac{\partial C}{\partial x} - k_r C = 0, \ 0 \le x \le L, \ 0 \le t \le t_f \tag{5.3.93}$$

subject to initial and boundary conditions

$$C\,|_{t=0} = C_0,\ C\,|_{x=0} = C_B,\ \frac{\partial C}{\partial x}\,|_{x=L} = 0,$$

where $C(x,t)$ is the concentration distribution, D is the dispersion coefficient, V is the flow velocity, k_r is the reaction rate coefficient, and $C_0(x)$ and $C_B(t)$ are given functions. Let us assume that the finite difference grid consists of N nodes with uniform block size Δx and the time domain is discretized into K time steps with uniform time step Δt. When the backward scheme is used, the finite difference equations for the kth time step are given by

$$\begin{cases} \beta_i C_{i,k} + \gamma_i C_{i+1,k} = C_{i,k-1} - \alpha_i C_B,\ i = 1 \\ \alpha_i C_{i-1,k} + \beta_i C_{i,k} + \gamma_i C_{i+1,k} = C_{i,k-1},\ 1 < i \le N-1. \\ (\alpha_i + \gamma_i) C_{i-1,k} + \beta_i C_{i,k} = C_{i,k-1},\ i = N \end{cases} \tag{5.3.94}$$
$$(1 \le k \le K)$$

In the above equations, the first subscript of $C_{i,k}$ denotes node number, and

$$\alpha_i = -\overline{D}_i - \overline{V}_i,\ \beta_i = 1 + 2\overline{D}_i - \overline{k}_{r,i},\ \gamma_i = -\overline{D}_i + \overline{V}_i, \tag{5.3.95}$$

where

$$\overline{V}_i = \frac{V_i \Delta t}{2\Delta x},\quad \overline{D}_i = \frac{D_i \Delta t}{\Delta x^2},\quad \overline{k}_{r,i} = k_{r,i}\Delta t.$$

Suppose that our purpose is to find the gradient of an objective function E with respect to all dimensionless nodal values $\overline{\mathbf{D}} = (\overline{D}_1, \overline{D}_2, \cdots, \overline{D}_N)^T$ of the distributed dispersion coefficient $D(x)$. The weighted least squares criterion E may be dependent on some or all nodal values of the concentration distribution and can be expressed in the following general form

$$E = \sum_{i=1}^{N}\sum_{k=1}^{K} w_{i,k}[C_{i,k} - C_{i,k}^{obs}]^2. \tag{5.3.96}$$

When $C_{i,k}$ is not observed, set the weight $w_{i,k} = 0$. From (5.3.94), the coefficient matrix of the forward equations is time independent and given by

$$\mathbf{G} = \begin{bmatrix} \beta_1 & \gamma_1 & 0 & \cdot & \cdot & \cdot & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & 0 & \cdot & \cdot & 0 \\ 0 & \alpha_3 & \beta_3 & \gamma_3 & 0 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & \alpha_{N-1} & \beta_{N-1} & \gamma_{N-1} \\ 0 & 0 & \cdot & \cdot & \cdot & \alpha_N + \gamma_N & \beta_N \end{bmatrix} \tag{5.3.97}$$

and **H** is a unit matrix. According to (5.3.91), we can find the adjoint equations immediately. The adjoint state associated with the final time step, $\boldsymbol{\Psi}_K$, is solved first from the *k*th adjoint equation, which is formulated using the transpose of **G**

$$\begin{cases} \beta_1 \psi_{1,K} + \alpha_2 \psi_{2,K} = -\partial f / \partial C_{1,K} \\ \gamma_{i-1} \psi_{i-1,K} + \beta_i \psi_{i,K} + \alpha_{i+1} \psi_{i+1,K} = -\partial f / \partial C_{i,K} \\ \qquad (2 \le i \le N-2) \\ \gamma_{N-2} \psi_{N-2,K} + \beta_{N-1} \psi_{N-1,K} + (\alpha_N + \gamma_N) \psi_{N,K} = -\partial f / \partial C_{N-1,K} \\ \gamma_{N-1} \psi_{N-1,K} + \beta_N \psi_{N,K} = -\partial f / \partial C_{N,K} \end{cases} . \tag{5.3.98}$$

Then, solving $\boldsymbol{\Psi}_{K-1}, \boldsymbol{\Psi}_{K-2}, \cdots, \boldsymbol{\Psi}_1$ one by one with the following adjoint equations

For $k = K-1, K-2, \cdots, 1$, do

$$\begin{cases} \beta_1 \psi_{1,k} + \alpha_2 \psi_{2,k} = \psi_{1,k+1} - \dfrac{\partial f}{\partial C_{1,k}} \\ \gamma_{i-1} \psi_{i-1,k} + \beta_i \psi_{i,k} + \alpha_{i+1} \psi_{i+1,k} = \psi_{i,k+1} - \dfrac{\partial f}{\partial C_{i,k}}, (2 \le i \le N-2) \\ \gamma_{N-2} \psi_{N-2,k} + \beta_{N-1} \psi_{N-1,k} + (\alpha_N + \gamma_N) \psi_{N,k} = \psi_{N-1,k+1} - \dfrac{\partial f}{\partial C_{N-1,k}} \\ \gamma_{N-1} \psi_{N-1,k} + \beta_N \psi_{N,k} = \psi_{N,k+1} - \dfrac{\partial f}{\partial C_{N,k}} \end{cases} \tag{5.3.99}$$

From (5.3.95), we get

$$\partial \alpha_i / \partial \bar{D}_i = -1, \ \partial \beta_i / \partial \bar{D}_i = 2, \ \partial \gamma_i / \partial \bar{D}_i = -1$$

$$\partial \alpha_i / \partial \bar{D}_j = \partial \beta_i / \partial \bar{D}_j = \partial \gamma_i / \partial \bar{D}_j = 0, \text{ when } j \ne i.$$

Because $\partial \mathbf{H} / \partial \bar{\mathbf{D}} = \mathbf{0}$ and $\partial f / \partial \bar{\mathbf{D}} = \mathbf{0}$, the gradient $\partial E / \partial \bar{\mathbf{D}}$ can be calculated according to (5.3.90)

$$\begin{cases} \dfrac{\partial E}{\partial \bar{D}_1} = \sum_{k=1}^{K} (2C_{1,k} - C_{2,k})\psi_{1,k} \\ \dfrac{\partial E}{\partial \bar{D}_i} = \sum_{k=1}^{K} (-C_{i-1,k} + 2C_{i,k} - C_{i+1,k})\psi_{i,k}, \quad (2 \le i \le N-2) \\ \dfrac{\partial E}{\partial \bar{D}_{N-1}} = \sum_{k=1}^{K} (-C_{N-2,k} + 2C_{N-1,k} - 2C_{N,k})\psi_{N-1,k} \\ \dfrac{\partial E}{\partial \bar{D}_N} = \sum_{k=1}^{K} (-C_{N-1,k} + 2C_{N,k})\psi_{N,k} . \end{cases} \tag{5.3.100}$$

It has been shown that discrete and continuous forms of the adjoint-state method always give consistent results (Sun and Yeh 1992a). ■

The adjoint-state method has been extensively used in EWR modeling for sensitivity analysis, inverse solution, uncertainty assessment, and experimental design. We will address these applications in the subsequent chapters of the book. Detailed discussions on this method can be found also in Sun (1994), Cacuci et al. (2003), Plessix (2006), Oliver et al. (2008), and Chavent (2009).

5.4 Automatic Differentiation

Automatic differentiation (AD) is a new technique of differentiation that can accurately calculate the derivatives of a function defined by a numerical code. In recent years, AD has been widely used in various science and engineering fields for parameter identification, uncertainty analysis, and optimal design. Detailed discussion on the theory, applications, and implementations of AD can be found in Rall (1981), Berz et al. (1996), and Griewank and Walther (2008).

A source code with inputs $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_m)$ and outputs $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ defines a function $\mathbf{u} = \mathbf{f}(\boldsymbol{\theta})$ $(\mathbb{R}^{\mathbf{m}} \to \mathbb{R}^{\mathbf{n}})$. AD "differentiates" the code directly by converting it to another source code and the latter can be used to evaluate the derivatives of the function. In the most general sense, the source code of an algorithm can be considered as a series of variable declarations and statements

$$\underbrace{s_1, s_2, \cdots, s_m}_{\text{input } \boldsymbol{\theta}}, \underbrace{s_{m+1}, s_{m+2}, \cdots, s_{m+p}}_{\text{intermediate}}, \underbrace{s_{m+p+1}, s_{m+p+2}, \cdots, s_{m+p+n}}_{\text{output } \mathbf{u}}, \tag{5.4.1}$$

where *m, p*, and *n* are the number of input, intermediate, and output variables, respectively. After assigning input variables $s_j = \theta_j$ $(j = 1, 2, \cdots, m)$ and running the code, we can obtain all intermediate and output variables

$$s_k = \phi_k(s_1, s_2, \cdots, s_{k-1}),\ k = m+1, m+2, \cdots, m+p+n, \tag{5.4.2}$$

where ϕ_k is a computable function.

Depending on how the "chain rule" of differentiation is used, AD has two modes: the *forward mode* that differentiates the original code from top to bottom and the *reverse mode* that differentiates the original code from bottom to top. The two modes will be explained in detail below.

5.4.1 The Forward Mode

According to the chain rule of differentiation, for any θ_j $(j = 1, 2, \cdots, m)$, we have

$$\frac{\partial s_k}{\partial \theta_j} = \sum_{\ell<k} \frac{\partial s_k}{\partial s_\ell}\frac{\partial s_\ell}{\partial s_j},\ k = m+1, \cdots, m+p+n. \tag{5.4.3}$$

The forward mode of AD differentiates the primary variables in (5.4.1) one by one, which requires calculation of $\dot{s}_{k,j} = \partial s_k / \partial \theta_j$ for $k = 1, 2, \cdots, m+p+n$ in the forward order,

$$\dot{s}_{1,j} \to \dot{s}_{2,j} \to \cdots \to \dot{s}_{m,j} \to \cdots \to \dot{s}_{m+p,j} \to \cdots \to \dot{s}_{m+p+n,j}. \tag{5.4.4}$$

For $k = 1, 2, \cdots, m$, we have

$$\dot{s}_{k,j} = \partial \theta_k / \partial \theta_j = \begin{cases} 1, & k = j \\ 0, & k \neq j \end{cases}.$$

For $k = m+1, \cdots, m+p+n$, $\dot{s}_{k,j}$ can be calculated by using the chain rule (5.4.3). Because $\ell < k$, all terms $\dot{s}_{\ell,j} = \partial s_\ell / \partial s_j$ in the equation are known by the time when $\dot{s}_{k,j}$ is calculated. All derivatives of u_i with respect to θ_j are thus generated through this forward calculation process,

$$\frac{\partial u_i}{\partial \theta_j} = \dot{s}_{m+p+i,j},\ (i = 1, 2, \cdots, n). \tag{5.4.5}$$

The AD code for the forward mode is constructed by adding a statement $\dot{s}_{k,j}$ after each primary statement s_k

$$s_1, \dot{s}_{1,j}, s_2, \dot{s}_{2,j}, \cdots, s_m, \dot{s}_{m,j}, \cdots, s_{m+p}, \dot{s}_{m+p,j}, \cdots, s_{m+p+n}, \dot{s}_{m+p+n,j}. \tag{5.4.6}$$

In order to calculate the Jacobian $\partial \mathbf{u} / \partial \boldsymbol{\theta}$, we need to run the code *m* times for $j = 1, 2, \cdots, m$.

Example 5.11 *Illustration of the forward mode of AD*
To illustrate how the forward mode of AD works, let us consider a simple function

$$u = f(\theta_1, \theta_2) = \sin(\theta_1 + \theta_2) + \theta_1 \exp(-\theta_2). \tag{5.4.7}$$

The code for calculating u is given by

Variable group	Expression	Results
Input	$s_1 = \theta_1$	
	$s_2 = \theta_2$	
Intermediate	$s_3 = s_1 + s_2$	$\theta_1 + \theta_2$
	$s_4 = \sin(s_3)$	$\sin(\theta_1 + \theta_2)$
	$s_5 = \exp(-s_2)$	$\exp(-\theta_2)$
	$s_6 = s_1 * s_5$	$\theta_1 \exp(-\theta_2)$
	$s_7 = s_4 + s_6$	$\sin(\theta_1 + \theta_2) + \theta_1 \exp(-\theta_2)$
Output	$u = s_7$	

To calculate $\partial u / \partial \theta_1$, the source code generated by the forward mode of AD is

$s_1 = \theta_1$	
$s_2 = \theta_2$	
$\dot{s}_1$	1
$\dot{s}_2$	0
$s_3 = s_1 + s_2$	
$\dot{s}_3 = \dot{s}_1 + \dot{s}_2$	1
$s_4 = \sin(s_3)$	
$\dot{s}_4 = \cos(s_3) * \dot{s}_3$	$\cos(\theta_1 + \theta_2)$
$s_5 = \exp(-s_2)$	
$\dot{s}_5 = -\exp(-s_2) * \dot{s}_2$	0
$s_6 = s_1 * s_5$	

$\dot{s}_6 = \dot{s}_1 * s_5 + s_1 * \dot{s}_5$	$\exp(-\theta_2)$
$s_7 = s_4 + s_6$	
$\dot{s}_7 = \dot{s}_4 + \dot{s}_6$	$\cos(\theta_1 + \theta_2) + \exp(-\theta_2)$
$u = s_7$	
$\dot{u} = \dot{s}_7$	

The code can also be used to calculate $\partial u / \partial \theta_2$ by changing only the inputs $\dot{s}_1 = 1$ to $\dot{s}_1 = 0$ and $\dot{s}_2 = 0$ to $\dot{s}_2 = 1$.

The above example shows that AD can accurately evaluate the derivatives of a function when the function is accurately evaluated by the original code, but the perturbation method can only give approximate answers. Comparing the forward mode of AD with the sensitivity equation method, we see that (i) the two methods require almost the same computational effort; in order to obtain the Jacobian of a function, the former needs to solve the sensitivity equation m times, while the latter needs to run AD code m times, where *m* is the dimension of the input variable; and (ii) the results produced by the two methods have the same accuracy as that of the forward solution because they are solved by the same numerical system. ■

The sensitivity equation method, however, requires making changes to the original code individually according to the type of the model equation, but AD is a general tool of numerical differentiation that can be applied to any codes regardless of the model structure. Therefore, using AD can save the time of developing new codes and avoid human errors. The forward mode of AD is now well developed. Available software packages available can differentiate codes written in Fortran, C++, Matlab, and Python. Detailed information can be found online at http://www.autodiff.org.

5.4.2 The Reverse Mode

The reverse mode (or the adjoint mode) of AD traverses the chain rule from left to right and, as a result, it differentiates the model code in the reverse direction from bottom to top. The reverse mode of AD differentiates the statements of the primary code by calculating $\bar{s}_{k,i} = \partial u_i / \partial s_k$ ($k = m + p + n, m + p + n - 1, \ldots, 2, 1$) one by one in the reverse order

$$\bar{s}_{1,i} \leftarrow \bar{s}_{2,i} \leftarrow \cdots \leftarrow \bar{s}_{m,i} \leftarrow \cdots \leftarrow \bar{s}_{m+p,i} \leftarrow \cdots \leftarrow \bar{s}_{m+p+n,i}. \tag{5.4.8}$$

For $k = m + p + n, m + p + n - 1, \cdots, m + p + 1$,

$$\bar{s}_{k,i} = \partial u_i / \partial s_k = \partial u_i / \partial u_k = \begin{cases} 1, & k = m + p + i \\ 0, & k \neq m + p + i \end{cases}.$$

For $k = m+p, m+p-1, \ldots, 1$, $\bar{s}_{k,i}$ can be calculated one by one by using the chain rule

$$\frac{\partial u_i}{\partial s_k} = \sum_{\ell > k} \frac{\partial u_i}{\partial s_\ell} \frac{\partial s_\ell}{\partial s_k}, \tag{5.4.9}$$

where all $\partial u_i / \partial s_\ell = \bar{s}_{\ell,i}$ $(\ell > k)$ are already calculated by the time when $\bar{s}_{k,i}$ is calculated. All derivatives of u_i with respect to the parameter vector $\boldsymbol{\theta}$ are thus obtained by repeating this process,

$$\frac{\partial u_i}{\partial \theta_k} = \bar{s}_{k,i} (k = 1, 2, \cdots, m). \tag{5.4.10}$$

To calculate the Jacobian $\partial \mathbf{u} / \partial \boldsymbol{\theta}$, we need to run the AD code (5.4.8) n times for $i = 1, 2, \cdots, n$.

Example 5.12 *Illustration of the reverse mode of AD*
Consider the same function used in Example 5.11,

$$u = f(\theta_1, \theta_2) = \sin(\theta_1 + \theta_2) + \theta_1 \exp(-\theta_2). \tag{5.4.11}$$

The code for calculating u is given in the first table of Example 5.11. The reverse mode of AD calculates

$$\begin{aligned}
\bar{s}_{7,7} &= \frac{\partial s_7}{\partial s_7} = 1 \\
\bar{s}_{6,7} &= \frac{\partial s_7}{\partial s_7}\frac{\partial s_7}{\partial s_6} = \bar{s}_{7,7}\frac{\partial s_7}{\partial s_6} = 1 \\
\bar{s}_{5,7} &= \bar{s}_{7,7}\frac{\partial s_7}{\partial s_5} + \bar{s}_{6,7}\frac{\partial s_6}{\partial s_5} = s_1 \\
\bar{s}_{4,7} &= \bar{s}_{7,7}\frac{\partial s_7}{\partial s_4} + \bar{s}_{6,7}\frac{\partial s_6}{\partial s_4} + \bar{s}_{5,7}\frac{\partial s_5}{\partial s_4} = 1 \\
\bar{s}_{3,7} &= \bar{s}_{7,7}\frac{\partial s_7}{\partial s_3} + \bar{s}_{6,7}\frac{\partial s_6}{\partial s_3} + \bar{s}_{5,7}\frac{\partial s_5}{\partial s_3} + \bar{s}_{4,7}\frac{\partial s_4}{\partial s_3} = \cos(s_3) \\
\bar{s}_{2,7} &= \bar{s}_{7,7}\frac{\partial s_7}{\partial s_2} + \bar{s}_{6,7}\frac{\partial s_6}{\partial s_2} + \bar{s}_{5,7}\frac{\partial s_5}{\partial s_2} + \bar{s}_{4,7}\frac{\partial s_4}{\partial s_2} + \bar{s}_{3,7}\frac{\partial s_3}{\partial s_2} \\
&= -s_1 \exp(-s_2) + \cos(s_3) \\
\bar{s}_{1,7} &= \bar{s}_{7,7}\frac{\partial s_7}{\partial s_1} + \bar{s}_{6,7}\frac{\partial s_6}{\partial s_1} + \bar{s}_{5,7}\frac{\partial s_5}{\partial s_1} + \bar{s}_{4,7}\frac{\partial s_4}{\partial s_1} + \bar{s}_{3,7}\frac{\partial s_3}{\partial s_1} = s_5 + \cos(s_3)
\end{aligned}$$

The last two equations produce

$$\begin{aligned} \frac{\partial u}{\partial \theta_1} &= \bar{s}_{1,7} = \exp(-\theta_2) + \cos(\theta_1 + \theta_2) \\ \frac{\partial u}{\partial \theta_2} &= \bar{s}_{2,7} = -\theta_1 \exp(-\theta_2) + \cos(\theta_1 + \theta_2). \end{aligned} \tag{5.4.12}$$

■

From Example 5.12, we see that the values of all primary variables $s_1, s_2, \cdots, s_{m+p+n}$ are required in the reverse order computation. If the original code overwrites all or some of these variables, the AD code must restore them before calculating $\bar{s}_{m+p+n,i}, \bar{s}_{m+p+n-1,i}, \cdots, \bar{s}_{1,i}$. This is a major difficulty associated with programming a reverse mode AD code. For example, when the primary code solves a time-dependent problem by a numerical method, the state variables of all time steps must be stored before they are overwritten or recalculated after they are overwritten. The extra store space or computation effort to be required may become unaffordable. In contrast, the forward mode of AD does not have this problem. As shown in (5.4.6), the value of s_k can be overwritten after $\dot{s}_{k,j}$ is calculated. Different software packages use different strategies to tackle this issue by using either extra computational effort and/or extra storage space. As a result, it is often necessary to modify a reverse mode code generated by an AD tool manually, especially when they are used to process a complicated EWR model. A list of current AD tools using the reverse mode can also be found from the website, http://www.autodiff.org.

Both the forward and reverse modes of AD have been used in the field of EWR modeling. For example, He et al. (2000) used AD as a tool of sensitivity analysis for air pollution modeling, Barhen and Reister (2003) used AD for parameter estimation and uncertainty analysis, Baalousha and Köngeter (2006) used AD for analyzing the risk of groundwater pollution, in which a Fortran code (MCB) used for solving coupled groundwater flow and mass transport problems (Sun 1996) is differentiated, Sambridge et al. (2007) gave an introduction to using AD for solving geophysical inverse problems, and Castaings et al. (2009) used AD for distributed hydrological modeling.

5.5 Applications of Model Differentiation

5.5.1 Numerical Optimization

We showed in Chaps. 2–4 that the problems of finding quasisolutions in the deterministic framework and finding point estimates in the stochastic framework can both be formulated as an optimization problem:

$$\min_{\boldsymbol{\theta}} S(\boldsymbol{\theta}), \ \boldsymbol{\theta} \in \mathbf{P}_{\mathbf{ad}}. \tag{5.5.1}$$

When a gradient-based optimization code is used to solve this problem, the code calls two subroutines after obtaining an updated $\boldsymbol{\theta}$ in each iteration step:

- Subroutine $\boldsymbol{\theta} \to \mathbf{S}(\boldsymbol{\theta})$ for calculating the value of the objective function
- Subroutine $\boldsymbol{\theta} \to \mathbf{g}(\boldsymbol{\theta})$ for calculating the gradient of the objective function

The former incorporates observation data and regularization terms into the forward solution code, and the latter is obtained from the former with a model differentiation method. In this case, we should choose the adjoint-state method because of its effectiveness—all components $\partial S / \partial \theta_j (j = 1, 2, \cdots, m)$ of the gradient can be obtained by solving the adjoint problem (or running the code for the reverse mode of AD) only once, regardless of the dimension m of $\boldsymbol{\theta}$. In contrast, when the sensitivity equation method is used to calculate $\mathbf{g}(\boldsymbol{\theta})$, the sensitivity problem must be solved m times. Examples of using the adjoint-state method for inversion can be found in Chavent et al. (1975), Seinfeld and Chen (1978), Neuman (1980), Sun and Yeh (1985), and Townley and Wilson (1985) for parameter identification; Sun and Yeh (1992b), Neupauer and Wilson (2001), Cirpka and Kitanidis (2000), and Michalak and Kitanidis (2004) for statistical inversion and contaminant transport simulation; Castaings et al. (2009) and Ding and Wang (2012) for the inversion of distributed hydrological models.

In the case of using a Gauss–Newton algorithm to minimize a sum of squares function, we need a subroutine $\boldsymbol{\theta} \to \mathbf{J}_{\mathbf{D}}(\boldsymbol{\theta})$ to calculate the Jacobian matrix in each iteration step. In this case, the sensitivity equation method is more effective because all elements of the matrix in (5.1.3) can be obtained by solving the sensitivity problem (or running the code of the forward mode of AD) m times, regardless the number n of observation data. In contrast, to obtain the Jacobian, the adjoint-state method requires solving the adjoint problem n times. Because we always have $m < n$ for a well-posed inverse problem, the sensitivity equation method is often incorporated into a Gauss–Newton algorithm and used in the inverse solution codes (Hill and Tiedeman 2007; Sun 1994).

5.5.2 Local Sensitivity Analysis

Sensitivity analysis quantifies variations in model output(s) caused by possible variations of model input(s). It can be used to rank the relative importance of different model parameters to the model output (for system control and model reduction), assess the uncertainty of model application on the basis of uncertainty in model parameters (for model uncertainty analysis), and, in the inverse direction, determine the accuracy requirement of model parameters from the accuracy requirement of model application (for model improvement). In fact, sensitivity analysis plays an important role in almost all aspects of model study and has been applied to various fields of science and engineering. Depending on the problem at hand, either a local or global sensitivity analysis needs to be carried out. This section gives only a brief discussion on the derivative-based method of local sensitivity analysis. Global sensitivity analyses and their applications to uncertainty analysis will be considered in Chap. 10.

5.5.2.1 Definition of Local Sensitivity

Let us consider a model $\mathbf{u} = \mathcal{M}(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is an m-dimensional vector of input parameter and $\mathbf{u}$ is an n-dimensional vector of output state. Let $\boldsymbol{\theta}^*$ be a nominal parameter obtained, for example, by inversion. A variation of $\Delta\boldsymbol{\theta}$ in $\boldsymbol{\theta}^*$ will lead to a variation $\Delta\mathbf{u}$ in $\mathbf{u}(\theta^*)$. For a continuously differentiable model, we have the following expression after applying Taylor's expansion,

$$\Delta\mathbf{u} = \mathbf{u}(\boldsymbol{\theta}^* + \Delta\boldsymbol{\theta}) - \mathbf{u}(\boldsymbol{\theta}^*) = \mathbf{J}(\boldsymbol{\theta}^*)\Delta\boldsymbol{\theta} + \text{HOT}, \tag{5.5.2}$$

where $\mathbf{J}(\boldsymbol{\theta}^*)$ is the Jacobian matrix $[\partial\mathbf{u} / \partial\boldsymbol{\theta}]_{n\times m}$ evaluated at θ^*. After omitting the higher-order terms, (5.5.2) gives

$$\Delta\mathbf{u} \approx \mathbf{J}(\boldsymbol{\theta}^*)\Delta\boldsymbol{\theta}. \tag{5.5.3}$$

Equation (5.5.3) implies that (1) the output variation $\Delta\mathbf{u}$ depends not only on the input variation $\Delta\boldsymbol{\theta}$, but also on the nominal parameter $\boldsymbol{\theta}^*$; (2) in a neighborhood of $\boldsymbol{\theta}^*$, $\Delta\mathbf{u}$ is proportional to $\Delta\boldsymbol{\theta}$, with $\mathbf{J}(\boldsymbol{\theta}^*)$ the coefficient of proportion, and (3) the Jacobian matrix represents the local sensitivity of the model output to model input.

As mentioned previously, Jacobian matrix is often called the sensitivity matrix and its elements, all first-order derivatives, the sensitivity coefficients. The derivative $\partial u_i / \partial\theta_j$ measures how sensitive an output component u_i is with respect to an input parameter θ_j when other input parameters are fixed. The *i*th row of the sensitivity matrix consists of sensitivity coefficients of u_i with respect to all parameters

$$\frac{\partial u_i}{\partial\theta_1}, \frac{\partial u_i}{\partial\theta_2}, \dots, \frac{\partial u_i}{\partial\theta_m}. \tag{5.5.4}$$

The *j*-th column of the sensitivity matrix consists of sensitivity coefficients of all model outputs to a single parameter

$$\left(\frac{\partial u_1}{\partial\theta_j}, \frac{\partial u_2}{\partial\theta_j}, \dots, \frac{\partial u_n}{\partial\theta_j}\right)^T. \tag{5.5.5}$$

From (5.5.3), the variation of a single output component u_i is given by

$$\Delta u_i \approx \frac{\partial u_i}{\partial\theta_1}\Delta\theta_1 + \frac{\partial u_i}{\partial\theta_2}\Delta\theta_2 + \dots + \frac{\partial u_i}{\partial\theta_m}\Delta\theta_m. \tag{5.5.6}$$

The contribution of the variation of one input component $\Delta\theta_j$ to Δu_i depends not only on the value of $\left|\partial u_i / \partial\theta_j\right|$, but also on the magnitude of $\Delta\theta_j$, $\left|\Delta\theta_j\right|$. Therefore, we cannot use the sensitivity coefficients in (5.5.4) alone to compare the relative importance of different input components to a model output unless all input components have the same size of variation (which is unlikely because of the difference in physical parameters). By the same token, we cannot compare the relative importance of one input parameter to all output components with the sensitivity coefficients in (5.5.5) unless the model is a single-state model.

In order to compare the sensitivities of different state variables with respect to different parameters, the dimensionless sensitivity of an output component u_i to an input parameter θ_j is defined as

$$S_{i,j} = \frac{\tau_j}{\lambda_i}\frac{\partial u_i}{\partial \theta_j} \quad (i = 1, 2, \cdots, n; j = 1, 2, \cdots, m), \tag{5.5.7}$$

where the coefficient λ_i has the same dimension as $u_i (i = 1, 2, \cdots, n)$, the coefficient τ_j has the same dimension as $\theta_j (j = 1, 2, \cdots, m)$, and all derivatives are evaluated at a nominal input parameter $\boldsymbol{\theta}^*$. There are different methods to set these coefficients, for example, by simply taking $\tau_j = \theta_j^*$ and $\lambda_i = u_i(\boldsymbol{\theta}^*)$. In the statistical framework, we can use the mean of θ_j and u_i (or their standard deviations), respectively, as τ_j and λ_i (Saltelli et al. 2008).

5.5.2.2 Calculation of Local Sensitivity

After the two sets of coefficients $\{\tau_j\}_{j=1}^m$ and $\{\lambda_i\}_{i=1}^n$ are specified, the problem of calculating local sensitivities (5.5.7) turns into the calculation of a sensitivity matrix, which can be done by any method of model differentiation that has been introduced in this chapter. As we have explained previously, for this purpose, the sensitivity equation method is more effective when $m < n$, while the adjoint-state method is more effective when $n < m$. In most practical problems of sensitivity analysis, a model is used to predict the values of only a few variables (model output) and we want to screen the most sensitive factors from a large number of parameters (model input); in this case, the adjoint-state method provides an indispensable tool for sensitivity analysis.

Example 5.14 *Sensitivity of a single-state output to a distributed parameter*
We consider calculation of sensitivity coefficients for a confined groundwater flow problem. Figure 5.2 shows a hypothetical two-dimensional confined aquifer, which is 3000 m long in the *x*-direction and 2000 m wide in the *y*-direction. Boundary sections AB and CD are inflow boundary with constant head equal to 100 m, and all other boundary sections are no-flow boundaries. The initial head is 100 m everywhere. There are three wells located at W_1, W_2, and W_3 with planned pumping rates of 2000, 10000 and 4000 m^3/day, respectively. The transmissivity $T(x, y)$ of the aquifer is not known exactly. We want use a model to find the sensitivity of the lowest head (or the maximum drawdown) in well W_2 with respect to the distributed transmissivity $T(x, y)$.

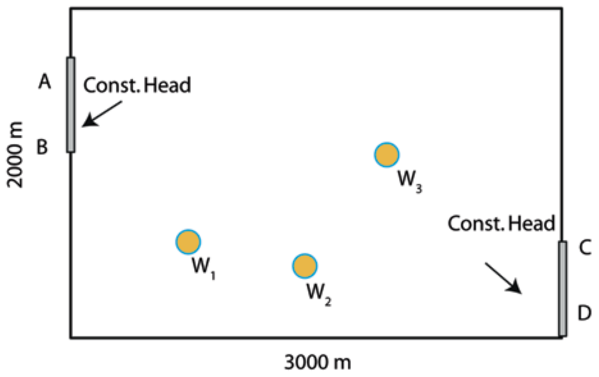


Fig. 5.2 A hypothetical two-dimensional confined aquifer: pumping rates from W_1, W_2, and W_3 are 2000, 10000, and 4000 m^3/day, respectively

Following the derivation given in Example 5.6, we first solve the forward problem (5.3.30) by a finite element code to obtain the head distribution ϕ of each node and its gradient $\nabla\phi$ of each element. Then, using the same code, we solve the adjoint problem (5.3.39) to obtain the adjoint-state distribution ψ of each node and its gradient $\nabla\psi$ in each element by resetting (i) the initial condition to zero, (ii) the given head boundary condition to zero, and (iii) the sink/source term to $\partial f / \partial\phi$. According to (5.3.14), the sink/source term is equal to one at W_2 and zero elsewhere. Denote the value of transmissivity $T(x, y)$ in the *j*th element as T_j, then according to (5.3.44) we have

$$\frac{\partial\phi(W_2)}{\partial T_j} = \int_0^{t_f} (\nabla\phi\cdot\nabla\psi)_j \Omega_j dt, \; (j = 1, 2, \cdots, N_e), \tag{5.5.8}$$

where t_f is the time that the system reaches steady state, Ω_j is the area of the *j*th element, and N_e is the total number of elements. To obtain all of the N_e derivatives, we only need to solve the forward problem and the adjoint problem once. We can use these derivatives as the dimensionless sensitivity for analysis because this problem has one state variable and all inputs are components of the same physical parameter.

Local sensitivity depends on the nominal input parameter $T^*(x, y)$. We assume that it is a realization of a log-normally distributed random field with mean $E(\ln T) = 6.9$ (m^2/day) and nearly homogeneous with variance $\sigma^2 = 0.01$. The distribution of sensitivity $\partial\phi(W_2) / \partial T$ calculated by (5.5.8) is shown in Fig. 5.3. Three major sensitive areas can be identified: around the pumping well W_2 and near the inflow boundary sections AB and CD.

We now consider a case in which the transmissivity field is more heterogeneous with the same mean, but higher variance $\sigma^2 = 2.0$ (shown in Fig. 5.4) is used as the model input. The corresponding sensitivity distribution is shown in Fig. 5.5.

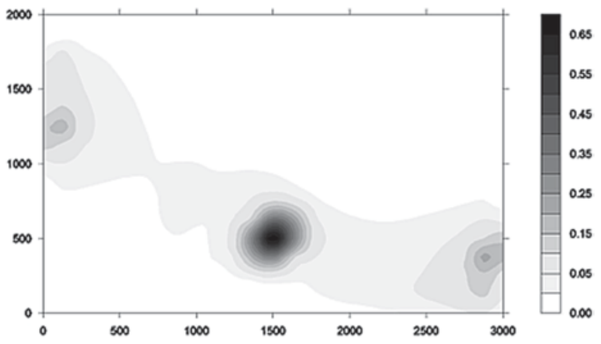


Fig. 5.3 Sensitivity $\partial\phi(W_2)/\partial T$ for the nearly homogeneous nominal transmissivity

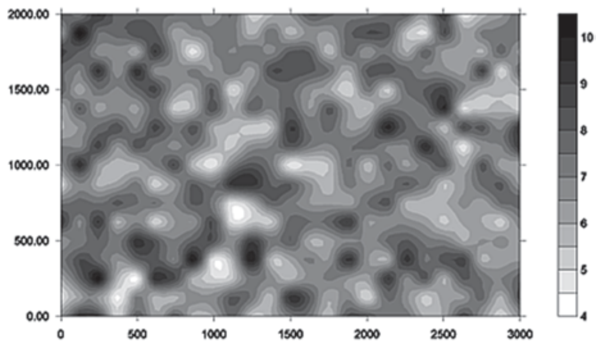


Fig. 5.4 Spatial distribution of the heterogeneous nominal transmissivity field

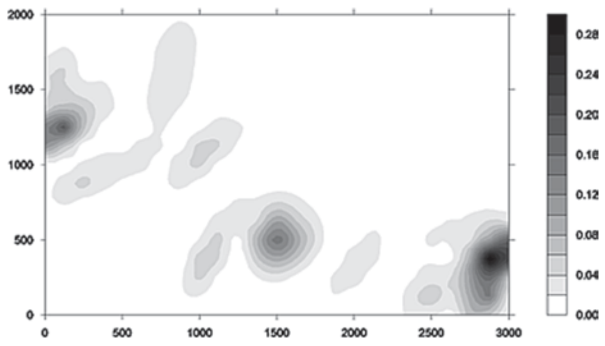


Fig. 5.5 Sensitivity $\partial\phi(W_2)/\partial T$ obtained for the heterogeneous nominal transmissivity field shown in Fig. 5.4

A quick comparison of Figs. 5.3 and 5.5 may suggest that the major sensitive areas of the two cases are the same, and the pattern of sensitivity distribution depends mainly on the pattern of the flow field, instead of the nominal transmissivity. A careful look at Figs. 5.3 and 5.5, however, indicates that the actual sensitivity values of the two cases are quite different. In the homogeneous case, the most sensitive area is around the well and the maximum sensitivity value is 0.68, while in the heterogeneous case, the most sensitive area is near the outflow boundary and the maximum sensitivity value becomes 0.30. Moreover, when the nominal parameter $T^*(x,y)$ is replaced by another realization generated from the same random field, the sensitivity values will change significantly too.

Nevertheless, we can draw a useful conclusion: in order to accurately estimate the drawdown in well W_2, the most important thing is to find the accurate transmissivity values in areas surrounding the well and near the flow boundaries.

5.5.3 Dimensionless Sensitivity Analysis

Sensitivity coefficients can also be used to measure the contribution of observation data to parameter identification. In Sun and Yeh (1990), the *contribution of an observation* $u_{D,i}^{obs}$ to the identification of parameter θ_j is defined by the following dimensionless variable

$$C_{i,j} = \frac{\varepsilon_j}{\eta_i}\left|\frac{\partial u_{D,i}}{\partial \theta_j}\right|, \tag{5.5.9}$$

where $u_{D,i}$ is the model output corresponding to the observation $u_{D,i}^{obs}$, η_i is the upper bound of observation error, and ε_j is the accuracy requirement of the identified parameter. Note that the contribution defined in (5.5.9) depends only on the data collection strategy, rather than their actual observed values. In Chap. 11, we will show how the sufficiency of a design can be quantified by calculating the contributions of data (i.e., data worth) during the design stage.

Example 5.15 *Contributions of observations to the identification of a parameter*
For the same problem considered in the Example 5.14, let us find the sensitivity distribution

$$\frac{\partial \phi_1}{\partial T_{W2}}, \frac{\partial \phi_2}{\partial T_{W2}}, \cdots, \frac{\partial \phi_N}{\partial T_{W2}}, \tag{5.5.10}$$

where T_{W2} is the local transmissivity around the well W_2, and N is the total number of nodes. The perturbation method is an easy way to calculate these derivatives, but the sensitivity equation method (or the forward mode of AD) can produce more accurate results by solving the forward and sensitivity problems once for each.

The contribution of an observation taken at node (i) to the identification of T_{W2} is defined by $\frac{\varepsilon}{\eta}\left|\partial \phi_i / \partial T_{W2}\right| (i = 1, 2, \cdots, N)$. Because $\frac{\varepsilon}{\eta}$ is the same for all nodes, we can simply use the absolute values of the derivatives in (5.5.10) to compare the contributions of observations taken from different locations.

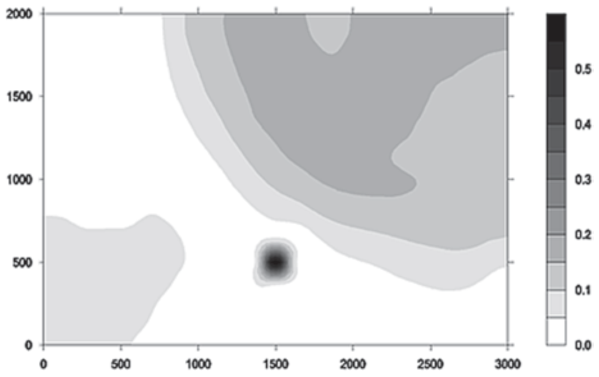


Fig. 5.6 Contribution distribution for the nearly homogeneous case

The nominal $T^*(x, y)$ fields used in Example 5.14 are reused here as the model inputs. Figure 5.6 shows the contribution distribution for the nearly homogeneous case, which suggests that data should be collected around well W_2 to make the largest contribution. In contrast, locations around the two low-flow corners would make less contribution, and areas near the constant-head boundaries would make little contribution.

For the heterogeneous transmissivity shown in Fig. 5.4, the corresponding contribution distribution is given in Fig. 5.7. It has similar pattern as the heterogeneous case, but observations taken from the upper-right low-flow area make more contribution, in addition to the area around well W_2.

This example emphasizes the fact that the sensitivity defined in (5.5.7) and the contribution of an observation defined in (5.5.9) can only be used in the local sense for nonlinear models. When the nominal input parameter has a large uncertainty, the result of local sensitivity analysis based on a fixed input parameter will become unreliable. In this case, a global sensitivity analysis is needed, which will be introduced in Chap. 10.

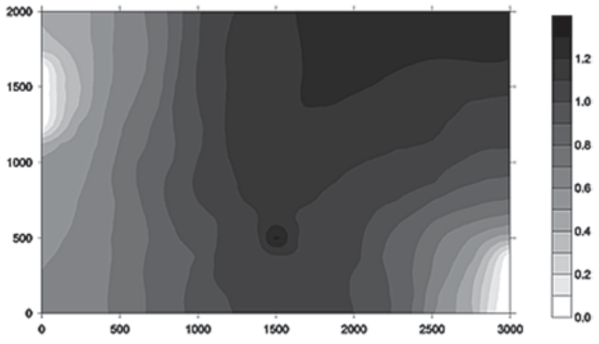


Fig. 5.7 Contribution distribution for the heterogeneous case

5.6 Review Questions

1. Derive the sensitivity equation model for the model given in Example 1.5.
2. Use the forward solution code used in the Review Question 4 of Chap. 1 to find $\partial C \,/\, \partial D$ and $\partial C \,/\, \partial R$ at a set of specified locations and times by the perturbation method and the sensitivity equation method, respectively. Compare the results obtained by these two methods.
3. Complete the four steps of the adjoint-state method described in Sect. 5.3.1 for the following model: $\dfrac{d^2u}{dt^2} + a(u, \boldsymbol{\theta}) = 0,\ 0 \le t \le t_f;\ u\,|_{t=0} = f,\ \dfrac{du}{dt}\,|_{t=0} = g.$
4. Give the details of derivation that leads to Eq. (5.3.29) from Eq. (5.3.25).
5. Show the adjoint operation rules #4, #5, and #6 in Table 5.1.
6. Use the adjoint operation rules to derive the adjoint-state problem for the model given in Example 5.4 with appropriate initial and boundary conditions.
7. Derive the second and the third equations in Eq. (5.3.83).
8. How the contribution of an observation to the identification of a parameter is defined and calculated? How do we evaluate the total contribution of a set of observations to the identification of a parameter and the contribution of an observation to the identification of a set of parameters?
9. In this chapter, model differentiation is used to calculate the gradient of an objective function for inversion and the Jacobian matrix for local sensitivity analysis, which method, the sensitivity equation method (the forward mode of AD) or the adjoint-state method (the reverse mode of AD) should be used in different cases?

Chapter 6
Model Dimension Reduction

In Chap. 2–5, we focused on the CIP of identifying or estimating finite-dimensional parameter vectors, in which it was assumed that the modeled system can be characterized completely and accurately by a set of scalar parameters. Generally speaking, characterization of more complex systems needs more complex models. The complexity of a model is measured by its DOF (i.e., the number of independent parameters to be estimated or the dimension of inversion). A real system, especially a distributed parameter system, may have high or even infinite dimensions. When the dimensionality of a model becomes too high, we will encounter the so-called *curse of dimensionality* (Bellman 1957): (i) the amount of local data needed for identifying the local behavior of the model becomes sparse or even nonexistent, (ii) the number of possible models increases exponentially, and (iii) the information matrix $\mathbf{J}^T\mathbf{J}$ becomes nearly singular and, thus, the inverse solution becomes unstable (i.e., the over-parameterization problem). We learned in Sect. 5.5 that for a single parameter vector to be identifiable, at least one observation that is sensitive to parameter value changes must be available. Thus, in the case of high dimensionality all inversion methods that we have learned become inefficient and the inverse problem becomes unsolvable because of data and computational limitations. As a tradeoff, certain details of a complex system have to be omitted during the construction of its model. When the structure of a model is overly simplified, however, the model may become useless because of its large structure error. Therefore, selection of an appropriate model structure is a challenging task. Nevertheless, it is the key to successful modeling of complex EWR systems.

An appropriate model structure depends not only on the complexity of the modeled system, but also on data availability, data format, and the intended use of the model. In previous chapters, we assumed that the model structure error is absent. Starting from this chapter, we will consider quantification and mitigation of model structure errors. The main purpose of this chapter is to give a comprehensive survey of various parameterization techniques. Section 6.1 describes techniques for parameterizing deterministic functions, including Voronoi diagram, radial basis functions, and local polynomial approximation. It is shown that many of parameterization techniques fall into the form of weighted sum of basis functions. Physical processes

N-Z. Sun, A. Sun, *Model Calibration and Parameter Estimation,*
DOI 10.1007/978-1-4939-2323-6_6

involved in EWR applications are spatially variable and the resulting parameterization is high-dimensional. Section 6.2 introduces methods for parameterizing random fields, including Gaussian random field, Markov random field, variogram analysis, and multipoint statistics. Various kriging algorithms will be described, not only from geospatial interpolation perspective, but also for parameterizing random fields. Section 6.3 describes linear and nonlinear techniques for reducing dimensions of parameters and model states while preserving certain aspects (e.g., covariance or local similarity) of the original parameters.

As we shall see in subsequent chapters, materials introduced in this chapter serve as bases for advanced subjects such as data-driven modeling, data assimilation, uncertainty quantification, and optimal model structure identification.

6.1 Parameterization of a Deterministic Function

6.1.1 Function Approximation for Inversion

The basic concept of parameterization was first mentioned in Sect. 2.1.4, where we introduced several commonly used parameterization methods for spatially distributed parameters. A distributed parameter or a distributed state variable may be a parametric or a nonparametric function of spatial and/or temporal coordinates. A parametric function has finite DOF and, thus, can be determined completely by a number of scalar parameters. For example, Fig. 6.1a illustrates such a parametric curve. A nonparametric function, on the other hand, has infinite DOF, as illustrated in Fig. 6.1b. Real-world examples of nonparametric functions include the hydraulic conductivity of a highly heterogeneous aquifer, the elevation distribution of a mountainous terrain, and the temporal distribution of precipitation over a watershed. It is important to keep in mind that a nonparametric function cannot be completely and accurately recovered by a finite number of measurements or identified by a finite number of data points.

To describe or estimate a nonparametric function $f(\mathbf{x})$, where $\mathbf{x}$ represents spatial and/or temporal coordinates, we first need to parameterize it, a process by which a nonparametric function is approximated by a parametric function $\hat{f}(\mathbf{x})$ with finite

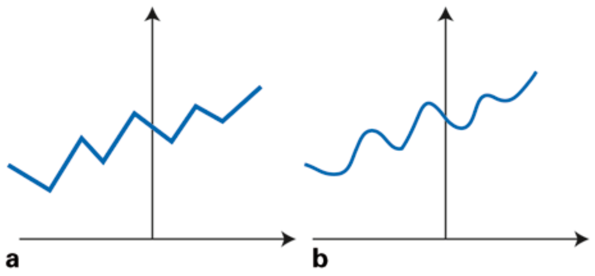


Fig. 6.1 Illustrations of **a** parametric curve and **b** nonparametric curve

DOF. In essence, parameterization transforms a complex physical process or an input–output relationship into a simplified representation, and the function $\hat{f}(\mathbf{x})$ can be seen as a model of $f(\mathbf{x})$ in the general sense. We know that when $f(\mathbf{x})$ is an element of the Hilbert functional space (see Appendix A), it can be expressed by

$$f(\mathbf{x}) = \sum_{j=1}^{\infty} c_j \phi_j(\mathbf{x}), \tag{6.1.1}$$

where $\{\phi_j(\mathbf{x})\}_{j=1}^{\infty}$ is a set of basis functions. When these basis functions are orthogonal to each other, (6.1.1) is called an orthogonal expansion of $f(\mathbf{x})$ and coefficients $\{c_1, c_2, \cdots, c_m, \cdots\}$ are given by the inner product

$$c_j = (f, \phi_j) = \int_{\Omega} f(\mathbf{x}) \phi_j(\mathbf{x}) d\Omega. \tag{6.1.2}$$

The first m terms of expansion (6.1.1) define an approximation function of $f(\mathbf{x})$,

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{m} c_j \phi_j(\mathbf{x}). \tag{6.1.3}$$

Because $\hat{f}(\mathbf{x})$ is completely specified by *m* coefficients $\{c_j\}_{j=1}^{m}$ and corresponding basis functions $\{\phi_j\}_{j=1}^{m}$, it is a parametric function and is called a *parameterization* or *a model* of $f(\mathbf{x})$, where m is called the *dimension of parameterization*. Equation (6.1.3), which was used previously in the zonation method (Sect. 2.1.4), is general and serves as the starting point of many parameterization methods to be introduced later in this chapter, as well as many linear and nonlinear regression methods to be described in Chap. 8. Although **x** mainly includes spatial and temporal coordinates in the context of this chapter, in general it can consist of any set of physical variables (or predictors) needed to capture an input–output relationship. By changing the type of basis functions and/or the dimension of parameterization, we obtain different models of the same function. Therefore, parameterization of a function is always nonunique.

After a set of *m* basis functions $\{\phi_j\}_{j=1}^{m}$ are selected, we can use the measured values of $f(\mathbf{x})$ at *n* points $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ to determine the *m* coefficients in (6.1.3) under the requirement that $\hat{f}(\mathbf{x}_i) \approx f(\mathbf{x}_i)$ for $i = 1, 2, \cdots, n$. Let the measurement vector be $\mathbf{d}_n = [f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_n)]^T$ and the unknown coefficient vector be $\mathbf{c}_m = [c_1, c_2, \cdots, c_m]^T$, we can rewrite (6.1.3) as a system of equations

$$\mathbf{G}\mathbf{c}_m = \mathbf{d}_n, \tag{6.1.4}$$

where **G** is an $n \times m$ matrix with elements $g_{ij} = \phi_j(\mathbf{x}_i)$. The solution of the coefficients $\mathbf{c}_m$ from (6.1.4) is the focus of linear inversion problems discussed extensively under Sect. 2.2. Below we consider three separate cases, namely, interpolation, regression, and dimension reduction.

6.1.1.1 Interpolation

When $n = m$ and matrix $\mathbf{G}$ is nonsingular, solving (6.1.4) is straightforward and the solution is $\mathbf{c}_m = \mathbf{G}^{-1}\mathbf{d}_n$. In this case, we have $\hat{f}(\mathbf{x}_i) = f(\mathbf{x}_i)$ at all measurement points as shown in Fig. 6.2a. In particular, if all basis functions satisfy the condition

$$\phi_j(\mathbf{x}_i) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

we simply have $\mathbf{c}_j = f(\mathbf{x}_j)$ for all $j = 1, 2, \ldots, m$. Such setting is behind the "parameterization by finite-element method" scheme described under Sect. 2.1.4.

6.1.1.2 Regression

When the problem is overdetermined ($n > m$) and $\mathbf{G}^T\mathbf{G}$ is invertible, we can find the least squares solution $\mathbf{c}_m = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{d}_n$ of (6.1.4). In this case, $\hat{f}(\mathbf{x})$ and $f(\mathbf{x})$ may not coincide at measurement points, but the difference between them (L_2-norm) is minimized as illustrated by Fig. 6.2b. For the general case of $n \neq m$, we can use the pseudoinverse solution $\mathbf{c}_m^{\dagger}$ of (6.1.4) to find $\hat{f}(\mathbf{x})$ (see Sect. 2.2). The main difference between interpolation and regression is that the former honors the measurement points exactly, whereas the latter minimizes the discrepancy between a fitted solution and the data.

6.1.1.3 Dimension Reduction

Using the truncated SVD in Sect. 2.2, we may improve the stability of linear inversion problem (6.1.4) by reducing the dimension of parameterization. This can be especially useful when $\hat{f}(\mathbf{x})$ is used to approximate $f(\mathbf{x})$ for inversion.

In a numerical model, a distributed parameter $f(\mathbf{x})$ is automatically parameterized into an N-dimensional vector $\mathbf{f}_N = [f_1, f_2, \cdots, f_N]^T$ after discretization, where N is the number of nodes (or elements), $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ are space coordinates of the nodes, and $f_i = f(\mathbf{x}_i)$ for $i = 1, 2, \cdots, N$. Depending on the actual application, the resulting number of nodes N can be large. Because we rarely have sufficient data to

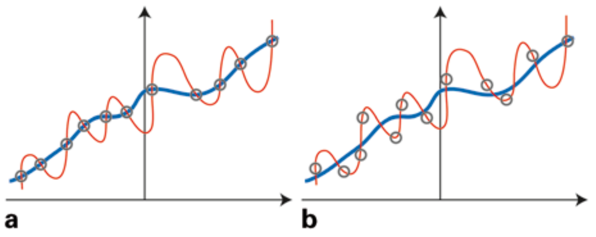


Fig. 6.2 Illustration of **a** interpolation and **b** regression. The thin curve is the actual function $f(\mathbf{x})$, the thick curve is approximation function $\hat{f}(\mathbf{x})$, and open circles represent measurements

identify such a high-dimensional vector, reducing the dimension of $\mathbf{f}_N$ or reparameterizing it into a low-dimensional vector becomes essential. The discretized form of parameterization (6.1.3) can be written as

$$\mathbf{f}_N = \mathbf{G}\mathbf{c}_m \text{ or } \mathbf{c}_m = \mathbf{G}^\dagger \mathbf{f}_N, \tag{6.1.5}$$

in which $\mathbf{G}$ is an $N \times m$ matrix having elements $g_{ij} = \phi_j(\mathbf{x}_i)$, the $m \times N$ matrix $\mathbf{G}^\dagger$ is its pseudoinverse, and $m \ll N$ in most cases. From (6.1.5), we see that a discrete parameterization is a linear transformation that projects a high-dimensional vector to a low-dimensional space for inversion.

6.1.1.4 Parameterization and Inversion

In the above discussion, a parameterization or an approximation of function $f(\mathbf{x})$ is determined directly by a set of measurements of the function itself. If the function is the unknown parameter, its approximation can be determined indirectly by the observations of state variables through inversion. After discretization, the inverse problem of identifying all nodal values of the unknown parameter, $\mathbf{f}_N$, requires minimizing an objective function $S(f_1, f_2, \cdots, f_N)$. After parameterization, the problem of identifying $\mathbf{f}_N$ becomes identification of a low-dimensional vector $\mathbf{c}_m$ by minimizing the same objective function. Using (6.1.5) and the chain rule, we have

$$\frac{\partial S}{\partial \mathbf{c}_m} = \frac{\partial S}{\partial \mathbf{f}_N}\frac{\partial \mathbf{f}_N}{\partial \mathbf{c}_m} = [\nabla S]\mathbf{G}, \tag{6.1.6}$$

where elements of the $N \times N$ matrix $[\nabla S]$ are the gradients of $S(f_1, f_2, \cdots, f_N)$ with respect to all nodes and can be obtained effectively by the adjoint-state method introduced in Chap. 5; and matrix $\mathbf{G}$ is defined in (6.1.5) by basis functions. Similarly, the Jacobian after parameterization is given by

$$\mathbf{J}(\mathbf{c}_m) = \frac{\partial \mathbf{u}_D}{\partial \mathbf{f}_N}\frac{\partial \mathbf{f}_N}{\partial \mathbf{c}_m} = \mathbf{J}(\mathbf{f}_N)\mathbf{G}, \tag{6.1.7}$$

where the Jacobian before parameterization, $\mathbf{J}(\mathbf{f}_N)$, can also be obtained effectively by using the adjoint-state method. The scalar form of (6.1.6) is given by

$$\frac{\partial S}{\partial c_j} = \sum_{i=1}^{N}\frac{\partial S}{\partial f_i}\frac{\partial f_i}{\partial c_j} = \sum_{i=1}^{N}\frac{\partial S}{\partial f_i}\phi_j(\mathbf{x}_i), \; j = 1, 2, \cdots, m. \tag{6.1.8}$$

The scalar form of (6.1.7) can be obtained similarly. Once we know how to calculate gradients $\partial S / \partial \mathbf{c}_m$ and the Jacobian $\mathbf{J}(\mathbf{c}_m)$, a gradient-based optimization method or a Gauss-Newton method described in Chap. 2 can be used to find the low-dimensional parameter vector $\mathbf{c}_m$. Note that the problem of identifying $\mathbf{c}_m$ is much easier than the problem of identifying the unknown high-dimensional vector $\mathbf{f}_N$. After $\mathbf{c}_m$ is identified, the discrete inversion $\mathbf{f}_N$ can be determined easily by (6.1.5). A continuous approximation $\hat{f}(\mathbf{x})$ of the unknown parameter $f(\mathbf{x})$ is then generated from $\mathbf{f}_N$ by interpolation.

The error of parameterization is measured in the parameter space by

$$e_{par} = \left\| f(\mathbf{x}) - \hat{f}(\mathbf{x}) \right\|. \tag{6.1.9}$$

Unfortunately, e_{par} cannot be evaluated directly by using (6.1.9) because $f(\mathbf{x})$ is unknown. Parameterization error is a type of model error because even in the absence of observation errors, we cannot find a $\hat{f}(\mathbf{x})$ such that the model outputs satisfy $u_D(\hat{\theta}) = u_D^{obs}$ exactly. In EWR modeling, model error characterization remains a challenging problem and an active research area. As mentioned before, model error is often the main cause of model failure. In Chap. 7, we will return to this important topic and learn how to reduce the parameterization error.

6.1.2 Interpolation as a Tool for Parameterization

Assume that we have m measured values of a function, $f(\mathbf{x})$, defined on a spatial (and/or temporal) domain $\mathbf{\Omega}$:

$$f_i = f(\mathbf{x}_i) + e_i, \quad i = 1, 2, \cdots, m \tag{6.1.10}$$

where $\mathbf{x}_i$ is called a *data site* or a *basis point* of interpolation, and e_i is the associated measurement error. The error terms are assumed to be zero-mean i.i.d. Gaussian random variables. Interpolation is a method that can estimate the function value $f(\mathbf{x})$ at any unmeasured point using the data in (6.1.10). After interpolation is completed for the entire region $\mathbf{\Omega}$, we obtain a parametric function $\hat{f}(\mathbf{x})$ called an *interpolant* such that at any measured site, $\hat{f}(\mathbf{x})$ is equal to the measured value of $f(\mathbf{x})$; and at any unmeasured site, $f(\mathbf{x})$ is unknown but $\hat{f}(\mathbf{x})$ can be calculated by an interpolation algorithm. The difference between $f(\mathbf{x})$ and $\hat{f}(\mathbf{x})$, or the error of interpolation, depends on (i) variability of the interpolated function, (ii) number and distribution pattern of the data sites, and (iii) the algorithm used for interpolation.

Interpolation produces a parametric representation of an incompletely measured function. It has been used extensively in computer graphics, signal and image processing, geographic information systems, and numerical solution visualization. In the field of EWR, various interpolation methods are used for generating distributed states (e.g., precipitation, temperature, soil moisture, concentration) from their discrete measurements. Knotters et al. (2010) presented a literature review of various interpolation methods used in EWR.

6.1.2.1 Interpolation for Inverse Solution

In the study of inverse problems, the unknown parameter $\theta(\mathbf{x})$ is regarded as the interpolated function and the interpolant $\hat{\theta}(\mathbf{x})$ is used as a parameterization of the unknown parameter. Let us again consider three separate cases:

- Case 1. The unknown parameter is measured at *m* sites in the field. In this case, we can use measurements $\{\theta_i = \theta(\mathbf{x}_i) + \varepsilon_i \mid i = 1, 2, \cdots, m\}$ directly to generate an interpolant $\hat{\theta}(\mathbf{x})$ as the identified parameter. Any structural information relevant to the parameter, such as discontinuity, can be incorporated into the interpolation process. Generally speaking, $\hat{\theta}(\mathbf{x})$ obtained in this way may not be a satisfactory approximation of the true parameter because of the insufficient number of measurements and the measurement error.
- Case 2. A set of state observations $\{\mathbf{u}_D^{obs}\}$ is available, which can be related to the unknown parameter $\theta(\mathbf{x})$ through some simulation model. In this case, we create *m* virtual "data sites" (called the *basis points*). The "measured values" at these points then become the parameter vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \cdots, \theta_m)^T$ to be identified by inverse solution. At the beginning, an initial guess of $\boldsymbol{\theta}$ is used to generate an interpolant $\hat{\theta}(\mathbf{x})$. During the iterative optimization process of inverse solution, $\hat{\theta}(\mathbf{x})$ is updated continuously by the current $\boldsymbol{\theta}$ until the optimum solution is reached. Any prior information on the identified parameter can be incorporated into this process either as regularization terms or as constraints. The number of virtual data sites, *m*, is now the dimension of parameterization.
- Case 3. Both state observations and measurements of the parameter itself are available. In this case, we have two approaches to use these data. The first approach is to combine *m* hypothetical measurements $\theta_1, \theta_2, \cdots, \theta_m$ and *k* real measurements $\theta_{m+1}, \theta_{m+2}, \cdots, \theta_{m+k}$ of the parameter to form a full set of data. In the optimization process of inversion, all of these data are used to generate the interpolant $\hat{\theta}(\mathbf{x})$, but only the *m* hypothetical measurements $\theta_1, \theta_2, \cdots, \theta_m$ are updated and optimized. The second approach treats $\theta_{m+1}, \theta_{m+2}, \cdots, \theta_{m+k}$ as soft rather than hard data. They are identified together with $\theta_1, \theta_2, \cdots, \theta_m$ in the optimization process, but are subject to a range constraint, which requires that their identified values be not far from the measured values. This approach can decrease the fitting residual at the cost of increasing the dimension of parameterization.

From the above discussion we see that any interpolation method is also a parameterization method, but not all interpolation methods are appropriate for inverse solution. When an interpolation method is chosen as a parameterization method for inversion, it should satisfy the following additional criteria:

1. The method should be able to handle scattered basis points so that irregular parameter structures can be characterized effectively and flexibly.
2. The method should be computationally effective because the parameterized model $\hat{\theta}(\mathbf{x})$ must be recalculated or updated in each iteration during the optimization process of inversion.

3. The method should allow incorporation of prior information pertaining to the interpolant.
4. The method should be flexible for modifying the shape of the interpolant in order to decrease the model error and fitting residuals. This can be done by introducing *shape parameters* into the basis functions, as will be shown in the following subsections.

We already described two simple parameterization methods, the zonation method and the finite element interpolation method, in Chap. 2. In the rest of this section, we will introduce more interpolation and approximation methods commonly used for parameterization.

6.1.3 Inverse Distance Weighting

Inverse distance weighting (IDW) is a popular weighted average method because of its simplicity. Let $\{f_i\}_{i=1}^m$ in (6.1.10) be a set of data collected from data sites $\{\mathbf{x}_i\}_{i=1}^m$, and $\{w_i(\mathbf{x})\}_{i=1}^m$ be a set of weighting functions satisfying the unbiasedness condition $\sum_{i=1}^m w_i(\mathbf{x}) = 1$. The following interpolant is called a *weighted average*

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{m} f_i w_i(\mathbf{x}). \tag{6.1.11}$$

Let $\|\mathbf{x} - \mathbf{x}_i\|$ be the Euclidian distance between the point of estimation $\mathbf{x}$ and a data site $\mathbf{x}_i$. In the simplest IDW method, the weighting functions of IDW are given by (Shepard 1968):

$$w_i(\mathbf{x}) = \frac{W_i(\mathbf{x})}{W}, \tag{6.1.12}$$

where

$$W_i(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_i\|^{-\beta}, \quad W = \sum_{i=1}^{m} W_i(\mathbf{x}).$$

The power $\beta > 0$ is called a shape parameter because it determines the shape of the interpolant $\hat{f}(\mathbf{x})$. When the value of β is large, the weight of nearby data sites will dominant those located farther away; conversely, when the value of β is small, farther away data sites become more influential. As a result, the simple global weight method (6.1.12) often produces undesirable results.

Figure 6.3 illustrates the $\hat{f}(\mathbf{x})$ surface obtained by IDW using the same data sites but different β values. It shows that when $\beta = 4$ the surface features (e.g., bumps) tend to be more spatially extensive than in the case of $\beta = 1$, indicating

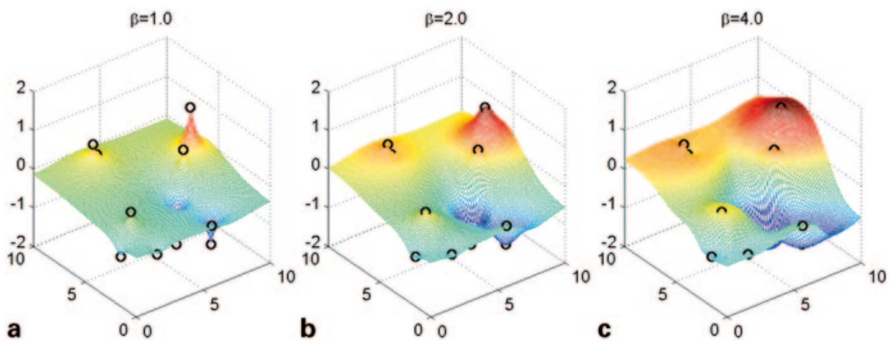


Fig. 6.3 Approximation function surface $\hat{f}(\mathbf{x})$ obtained by IDW for **a** $\beta = 1$, **b** $\beta = 2$, and **c** $\beta = 4$, respectively. Open circles correspond to randomly generated data sites, which are the same for all three cases

the more dominating effect of nearby data sites on interpolation. The default value of β used in the IDW module of many commercial software packages is 2. In the following, we will describe several techniques that can improve the accuracy of the global weight IDW.

6.1.3.1 Location-Dependent Shape Parameters

Unless data sites are uniformly distributed, different locations should use different β values. When a site is surrounded by clustered data sites, small β value should be used to give nearly equal weights to these measurements. On the other hand, when a site is surrounded by relatively dispersed data sites, large β value should be used to give large weights to nearby measurements. This suggests that the IDW weighting functions in (6.1.12) should be changed such that the shape parameter β is location dependent

$$w_i(\mathbf{x}, \beta_i) = \frac{W_i(\mathbf{x}, \beta_i)}{W}, \tag{6.1.13}$$

where

$$W_i(\mathbf{x}, \beta_i) = \left\| \mathbf{x} - \mathbf{x}_i \right\|^{-\beta_i}, \quad W = \sum_{i=1}^{m} W_i(\mathbf{x}, \beta_i).$$

Lu and Wong (2008) developed an adaptive algorithm that can determine the β values for different locations based on the pattern of data sites. In the next chapter, we will see that these shape parameters are additional tuning parameters to be identified when IDW is used as a parameterization method for inversion.

6.1.3.2 Localization

Localization means that when determining the value of $\hat{f}(\mathbf{x})$, only the m_w nearest measurements of $\mathbf{x}$ in a local neighborhood are considered, instead of all *m* measurements of $\mathbf{x}$. The undesirable effect of faraway measurements (e.g., spurious correlations) can thus be avoided. To make the IDW localized, for each interpolation location $\mathbf{x}$ we first find m_w nearest data sites surrounding it and then calculate the interpolant by

$$\hat{f}(\mathbf{x}) = \sum_{i \in \mathcal{I}_{\mathbf{x}}} f_i w_i(\mathbf{x}), \tag{6.1.14}$$

where $\mathcal{I}_{\mathbf{x}}$ is the index set of the m_w nearest data sites of $\mathbf{x}$, the weight functions are defined by

$$w_i(\mathbf{x}) = \frac{W_i(\mathbf{x})}{W}, \quad W_i(\mathbf{x}) = \left[\frac{1}{\|\mathbf{x} - \mathbf{x}_i\|} - \frac{1}{R_{\mathbf{x}}}\right]^{\beta}, \quad W = \sum_{i \in \mathcal{I}_{\mathbf{x}}} w_i(\mathbf{x}), \tag{6.1.15}$$

and $R_{\mathbf{x}} = \max_{i \in \mathcal{I}_{\mathbf{x}}} \|\mathbf{x} - \mathbf{x}_i\|$ is the largest distance from $\mathbf{x}$ to its m_w nearest data sites.

6.1.3.3 Regression

For each data site $\mathbf{x}_i$, we find its m_q nearest data sites and then use a low-order (constant, linear, quadratic) polynomial $Q_i(\mathbf{x})$ to fit the $m_q + 1$ measured values by least squares regression, with the constraint that $|Q_i(\mathbf{x}_i) - f_i|$ is less than an estimated upper bound of the measurement error. The function $Q_i(\mathbf{x})$ can be seen as an approximation of the interpolant $f(\mathbf{x})$ in the local neighborhood of $\mathbf{x}_i$. For example, when linear approximation is used for a 2D problem, we have a plane surface

$$Q_i(x, y) = a_i + b_i x + c_i y,$$

which has three unknown coefficients. When bilinear approximation is used, we have a bilinear surface

$$Q_i(x, y) = a_i + b_i x + c_i y + d_i xy,$$

in which the number of unknown coefficients is increased to 4. Similarly, for biquadratic surface we have

$$Q_i(x, y) = a_i + b_i x + c_i y + d_i xy + e_i x^2 + f_i y^2,$$

which has six coefficients. To determine $Q_i(\mathbf{x})$ by regression, m_q should be larger than the number of unknown coefficients. Using $Q_i(\mathbf{x})$ to replace f_i in algorithm (6.1.14) can improve accuracy of the localized IDW (Renka 1988).

Localization and regression are commonly used techniques for improving the accuracy of an interpolation or approximation method. We will use them repeatedly in this and subsequent chapters.

6.1.4 Voronoi Diagram and Delaunay Tessellation

So far, we have not addressed how all surrounding sites of a data site are found when a localized interpolation method is used. This problem is not easy to answer when the number of measurement points is large and their distribution is irregular. In this section, we introduce *Voronoi diagram*, a classical method for spatial tessellation that provides a natural and effective way to solve this problem.

Voronoi diagram is a long-standing topic in mathematics. It was first systematically studied by German mathematician Johann Peter Gustav Lejeune Dirichlet (1805–1859) and Russian mathematician Georgy Feodosevich Voronoy (1868–1908). Along with the development of computational technology, however, Voronoi diagram has become an important study area of computational geometry and found interesting applications in many fields. Voronoi diagram was rediscovered in many fields. For example, it is referred to as the Thiessen method for calculating areal averages in surface hydrology for a long time (Sambridge et al. 1995).

Let $\mathcal{P} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ be a set of *m* distinct points in a bounded or unbounded region $\mathbf{\Omega}$. These points are called *generators of a Voronoi diagram*. The *Voronoi cell* of a generator $\mathbf{x}_i$, denoted by $V_{\mathcal{P}}(\mathbf{x}_i)$, is defined as a region surrounding $\mathbf{x}_i$ such that all points in $V_{\mathcal{P}}(\mathbf{x}_i)$ are closer to $\mathbf{x}_i$ than to any other generator of the set, viz.

$$V_{\mathcal{P}}(\mathbf{x}_i) = \left\{\mathbf{x} \middle| \left\|\mathbf{x} - \mathbf{x}_i\right\| \le \left\|\mathbf{x} - \mathbf{x}_j\right\|, \text{ for all } \mathbf{x}_j \in \mathcal{P},\ \mathbf{x} \in \Omega\right\}. \tag{6.1.16}$$

It is natural to define $V_{\mathcal{P}}(\mathbf{x}_i)$ as the neighborhood region of a generator $\mathbf{x}_i$. In 2D, Voronoi cell is a polygon and in 3D it is a polyhedron. Voronoi cells of all generators collectively form a tessellation of $\mathbf{\Omega}$, which is called the *Voronoi diagram* of the set $\mathcal{P}$. Figure 6.4a shows an example of Voronoi diagram for random generators, in which all Voronoi cells are colored.

By connecting all generators whose Voronoi cells have the common boundaries, we obtain another tessellation of $\mathbf{\Omega}$ which is called the *Delaunay tessellation*. Voronoi diagram and Delaunay tessellation are dual, meaning when one is known the other is completely determined. Figure 6.4b shows a Delaunay triangulation that is the dual of the Voronoi diagram in Fig. 6.4a. Delaunay tessellation is often used to generate meshes for FEM simulations.

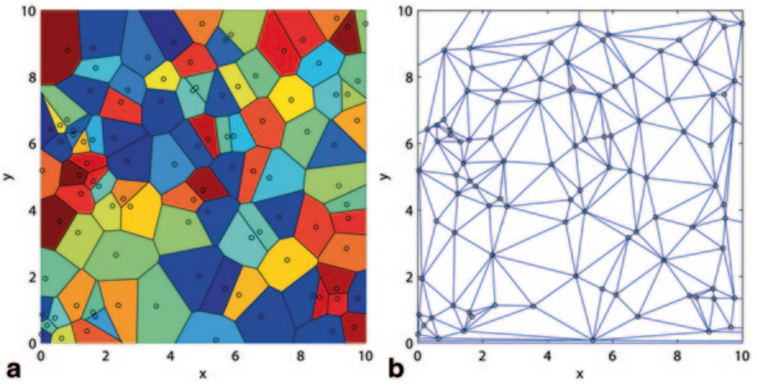


Fig. 6.4 Illustrations of **a** Voronoi diagram and **b** Delaunay tessellation, where the Voronoi generators (open circles) are generated randomly

Several algorithms have been developed for generating the Voronoi diagram corresponding to a given set of generators in 2D and higher dimensions (De Berg 2000; Okabe 2000). For example, the Fortune's sweep line algorithm for 2D (Fortune 1987) and gift wrapping method for 3D (Sugihara 1994). We can also create the Delaunay tessellation for the given generators first and then obtain the dual Voronoi diagram (Barber et al. 1996). In-depth discussion on Voronoi diagram and relevant topics can be found in Okabe (2000). More information can be found, for example, from the webpage of Computational Geometry Algorithms Library at http://www.cgal.org.

A number of commercial and open-source programs provide routines for generating Voronoi diagrams. For example, Fig. 6.4 was generated using Matlab functions `voronoin` and `Delaunay`. SciPy provides a similar routine, `scipy.spatial.Voronoi`. The computational geometry library underlying both Matlab and SciPy implementations is Qhull, which implements a Quickhull algorithm for computing the convex hull (i.e., convex envelope of a set of points in the Euclidean plane). More details can be found at http://www.qhull.org.

After this brief introduction on Voronoi diagram, we now return to our main subject, namely, using Voronoi diagrams for nearest-neighbor interpolation. In this case, the data sites (or the basis points of interpolation), $\{\mathbf{x}_i\}_{i=1}^m$, play the role of generators. In the following, we introduce three interpolation methods based on the Voronoi diagram.

6.1.4.1 Nearest-Neighbor Interpolation

Nearest-neighbor interpolation can be represented in a general form by

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{m} f_i \phi_i(\mathbf{x}, \mathbf{v}), \tag{6.1.17}$$

where $\{\phi_i(\mathbf{x},\mathbf{v})\}_{i=1}^m$ is set of basis functions which, in turn, depend on shape parameters $\mathbf{v}=\{\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_m\}$. If we define

$$\phi_i(\mathbf{x},\mathbf{v}) = \begin{cases} 1, & \text{when } \mathbf{x}\in V_{\mathcal{P}}(\mathbf{x}_i) \\ 0, & \text{otherwise} \end{cases}$$

we will have $\hat{f}(\mathbf{x}) = f_i$ for $\mathbf{x}\in V_{\mathcal{P}}(\mathbf{x}_i)$, and the interpolant is a piecewise constant function in this case. Therefore, we can use Voronoi diagram to generate a zonation structure, in which each zone delineates the nearest neighbor of a data site.

The IDW can also be seen as a nearest-neighbor interpolation method if the basis functions in (6.1.17) are defined in such a way that

$$\phi_i(\mathbf{x},\ \mathbf{v}) = \begin{cases} w_i(\mathbf{x},\beta_i), & \text{when } i\neq j \\ 1-\sum\limits_{i\in\mathcal{I},i\neq j} w_i(\mathbf{x},\beta_i), & \text{otherwise} \end{cases},\ \mathbf{x}\in V_{\mathcal{P}}(\mathbf{x}_j) \tag{6.1.18}$$

where $\{w_i(\mathbf{x},\beta_i)\mid i=1,2,\cdots,m\}$ are weighting functions of IDW defined in (6.1.13) and $\mathcal{I}$ is the integer set $\{1,2,\cdots,m\}$. In this case, the shape parameter $\mathbf{v}$ contains not only the locations of data sites, but also the power parameters $\boldsymbol{\beta}=\{\beta_1,\beta_2,\cdots,\beta_m\}$.

Sun and Yeh (1985) presented an interpolation method that introduces additional shape parameters to the basis functions in (6.1.18) to make the nearest-neighbor interpolation more flexible for representing different types of functions. In their method, a multiplier $\sigma_i(\mathbf{x},\alpha_i)$ is applied to $w_i(\mathbf{x},\beta_i)$ such that (6.1.18) is replaced by

$$\phi_i(\mathbf{x},\ \mathbf{v}) = \begin{cases} \sigma_i(\mathbf{x},\alpha_i)w_i(\mathbf{x},\beta_i), & \text{when } i\neq j, \\ 1-\sum\limits_{i\in\mathcal{I},i\neq j} \sigma_i(\mathbf{x},\alpha_i)w_i(\mathbf{x},\beta_i), & \text{otherwise,} \end{cases}\ \mathbf{x}\in V_{\mathcal{P}}(\mathbf{x}_j) \tag{6.1.19}$$

where $\sigma_i(\mathbf{x},\alpha_i)$ is a user-defined function that contains additional shape parameters $\boldsymbol{\alpha}=\{\alpha_1,\alpha_2,\dots,\alpha_m\}$. If we define

$$\sigma_i(\mathbf{x},\alpha_i) = (d_{j,i}-d_{\mathbf{x},i})/d_{j,i}^{\alpha_i},\ (i\neq j),$$

where $d_{j,i}$ is the distance between sites $\mathbf{x}_i$ and $\mathbf{x}_j$, and $d_{\mathbf{x},i}$ is the distance between point $\mathbf{x}$ and site $\mathbf{x}_i$, then the interpolant (6.1.17) becomes a piecewise function when $\alpha_i>2$, and a continuous function when all $\alpha_i=1$ and $\beta_i>10$. Between the two extreme cases, other function shapes can be obtained by adjusting these shape parameters. Moreover, the method can also produce reasonable extrapolation results when the value of $\sigma_i(\mathbf{x},\alpha_i)$ becomes negative (see Sun and Yeh (1985) and Sun (1994) for details). The set of shape parameters now becomes

$\boldsymbol{\nu} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$. In Tsai and Yeh (2004), the IDW weighting functions in (6.1.19) were replaced by general weighting functions, and functions $\sigma_i(\mathbf{x}, \alpha_i)$ in (6.1.19) were simply replaced by $0 \le \beta_i(\mathbf{x}) \le 1$.

Nearest-neighbor interpolation represents a useful parameterization method for inverse solution according to the criteria presented at the end of Sect. 6.1.2: (i) the generators (i.e., basis points) can accommodate arbitrary spatial patterns; (ii) $\hat{f}(\mathbf{x})$ can be obtained with minimal computational effort; (iii) prior information on parameter structure can be incorporated into the determination of tessellation (or zonation) pattern, as well as the values of shape parameters; and (iv) the tessellation pattern can be optimized by simply modifying the locations of generators. As we will show in Chap. 7, the flexibility offered by the nearest-neighbor parameterization greatly facilitates the optimization of model structures.

6.1.4.2 Interpolation with Delaunay Tessellation

Delaunay tessellation generates a spatial discretization mesh for FEM when the locations of nodes are assigned. For interpolation, nodes play the role of data sites. Using the finite element basis functions, for any point within an element, the interpolant value at the point can be obtained from the nodal values of the element. Therefore, interpolation with Delaunay tessellation is the same as the finite element interpolation introduced in Chap. 2. The only difference here is that the finite element mesh is automatically generated from the node distribution. In the next chapter, we will take advantage of this feature during the model structure identification process to optimize the pattern of parameterization or to generate meshes for multiscale inversion.

6.1.4.3 Natural Neighbor Interpolation

For any given site $\mathbf{x} \in \Omega$, Voronoi diagram can help us find its surrounding data sites for local interpolation. A data site is called a *natural neighbor* of another data site if their Voronoi cells share a common boundary. Thus, when $\mathbf{x} = \mathbf{x}_i$ is a data site, we can find all its natural neighbors by drawing the Voronoi diagram. Figure 6.5a shows such an example, in which the open circle is a data site and all its 7 natural neighbors, labeled by small hexagons, were found by drawing a Voronoi diagram. When $\mathbf{x}$ is not a data site, we can use it as an additional generator to redraw the Voronoi diagram and then find its natural neighbors. As an example, in Fig. 6.5b, the cross symbol represents a nondata site and the shaded area is the new Voronoi cell corresponding to it, which we will denote by $V_{\mathcal{P}+}(\mathbf{x})$; again the natural neighbors of $\mathbf{x}$ are labeled by small hexagons. Comparing Fig. 6.5b to 6.5a, we see that the newly formed cell $V_{\mathcal{P}+}(\mathbf{x})$ is created by "grabbing" a piece of each of its natural neighbors. The overlap between $V_{\mathcal{P}+}(\mathbf{x})$ and its neighbors, as we will show below, can be used to define weights.

After all natural neighbors of $\mathbf{x}$ are found, three local interpolation methods are often used to obtain an interpolant:

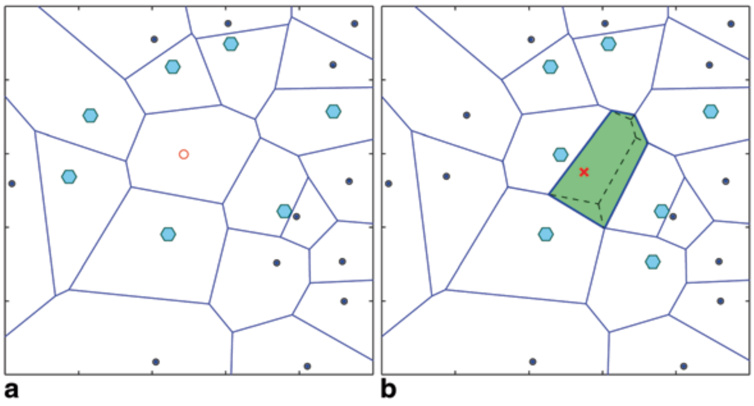


Fig.6.5 Natural neighbors (hexagons) of a **a** data site $\mathbf{x}_i$ (*open circle*), and **b** nondata site $\mathbf{x}$ (*cross*), where the shaded area in *green* is $V_{\mathcal{P}+}(\mathbf{x})$, and the common areas between original data site and its natural neighbors are labeled by *dashed lines* within the shaded area

1. *Distance weighting*, which uses the localized IDW algorithm (6.1.14) and (6.1.15), as described in Sect. 6.1.3.
2. *Area weighting* (Sibson 1981), in which the weighting function in (6.1.11) is defined by

$$w_i(\mathbf{x}) = \frac{\kappa(\mathbf{x}, \mathbf{x}_i)}{\kappa(\mathbf{x})}, \tag{6.1.20}$$

where $\kappa(\mathbf{x}, \mathbf{x}_i)$ is the common area between $V_{\mathcal{P}}(\mathbf{x}_i)$ and $V_{\mathcal{P}+}(\mathbf{x})$ (e.g., areas with dashed borders in the shaded cell of Fig. 6.5b), and $\kappa(\mathbf{x})$ is the whole area of $V_{\mathcal{P}+}(\mathbf{x})$ (i.e., the shaded cell in Fig. 6.5b). Obviously, $w_i(x)$ satisfies the unbiasedness condition. Weighting function (6.1.20) is often used as the interpolation function in mesh-free FEM (Alfaro et al. 2007).

3. *Regression*. Using a low-order polynomial in the local area to fit measurements of the natural neighbors as described in Sect. 6.1.3.

When natural neighbor interpolation is used as a parameterization method for inverse solution, we prefer to use the localized IDW method (i.e., distance weighting) because of its simplicity and effectiveness.

6.1.5 Approximation with Radial Basis Functions

Radial basis function (RBF) has become an important tool in the past decade in many subject areas such as surface reconstruction, object representation, density estimation, nonparametric modeling, model reduction, and mesh-free methods

(Buhmann 2003). It is also a useful tool for structure and parameter identification because of its simplicity, effectiveness, and ability to handle scattered data.

The form of RBF-based interpolation is expressed also by (6.1.3),

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{m} c_i \phi(r_i, \mathbf{v}), \tag{6.1.21}$$

where $\mathbf{v}$ represents shape parameters, $r_i = \|\mathbf{x} - \mathbf{x}_i\|$ is the separation distance between a point $\mathbf{x}$ and a data site $\mathbf{x}_i$, and all basis functions are determined by a single radial basis function $\phi(r, \mathbf{v})$ such that $\phi_i(\mathbf{x}, \mathbf{v}) = \phi(r_i, \mathbf{v})$. The most commonly used radial basis functions include:

- *Gaussian RBF*

$$\phi(r, \varepsilon) = \exp(-\varepsilon^2 r^2),$$

where the shape parameter ε is inversely proportional to the standard deviation of the Gaussian distribution. Increasing the value of ε narrows shape of the RBF and, thus, makes the interpolation more localized. The Gaussian RBF is also known as Gaussian kernel in kernel-based methods (see Chap. 8).

- *Generalized multiquadric RBF*

$$\phi(r, \varepsilon, p) = (1 + \varepsilon^2 r^2)^p,$$

where ε and p are two shape parameters. When $p = 1/2$, it is simply called the multiquadric RBF, and when $p = -1/2$, it is called the inverse multiquadric RBF. In the special case when $p < 0$ and ε is large, the IDW is obtained.

- *Thin-plate spline RBF:* $\phi(r) = r^2 \log r$.

Because the condition

$$\phi_j(\mathbf{x}_i) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

is not satisfied by RBF, we do not have $c_i = f(\mathbf{x}_i)$ $(i = 1, 2, \cdots, m)$. Instead, these coefficients must be estimated by solving the system of equations (6.1.4), where the matrix **G** for RBF is asymmetric and usually positive definite, but may become conditionally positive definite for the thin-plate spline RBF. In the latter case, we can add a low-order polynomial $p(x)$ to the right-hand side of (6.1.21) to ensure the uniqueness of the interpolant (Buhmann 2003). When RBF is used as a parameterization method for inverse solution, it is important to make it localized and keep m small.

In Chap. 7, RBF will be used in the level set method for inversion. In Chap. 8, RBF will be used to develop a neural network model.

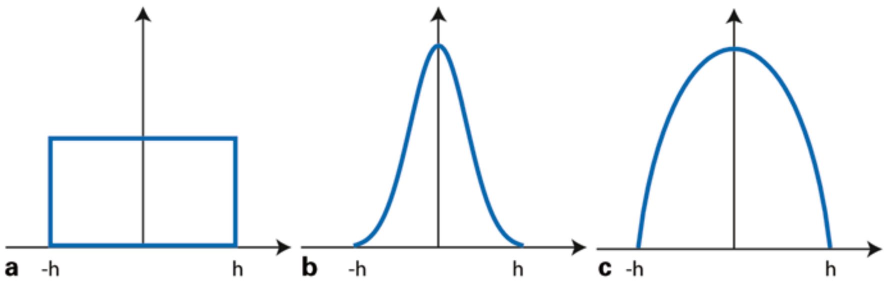


Fig. 6.6 Commonly used localization window functions **a** rectangular, **b** Gaussian, and **c** quadric, where h is half-width of the window

6.1.6 Local Polynomial Approximation

Local polynomial approximation (LPA) can be used to find the natural neighbors of an arbitrary point (see Sect. 6.1.3). By the same token, LPA can also be used to define a window function $w(r,h)$, which is a localized RBF, to generate a local approximation of the interpolated function. A window function specifies a local neighborhood that satisfies the condition

$$w(r,h) = 0, \quad \text{when } r > h, \tag{6.1.22}$$

where the half-width h is a shape parameter. Commonly used window functions are rectangular, Gaussian, and quadric, as shown in Fig. 6.6.

Within a window, LPA uses a low-order (constant to cubic) polynomial $\hat{f}(\mathbf{x},\mathbf{c})$ as an approximation of $f(\mathbf{x})$, and the coefficient vector $\mathbf{c}$ of the polynomial is determined by regression using measurements of $f(\mathbf{x})$ that fall within the window. Let the data sites within the window be $\{\mathbf{x}_1^w, \mathbf{x}_2^w, \cdots, \mathbf{x}_s^w\}$, $r_i^w = \left\|\mathbf{x} - \mathbf{x}_i^w\right\|$, and $f_i^w = f(\mathbf{x}_i^w)$, the coefficient vector $\mathbf{c}$ can be obtained by minimizing the following weighted least-squares objective function

$$S(\mathbf{c}) = \sum_{i=1}^{s} w(r_i^w, h)\left(f_i^w - \hat{f}(\mathbf{x},\mathbf{c})\right)^2. \tag{6.1.23}$$

In LCA, the window size h controls the number of data sites in a window and, thus, the accuracy of approximation.

Intersection of confidence intervals (ICI) is a method that can adaptively determine an appropriate window size for a given point $\mathbf{x}$. ICI searches for the largest window possible such that measurements falling in the window can be fit well by the polynomial with a predetermined order. An exposition of the LPA/ICI method and its applications can be found in Katkovnik et al. (2006). Compared with spline, LPA can produce piecewise constant, continuous, and smooth interpolants using scattered data.

We have introduced several commonly used methods for parameterization. There are, of course, other useful parameterization methods suitable for different cases, for example, polynomial approximation (Phillips 2003), discrete cosine transform (Jafarpour and McLaughlin 2008), and wavelet transformation (Liu 1993) The latter is especially useful for multiresolution inversion (Bhark et al. 2011; Efendiev and Hou 2009).

The truncated expansion (6.1.3) is expressed as a linear combination of a set of linear or nonlinear basis functions. This is the most popular, although not the only form of parameterization. Nonlinear parameterization, in which the approximate function $\hat{f}(\mathbf{c}, \mathbf{x})$ is nonlinear with respect to its parameters $\mathbf{c}$, maybe useful when the criteria presented in Sect. 6.1.2 are satisfied.

6.2 Parameterization of Random Fields

In Chap. 4, the unknown parameter (e.g., hydraulic conductivity) is regarded as a realization of a random field, specified by its joint PDF over a finite set of points. We assumed that the prior PDF of the unknown parameter is known and then obtained its posterior distribution through Bayesian inference. This is an example of *statistical parameterization*, for which the main goal is modeling or parameterizing a *random function*, which is an indexed collection of random variables (see Appendix B). When the index is time, the random function is commonly referred as a stochastic process. When the index is spatial coordinates, the random function is often called a random space function or, simply, random field. In this chapter, we shall use stochastic processes and random fields interchangeably. The selection of an appropriate statistical parameterization is determined by sample statistics. In general, three types of statistics can be calculated during the data exploration stage: univariate, bivariate (two-point), and multivariate (multipoint) statistics. Univariate analysis is used to calculate single-point sample statistics (e.g., mean, variance, and histogram), whereas the two-point and multipoint statistics parameterize the covariance, and the spatial pattern of distributed attributes. Because of data limitations, most statistical parameterization techniques are limited to characterization of low-order statistical moments in practice. In this section, we introduce several statistical parameterization methods commonly used in statistical inversion problems.

6.2.1 Gaussian Random Field

Let $\theta(\mathbf{x})$ be a random field defined in a temporal or a spatial region Ω. As a distributed function, its dimension (DOF) may be infinite. If we consider a finite set of m points in the region, $\mathcal{P} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$, we will have an m-dimensional random vector

$$\boldsymbol{\theta} = (\theta_1, \theta_2, \cdots, \theta_m)^T, \quad \text{where } \theta_i = \theta(\mathbf{x}_i), \quad i = 1, 2, \cdots, m. \tag{6.2.1}$$

In spatial statistics, the elements in set $\mathcal{P}$ may represent (i) point-level data (e.g., groundwater levels), (ii) lattice data, and (iii) point process data (Cressie 1993). Statistical parameterization is about modeling the joint PDF of $\boldsymbol{\theta}$.

Gaussian random field (GRF) is one of the most popular statistical parameterization methods for parameterizing a random field, partly because it has a number of well-known analytical properties. A GRF is fully specified by its mean and covariance and its joint PDF is given by

$$p(\boldsymbol{\theta}) = (2\pi)^{-m/2} \left|\mathbf{C}\right|^{-1/2} \exp\left(-\frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\mu})^T \mathbf{C}^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu})\right), \tag{6.2.2}$$

where m is the dimensionality of $\boldsymbol{\theta}$, $\boldsymbol{\mu}$ is its mean, and $\mathbf{C}$ is its covariance matrix. The quadratic quantity appearing the exponent of (6.2.2) defines a distance

$$\mathbf{d}^2 \triangleq (\boldsymbol{\theta}-\boldsymbol{\mu})^T \mathbf{C}^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu}),$$

in which $\mathbf{d}$ is known as the *Mahalanobis distance*. The joint PDF of GRF will be constant on surfaces of the same $\mathbf{d}$.

If a nondiagonal element of $\mathbf{C}$, $\mathrm{cov}(\theta_i, \theta_j)$, is nonzero, it means the value of $\theta(\mathbf{x})$ at the two sites $\mathbf{x}_i$ and $\mathbf{x}_j$ are correlated. In general, a Gaussian model describes a globally correlated structure. For a Gaussian distribution to be well defined, it is necessary for the covariance matrix $\mathbf{C}$ to be positive definite and symmetric. The covariance matrix $\mathbf{C}$ of a full Gaussian model will have $m(m+1)/2$ independent parameters which, when combined with the m independent parameters in $\boldsymbol{\mu}$, give rise to a total of $m(m+3)/2$ independent parameters (i.e., the number of DOF). When the dimension m is large, the computational demand associated with inverting $\mathbf{C}$ and estimation of the hyperparameters becomes prohibitive. Therefore, simplifications must be made to the structure of $\mathbf{C}$ when m is large, for example, by assuming statistical stationarity:

- A random field is said to be *strongly stationary*, if its joint PDF is constant in the definition region.
- A random field is said to be *second-order stationary*, if (i) its mean is constant, and (ii) its covariance depends only on the distance and direction separating any two locations. For any increment $\mathbf{r}$, we have

$$E[\theta(\mathbf{x}+\mathbf{r}) - \theta(\mathbf{x})] = 0 \text{ and } Cov[\theta(\mathbf{x}+\mathbf{r}), \theta(\mathbf{x})] = C(\mathbf{r}). \tag{6.2.3}$$

For an isotropic field, $C(\mathbf{r})$ reduces to $C(r)$, where $r = \left\|\mathbf{r}\right\|$ is the length of $\mathbf{r}$. In this case, the mean function is a constant and the covariance function depends only on a single scalar parameter. For a random vector defined in (6.2.1), Eq. (6.2.3) can be rewritten as

$$E[\theta(\mathbf{x}_i) - \theta(\mathbf{x}_j)] = 0 \text{ and } Cov[\theta(\mathbf{x}_i), \theta(\mathbf{x}_j)] = C(r), \tag{6.2.4}$$

where $r = \left\| \mathbf{x}_i - \mathbf{x}_j \right\|$ under the isotropic assumption.

Let $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ be two Gaussian random vectors. The joint distribution of $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ is

$$p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \begin{pmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{pmatrix} \right), \tag{6.2.5}$$

in which $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are means of $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, and $\mathbf{C}_{11}$, $\mathbf{C}_{22}$, $\mathbf{C}_{12}$, $\mathbf{C}_{21}$ are block partitions of the covariance matrix of $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)^T$. The conditional distribution of $\boldsymbol{\theta}_2$ given $\boldsymbol{\theta}_1$ is a Gaussian

$$p(\boldsymbol{\theta}_2 | \boldsymbol{\theta}_1) \sim \mathcal{N}(\boldsymbol{\mu}_2 + \mathbf{C}_{21}\mathbf{C}_{11}^{-1}(\boldsymbol{\theta}_1 - \boldsymbol{\mu}_1), \mathbf{C}_{22} - \mathbf{C}_{21}\mathbf{C}_{11}^{-1}\mathbf{C}_{12}). \tag{6.2.6}$$

Thus, (6.2.5) and (6.2.6) indicate that if a joint distribution is Gaussian, the conditional distribution is also a Gaussian. A well-known example of the GRF is the hydraulic conductivity of a heterogeneous aquifer, which is commonly modeled as a random field with log-normal distribution (Freeze 1975). The selection of log-normal distribution is supported by the work of Hoeksema and Kitanidis (1985), who analyzed data from about 20 aquifers in the US.

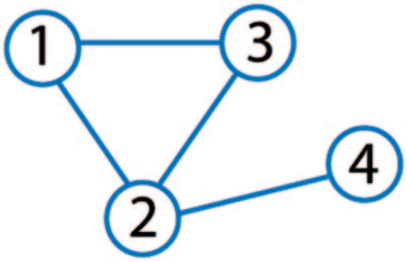
The generalization of the GRF discussed here to multivariate random variables is Gaussian processes (GP), which has gained popularity in the machine learning field in recent decades. We will introduce a GP-based regression algorithm in Sect. 8.4.

6.2.2 *Markov Random Field*

Markov random field (MRF) is another commonly used statistical parameterization technique. It is simple in structure and computationally efficient. Unlike GRF that describes a global correlation structure, MRF describes local correlation structures on a lattice. To explain this, let us first introduce the concept of undirected graph. An *undirected graph* consists of a set of nodes and a set of edges that connect these nodes. Finite difference and finite element grids (regular and irregular lattices) are all examples of undirected graphs. In an undirected graph, if two nodes *i* and *j* are connected by an edge, then node *i* is said to be a neighbor of node *j* and vice versa. Let us denote the neighboring relationship of any two nodes by $i \sim j$. All neighbors of a node form a local structure associated with that node, and the number of neighbors is typically much smaller than the total number of nodes in a graph.

To illustrate, Fig. 6.7 shows a simple undirected graph with four nodes and four edges. Thus, the neighbors of node 1 are nodes 2 and 3, and node 4 only has node 2 as its neighbor. We can use all sites of $\mathcal{P} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ to generate a Voronoi diagram and then generate a Delaunay tessellation of the region. In such case, the natural neighbors of each site form a local structure of that site.

Fig. 6.7 An undirected graph with four nodes and four edges.



Let us consider an m-dimensional random vector $\boldsymbol{\theta}$ defined on lattice $\mathcal{P}$. An MRF model assumes that the component θ_i is correlated to θ_j only if $\mathbf{x}_j$ is a neighbor of $\mathbf{x}_i$. In other words, the distribution of an MRF model is localized. For example, the Gaussian MRF (GMRF) is a localized Gaussian model defined by

$$p(\boldsymbol{\theta}) \propto \exp\left[-\frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\nu})^T \mathbf{Q}(\boldsymbol{\theta}-\boldsymbol{\nu})\right], \tag{6.2.7}$$

where $\mathbf{Q}$, which is the inverse of a covariance matrix, is called the *precision matrix*; and $\boldsymbol{\nu}$ is the so-called location parameter of $\boldsymbol{\theta}$. Because the off-diagonal elements of $\mathbf{Q}$ is not zero if and only if $i \sim j$, $\mathbf{Q}$ is often a highly sparse matrix. Thus, computations involving (6.2.7) are generally more efficient than using the GRF in (6.2.2).

MRF is a flexible parameterization method. By choosing different local distributions, we can obtain different MRF models for characterizing discontinuous, continuous, and smooth parameters. For detailed discussions on MRF and its applications, the reader may refer to the monograph of Rue and Held (2005).

For inverse solution, the following *pairwise-interaction* MRF is often used (Besag et al. 1995)

$$p(\boldsymbol{\theta}) \propto \exp\left\{-\sum_{j\sim i} w_{ij}\Phi\left(\gamma(\theta_i-\theta_j)\right)\right\}, \tag{6.2.8}$$

where γ is a scale parameter; $\Phi(u)=\Phi(-u)$ is a symmetric function; the summation is over all node pairs that are neighbors; and w_{ij} are weights that control the spatial dependence structure of sites. The summation in the exponent of (6.2.8) is called energy function, for which a physical interpretation is that low-energy states are the more likely states. Recall that a similar energy function is used in the simulated annealing algorithm described in Sect. 4.3.4.

When $\Phi = |u|$ is used and all weights are set equal, (6.2.8) becomes a stochastic version of the median filter often used in image analysis (Besag et al. 1995):

$$p(\boldsymbol{\theta}) \propto \gamma \exp\left(-\frac{1}{2}\sum_{j\sim i} w_{ij}\left|\theta_i-\theta_j\right|\right),$$

where w_{ij} are weights. When $\Phi = \frac{1}{2}u^2$ is used, (6.2.8) becomes a special GMRF model

$$p(\boldsymbol{\theta}) \propto \gamma^{m/2} \exp\left(-\frac{1}{2}\gamma\boldsymbol{\theta}^T\mathbf{W}\boldsymbol{\theta}\right), \tag{6.2.9}$$

in which the weight matrix $\mathbf{W}$ plays the role of precision matrix. A possible definition of the matrix is

$$w_{ij} = \begin{cases} n_i, & \text{if } i = j \\ -1, & \text{if } i \sim j \\ 0, & \text{otherwise} \end{cases}, \tag{6.2.10}$$

where n_i is the number of neighbors of site i. In this case, the MRF depends only on a single parameter γ. A GMRF defined using (6.2.10) is called a local planar model, which follows a Gaussian distribution with mean and covariance given as (Besag et al. 1995)

$$p(\theta_i | \theta_{-i}, \gamma) \propto \mathcal{N}\left(\frac{1}{n_i}\sum_{j\sim i}\theta_j, \frac{1}{n_i\gamma}\right), \tag{6.2.11}$$

where $p(\theta_i | \theta_{-i}, \gamma)$ denotes the conditional distribution of θ_i given the rest of the nodes, θ_{-i}, in its neighborhood.

Because of its simple structure and computational efficiency, the MRF is often used as a prior distribution in MCMC. When the precision matrix in a GMRF (6.2.7) is not full rank, the GMRF is called an *intrinsic GMRF* (Rue and Held 2005). An example is given in (6.2.10), in which the precision matrix **W** is sparse. Thus, (6.2.9) is an intrinsic GMRF, which is extensively used as a prior distribution in the estimation of locally correlated random fields. In EWR applications, MRF has been used to model, for example, porous media permeability (Lee et al. 2002; Ferreira and Lee 2007) and contaminant concentration field (Wang and Zabaras 2006).

Example 6.1 *MRF on 2D Lattice*
Figure 6.8 shows the local neighborhood of a data site on a regular 2D lattice, which includes the eight shaded cells surrounding it. It is also common to include only the four nearest cells (labeled with circles) as neighbors. As mentioned in the text, different patterns of spatial association can be accommodated by using different neighboring systems and the lattices need not be regular.

Example 6.2 *Application of GMRF*
Let us assume that the prior of a random field $\theta(\mathbf{x})$ can be modeled using the GMRF (6.2.9) and the likelihood function is Gaussian. Then the posterior PDF is also Gaussian and can be expressed as (Rue and Held 2005)

$$p(\boldsymbol{\theta} | \mathbf{u}^{obs}, \gamma) \propto \frac{\gamma^{m/2}}{\left|\mathbf{V}_D\right|^{1/2}} \exp\left\{-\frac{\mathbf{e}^T\mathbf{V}_D^{-1}\mathbf{e}}{2}\right\} \exp\left\{-\frac{1}{2}\gamma\boldsymbol{\theta}^{\mathrm{T}}\mathbf{W}\boldsymbol{\theta}\right\}, \tag{6.2.12}$$

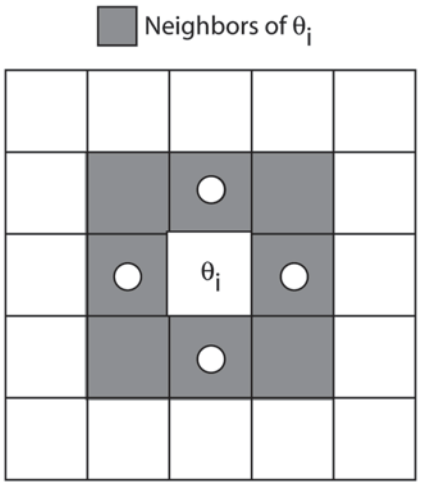


Fig. 6.8 A data site and its 3×3 local neighbor system (*shaded area*) on a 2D regular lattice grid; alternatively, the neighbors of θ_i may only include its four closest neighbors (*open circles*)

where the first and second exponential terms are the likelihood function and the prior, respectively; $\mathbf{e}$ is an error vector between observations $\mathbf{u}^{obs}$ and model outputs, and $\mathbf{V}_D$ is the corresponding error covariance matrix. The scale parameter γ is treated as a hyperparameter, for which the prior PDF is often assumed to follow a Gamma distribution,

$$p(\gamma \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \gamma^{\alpha-1} \exp^{-\beta\gamma}, \tag{6.2.13}$$

where α and β are shape parameters. Using the Bayes theorem, we arrive at the following posterior distribution

$$p(\boldsymbol{\theta}, \gamma \big| \mathbf{u}^{obs}) \propto \frac{\gamma^{m/2}}{\left|\mathbf{V}_D\right|^{1/2}} \exp\left\{-\frac{\mathbf{e}^T \mathbf{V}_D^{-1} \mathbf{e}}{2}\right\} \exp\left\{-\frac{1}{2}\gamma \boldsymbol{\theta}^T \mathbf{W} \boldsymbol{\theta}\right\} p(\gamma). \tag{6.2.14}$$

The posterior PDF in (6.2.14) can be estimated using MCMC.

6.2.3 Variogram

Variogram is a type of two-point statistics used to parameterize the variability of a random field. It was originally introduced in mining geostatistics for estimating a spatial variable from its scattered measurements (Matheron 1962). We will see in the next section that variogram is at the center of classical geostatistical methods (i.e., kriging). Variograms have been covered in a large body of geostatistical literature for different disciplines (Atkinson 2010; Caers 2005; Deutsch and Journel 1998; Diggle and Ribeiro 2007; Goovaerts 1997; Isaaks and Srivastava 1989; Cressie 1993; Rubin 2003).

6.2.3.1 Definition

Let $\{\theta(\mathbf{x}) \mid \mathbf{x} \in \Omega\}$ be a random field and $(\mathbf{x}_i, \mathbf{x}_j)$ be a pair of points in $\boldsymbol{\Omega}$. We will call $\mathbf{r} = \mathbf{x}_j - \mathbf{x}_i$ the separation (or lag) vector, $\mathbf{x}_j$ the tail and $\mathbf{x}_i$ the head points, respectively, of the pair. At these two points, $\theta(\mathbf{x}_i)$ and $\theta(\mathbf{x}_j)$ are two random variables. The variogram of the random field associated with this pair is defined by

$$2\gamma(\mathbf{x}_i, \mathbf{x}_j) = Var\left(\theta(\mathbf{x}_i) - \theta(\mathbf{x}_j)\right), \tag{6.2.15}$$

where $Var(\cdot)$ represents the variance of a random variable, and $\gamma(\mathbf{x}_i, \mathbf{x}_j)$ is called the semivariogram (note: a subscript θ will be added to γ to specify its associated random field when other random fields are considered at the same time). Variogram measures the statistical difference of a random field at two different locations. In general, it depends on both $\mathbf{x}_i$ and $\mathbf{x}_j$, but for a second-order stationary random field defined by (6.2.3), the semivariogram may be simplified to $\gamma(\mathbf{r})$. In this case, the two statistics, $\gamma(\mathbf{r})$ and $C(\mathbf{r})$, are related by

$$\gamma(\mathbf{r}) = \frac{1}{2}[Var(\mathbf{x}_i) + Var(\mathbf{x}_j) - 2C(\mathbf{x}_i, \mathbf{x}_j)] = C(0) - C(\mathbf{r}), \tag{6.2.16}$$

where $C(0) = \sigma^2$ is the constant variance. Figure 6.9 plots $\gamma(\mathbf{r})$ and $C(\mathbf{r})$ as functions of the separation distance $r = \|\mathbf{r}\|$.

From Fig. 6.9, we see that covariance and variogram are two complementary concepts: the former measures correlation (similarity), while the latter measures the variability (dissimilarity). When the distance between two locations increases, the similarity of the random field at the two sites tends to decrease, while their dissimilarity increases. Unlike the covariance function, however, variogram has two advantages. First, variogram can be calculated without knowing the mean; second, when the random field is not second-order stationary, the variance of the field could become unbounded, in which case $C(\mathbf{r})$ is undefined but $\gamma(\mathbf{r})$ still exists. Let us define some weaker stationarity used in geostatistics:

- A random field is said to be *increment stationary* if its semivariogram depends only on the distance and direction separating any two locations, viz.

$$\gamma(\mathbf{r}) = \frac{1}{2}Var[\theta(\mathbf{x} + \mathbf{r}) - \theta(\mathbf{x})]. \tag{6.2.17}$$

- Further, a random field is said to be *intrinsically stationary* if it has a constant mean, in addition to satisfying (6.2.17). Thus, second-order stationarity is a special class of intrinsic stationarity.

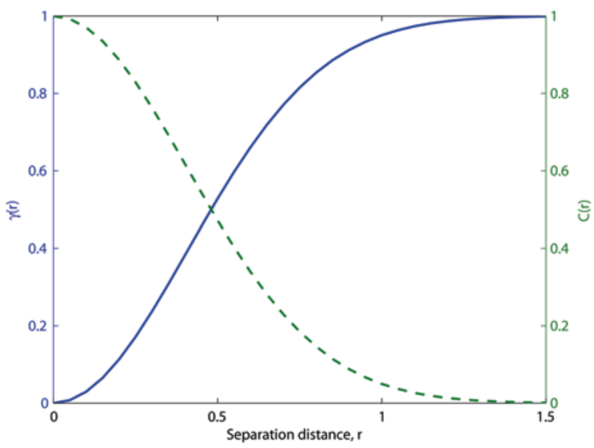


Fig. 6.9 Semivariogram (*left axis*) and covariance (*right axis*) are complementary functions

For an intrinsically stationary random field, $C(\mathbf{r})$ may not exist, but $\gamma(\mathbf{r})$ is always available. With $\gamma(\mathbf{r})$, the structure of a random field can be characterized by a smaller number of statistical parameters.

6.2.3.2 Sample Variogram

In practice, variograms are estimated from sample data. Assume that we have samples of $\theta(\mathbf{x})$ taken from n data sites. The total number of possible site pairs is $n(n+1)/2$. Let $n(\mathbf{r})$ be the number of all pairs of data sites that have similar lags, $\mathbf{r} + \Delta\mathbf{r}$, where $\Delta\mathbf{r}$ is a tolerance interval, and let $z_{h,k}$ and $z_{t,k}$ be the sampled values at the head and tail points of the kth pair, respectively, then a *sample* or *experimental semivariogram* is calculated as a function of $\mathbf{r}$

$$\hat{\gamma}(\mathbf{r}) = \frac{1}{2n(\mathbf{r})} \sum_{k=1}^{n(\mathbf{r})} (z_{h,k} - z_{t,k})^2. \tag{6.2.18}$$

Similarly, the sample covariance function is defined as

$$\hat{C}(\mathbf{r}) = \frac{1}{n(\mathbf{r})} \sum_{k=1}^{n(\mathbf{r})} (z_{h,k} - \hat{\mu}_h)(z_{t,k} - \hat{\mu}_t), \tag{6.2.19}$$

where $\hat{\mu}_h$ and $\hat{\mu}_t$ are sample means of $\{z_{h,k}\}$ and $\{z_{t,k}\}$, respectively. Comparing (6.2.18) and (6.2.19), we can clearly see that $\hat{\gamma}(\mathbf{r})$ is easier to calculate than $\hat{C}(\mathbf{r})$ because the former does not involve sample means. Note that $z_{h,k}$ and $z_{t,k}$ can be measurements of different types of physical variables, in which case (6.2.18) gives the so-called cross-variogram.

6.2.3.3 Variogram Models

Once an experimental semivariogram is obtained, we can fit a parametric variogram model to it. In fact, this is a required step for obtaining kriging estimates. For isotropic random fields, variogram models are commonly characterized by three parameters:

- *Nugget effect* (υ), which represents uncorrelated small-scale variability. When r is small but the sample variogram $\hat{\gamma}(r)$ is not close to zero, the nugget effect parameter should be considered. Its effect is to add a discontinuity to $\gamma(r)$ at $r = 0$.
- *Sill,* which is the upper bound of $\gamma(r)$. In the absence of nugget, sill is equal to the variance.
- *Range* (l)*,* which represents the large-scale correlation. It is defined as the minimum separation distance that $\gamma(r)$ becomes equal or nearly equal to the sill (more than 95% of the sill). In other words, when the distance between two points is larger than l, the random field at these two points can be effectively treated as uncorrelated.

Note that not all variograms have sill and range. In the following, we list several most commonly used variogram models in the geostatistics literature. When the sample variogram shows the existence of a sill, we can choose one of the following basic models:

- *Exponential model:*

$$\gamma(r) = \sigma^2[1 - \exp(-3h)], \text{ where } h = r / l \tag{6.2.20}$$

- *Gaussian model:*

$$\gamma(r) = \sigma^2[1 - \exp(-3h^2)], \text{ where } h = r / l \tag{6.2.21}$$

- *Spherical model:*

$$\gamma(r) = \begin{cases} \sigma^2\left[\dfrac{3}{2}h - \dfrac{1}{2}h^3\right], & \text{when } h = \dfrac{r}{l} \le 1 \\ \sigma^2, & \text{when } h = \dfrac{r}{l} > 1 \end{cases} \tag{6.2.22}$$

When the sample variogram shows no sill and the random field is increment stationary, we can use the *power law model*

$$\gamma(r) = ar^{\omega}, a > 0,\ 0 < \omega < 2 \tag{6.2.23}$$

When $\omega > 1$, the increments are positively correlated, leading to smooth random fields; whereas when $\omega < 1$, the increments are negatively correlated, leading to random fields with rugged appearance (Rubin 2003). A demonstration of the power law model was given by Gelhar (1986), who showed that the experimental variogram of log-transformed hydraulic conductivity increases as the scale of observation increases.

When the sample variogram shows evidence of small-scale variability, we need to use the nugget effect parameter to account for it. For example, the exponential model having a nugget effect can be written as

$$\gamma(r) = \begin{cases} 0, & \text{when } r = 0 \\ \upsilon + (\sigma^2 - \upsilon)[1 - \exp(-3h)], & \text{when } h = \dfrac{r}{l} > 0 \end{cases} \tag{6.2.24}$$

As a matter of fact, new variograms can be generated by combining the basic variogram models described in the above.

For statistical anisotropic random fields, the sample variogram will have different correlation ranges along different directions. For example, in geological settings, the range in the vertical direction is often significantly shorter than that in the horizontal direction. In this case, we may simply replace $h = r / l$ in the fundamental models (except for the power model) by

$$h = \sqrt{(r_x / l_x)^2 + (r_y / l_y)^2 + (r_z / l_z)^2}, \tag{6.2.25}$$

where (r_x, r_y, r_z) and (l_x, l_y, l_z) are components of separation vector **r** and ranges along the three coordinate directions, respectively.

Figure 6.10 compares three variogram models. The exponential model exhibits stronger persistence in correlation than the other two. The Gaussian model, on the other hand, produces smooth spatial variations. In practice, the selection of an appropriate model is not easy because of insufficient data and unavoidable measurement errors. To a large extent, it is determined by the modeler based on his/her subjective judgment. Once a variogram model is selected, the variogram parameters, such as $\upsilon, \sigma^2, l_1, l_2, l_3, a$ and ω, can be estimated by solving a least-squares problem to fit the model output with the sampling variogram. Many geostatistical software packages, such as GSLIB (Deutsch and Journel 1998) and GSTAT (Pebesma 2004), offer the capability to fit the variogram parameters automatically for user-selected variogram models.

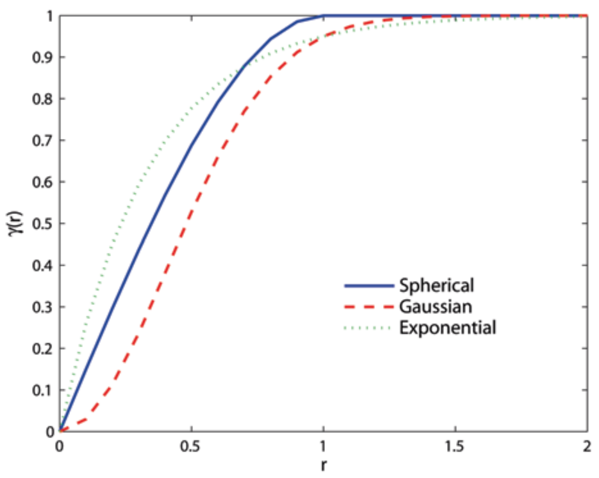


Fig. 6.10 Basic variogram models. The nugget effect is set to 0, and sill and range values are both set to 1.0

6.2.3.4 Variogram Model for Categorical Variables

So far, we have focused on variograms of continuous random variables. Many attributes, such as rock facies or land use/land cover types, are categorical in nature. For such variables, we may define an indicator spatial random function (Goovaerts 1997; Journel and Isaaks 1984)

$$I_k(\mathbf{x}) = \begin{cases} 1, & \text{if } \theta(\mathbf{x}) = k \\ 0, & \text{otherwise} \end{cases} \tag{6.2.26}$$

where $k = 1, \cdots, K$ represent different classes or categories of an attribute. In other situations, we may want to digitize a continuously varying parameter into pixels or clusters and use an indicator to label each group

$$I_k(\mathbf{x}) = \begin{cases} 1, & \text{if } \theta(\mathbf{x}) \in \Delta k \\ 0, & \text{otherwise} \end{cases} \tag{6.2.27}$$

where $\Delta k \ (k = 1, \cdots, K)$ represent nonoverlapping ranges. Equation (6.2.27) is particularly useful for geophysical and remote sensing applications, where the physical parameter under study often corresponds to a range of signals in the frequency domain. In either case, the following indicator semivariogram can be defined

$$\gamma_k(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2} Var[I_k(\mathbf{x}_i) - I_k(\mathbf{x}_j)], \quad k = 1, \cdots, K. \tag{6.2.28}$$

Analogous to variograms for continuous random variables, (6.2.28) measures continuity of an indicator variable. Thus, the greater the indicator variogram value, the less connected a class is in space. Similarly, the indicator cross-variogram can be defined between two classes as

$$\gamma_{kl}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2} E\left\{[I_k(\mathbf{x}_i) - I_k(\mathbf{x}_j)][I_l(\mathbf{x}_i) - I_l(\mathbf{x}_j)]\right\}.$$

We see that the indicator approach is nonparametric in the sense that it does not require a prior model, nor does it require estimation of the parameters of PDFs. Applying indicator statistics requires the ability to define K cutoff values in an unambiguous manner, which is not always straightforward. To facilitate the use of indicator statistics in parameter estimation or geostatistical simulation, the concept of transition probability is introduced which, similar to its counterpart in MCMC, defines the conditional probability that a class occurs at one (unsampled) location given a different class is observed at another location

$$P_{kl} = \Pr(I_l(\mathbf{x}_j) = 1 \big| I_k(\mathbf{x}_i) = 1).$$

In the context of geologic facies modeling, Carle and Fogg (1997) related transition probability to facies volumetric proportions, mean lengths, and juxtapositional tendencies (i.e., the tendency of facies occurring adjacent to each other), which essentially provide a multipoint statistical framework for simulating facies distributions. More details on multipoint statistics will be given shortly in Sect. 6.2.5.

Example 6.3 *Variogram of a Satellite Image*
Satellite imagery has become an indispensable resource for EWR applications. As satellites fly by physical objects, they register digital images at their own detection resolution. Each pixel of the image has a digital number (or pixel value) which may correspond to, for example, the elevation or reflectivity of the object as seen by the satellite. Analyses of the image patterns can help to characterize the roughness, regularity, symmetry, and uniformity of the image. There is a tremendous interest in relating satellite data to distributions of physical attributes either at or below the spatial resolution of a satellite.

Variograms can be used to parameterize the spatial distributions of digital numbers. Figure 6.11a (left) shows a scene acquired by the Landsat-7 satellite (http://landsat.usgs.gov), which remotely senses land coverage. The scene has a size of 400×640 pixels (size reduced for visualization) and corresponds to a mountainous region. We now perform variogram analysis on a part of the image (Fig. 6.11b, right), which has a size of 200×200 pixels and corresponds to a small foothill town. The resolution of each pixel is 15 m. The sample variogram of the digital numbers is constructed as

$$\hat{\gamma}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2} E\{[DN(\mathbf{x}_i) - DN(\mathbf{x}_j)]^2\}, \tag{6.2.29}$$

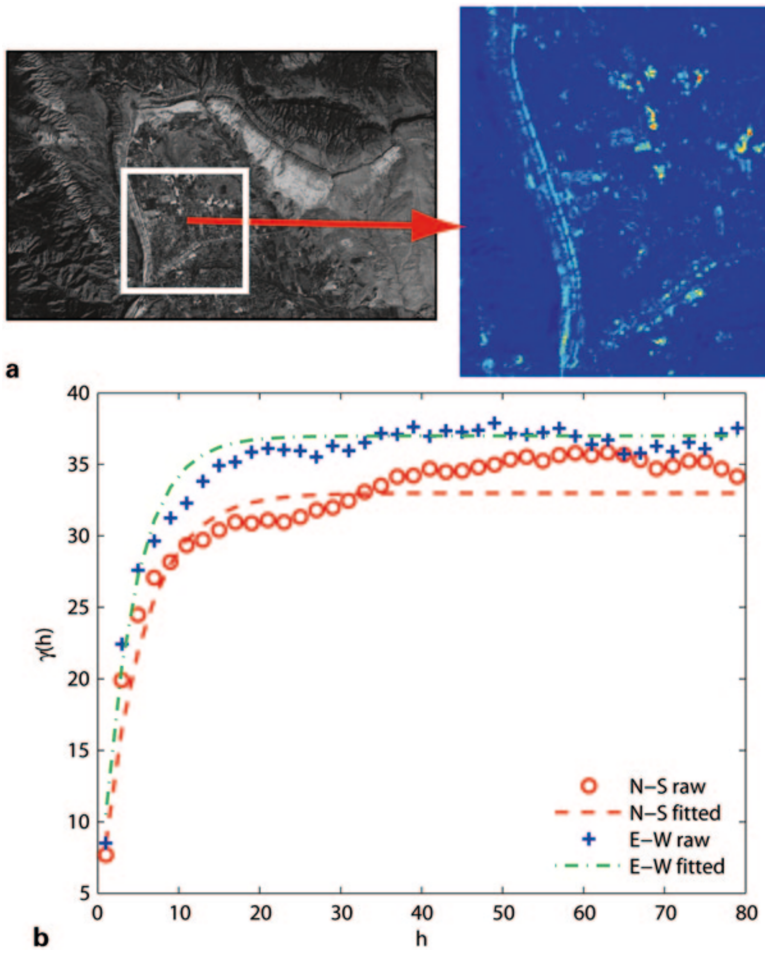


Fig. 6.11 **a** A scene from Landsat-7 (*left*). The extracted subarea (*right*) has a size of 200 × 200 pixels and is the subject of variogram analysis. **b** Variograms for N-S and E-W directions

where *DN* denotes the digital number. Figure 6.11b shows the resulting sample variogram for two orthogonal directions. Both variograms show a rather quick loss of correlation with separation distance, although the correlation in the N-S direction is slightly stronger than that in the E-W direction, reflecting the spatial orientation of the township. Both variograms were fit using an exponential model with the same nugget effect, but slightly different ranges.

6.2.3.5 Modeling Space-Time Processes

Many physical processes in EWR modeling exhibit both spatial and temporal variability. Examples include precipitation, vegetation, terrestrial water storage, and surface ozone levels. The geostatistical approach to modeling such physical quantity is to define a space-time stochastic process $\theta(\mathbf{x}, t)$. The notion of space-time process is necessary when time cannot be simply treated as another dimension or when observations from different times cannot be assumed as realizations from the same underlying spatial process. Similar to (6.2.15), a general variogram can be defined according to

$$\gamma_{st}(\mathbf{x}_i, t_1; \mathbf{x}_j, t_2) = \frac{1}{2} Var\left(\theta(\mathbf{x}_i, t_1) - \theta(\mathbf{x}_j, t_2)\right), \tag{6.2.30}$$

where the subscripts *s* and *t* denotes space and time, respectively. The space-time process is spatially stationary if γ_{st} is a function of separation vector $\mathbf{x}_i - \mathbf{x}_j$ only, and temporally stationary if γ_{st} only depends on time lag $t_1 - t_2$.

In the simplest case, the space-time process $\theta(\mathbf{x}, t)$ is assumed separable and can be modeled via either an additive or product model (assuming the mean is deterministic)

$$\theta(\mathbf{x}, t) = S(\mathbf{x}) + T(t) \text{ or } \theta(\mathbf{x}, t) = S(\mathbf{x})T(t), \tag{6.2.31}$$

in which *S* and *T* are uncorrelated stochastic process. For example, variations in groundwater well levels may be decomposed as intrawell variations superposed on top of a purely temporal regional variation. In the case of stationarity, possible separable covariance models are

Additive model:

$$C_{st}(\mathbf{r}, \tau) = C_s(\mathbf{r}) + C_t(\tau) \tag{6.2.32}$$

Product model:

$$C_{st}(\mathbf{r}, \tau) = C_s(\mathbf{r})C_t(\tau) \tag{6.2.33}$$

Product-sum model:

$$C_{st}(\mathbf{r}, \tau) = a_1 C_s(\mathbf{r})C_t(\tau) + a_2 C_s(\mathbf{r}) + a_3 C_t(\tau) \tag{6.2.34}$$

where $\mathbf{r}$ and τ are separation vector and temporal lag, respectively. The corresponding variograms can be derived easily.

A nonseparable stationary covariance model that is widely used today was proposed by Cressie and Huang (1999)

$$C(\mathbf{r}, u) = \sigma^2 \exp(-a^\delta |\tau| - c^2 \|\mathbf{r}\|^2 - \beta |\tau|^\delta \|\mathbf{r}\|^2),$$

where σ^2 is the variance of the space-time process, *a* and *c* are nonnegative scaling parameters, β is a space-time interaction parameter, and δ is either one or two. Clearly, when $\beta = 0$, the Cressie–Huang model degenerates into a separable model. Gneiting (2002) proposed another general nonseparable covariance model

$$C_{st}(\mathbf{r},\tau) = \frac{\sigma^2}{\left(a\left|\tau\right|^{2\alpha}+1\right)^{3\beta/2}} \exp\left(-\frac{c\left\|\mathbf{r}\right\|^{2\kappa}}{\left(a\left|\tau\right|^{2\alpha}+1\right)^{\beta\kappa}}\right),$$

in which σ^2 is the variance of the space-time process, α and κ are smoothness parameters in (0,1], β is a space-time interaction parameter in (0, 1], and a and c are nonnegative scaling parameters. In general, nonseparable covariance models are more flexible and realistic than separable covariance models, but require more consideration in model selection and more samples. Additional discussion on spatial and temporal statistics can be found in (Kyriakidis and Journel 1999; Gneiting et al. 2007; Cressie and Huang 1999; Stein 2005).

6.2.4 Geostatistical Interpolation

Geostatistical interpolation attempts to estimate the values of a random field at unobserved locations based on scattered point measurements. Various geostatistical interpolation algorithms, better known as kriging algorithms, have been developed for performing spatial interpolation on different types of random fields. The building block of kriging algorithms is variogram, which has been described in the previous section. In this section, we will first give a brief introduction to various kriging algorithms. Then we show how kriging can be used effectively as a parameterization method for inversion. For the history, theory, algorithms, and applications of geostatistics, readers may refer to a large number of related monographs and textbooks (e.g., Cressie 1993; Deutsch and Journel 1998; Goovaerts 1997; Isaaks and Srivastava 1989; Stein et al. 1999; Ripley 2004).

6.2.4.1 Simple Kriging

Simple kriging (SK) provides the simplest geostatistical interpolation algorithm. Let us assume that the underlying random field is second-order stationary with a known mean. Let $\{\mathbf{x}_i\}_{i=1}^m$ denote locations in a spatial domain $\mathbf{\Omega}$ where the random field $\theta(\mathbf{x})$ is observed. The SK estimator for $\theta(\mathbf{x})$ at an unobserved location $\mathbf{x}_0$ is a linear combination of observed values

$$\hat{\theta}(\mathbf{x}_0) = \mu + \sum_{i=1}^{m} \lambda_i^0[\theta(\mathbf{x}_i) - \mu], \quad \mu = E\left\{\theta(\mathbf{x})\right\} \tag{6.2.35}$$

where λ_i^0 are SK weights associated with $\mathbf{x}_0$. To estimate the unknown weights, the SK minimizes the mean square error

$$MSE = E\{(\theta - \hat{\theta})^2\}. \tag{6.2.36}$$

After substituting (6.2.35) into (6.2.36) and some algebraic manipulations, we obtain the following equation

$$MSE = \sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i^0 \lambda_j^0 C(\mathbf{x}_i, \mathbf{x}_j) + C(0) - 2\sum_{i=1}^{m} \lambda_i^0 C(\mathbf{x}_0, \mathbf{x}_i). \tag{6.2.37}$$

The SK weights can be solved by minimizing (6.2.37), which yields the so-called SK system of equations

$$\sum_{j=1}^{m} \lambda_j^0 C(\mathbf{x}_i, \mathbf{x}_j) = C(\mathbf{x}_i, \mathbf{x}_0), \tag{6.2.38}$$

where $C(\mathbf{x}_i, \mathbf{x}_j) = C(\mathbf{x}_i - \mathbf{x}_j)$ because of stationarity. From (6.2.38), we see that (i) the SK system of equations is only a function of measurement locations, but not the actual measurement values; and (ii) only the right-hand side of the equations needs to be evaluated for each new estimation location $\mathbf{x}_0$. After obtaining the SK weights, we can compute the SK variance as

$$Var(\hat{\theta}_0) = C(0) - \sum_{i=1}^{m} \lambda_i^0 C(\mathbf{x}_0, \mathbf{x}_i), \tag{6.2.39}$$

which quantifies the reduction in uncertainty due to measurements. Note that (i) the SK variance is not a measure of spatial variability and (ii) the assumption of stationarity is not required for SK, but is widely made in practice.

6.2.4.2 Ordinary Kriging

Unlike the SK, ordinary kriging (OK) does not require that the mean be known explicitly. If $\theta(\mathbf{x})$ can be modeled as an intrinsically stationary random field $\boldsymbol{\Omega}$ that satisfies (see also Sect. 6.2.3)

$$E[\theta(\mathbf{x} + \mathbf{r}) - \theta(\mathbf{x})] = 0, \tag{6.2.40}$$

$$Var[\theta(\mathbf{x} + \mathbf{r}) - \theta(\mathbf{x})] = 2\gamma(\mathbf{r}), \tag{6.2.41}$$

then we can apply OK. Again, let $\{\theta_i = \theta(\mathbf{x}_i) \mid i = 1, 2, \cdots, m\}$ be a set of measurements of $\theta(\mathbf{x})$ and we would like to estimate $\theta_0 = \theta(\mathbf{x}_0)$ for any unobserved site $\mathbf{x}_0$. As in the SK, we parameterize the unknown as a linear combination of measured values

$$\hat{\theta}_0 = \sum_{i=1}^{m} \lambda_i^0 \theta_i + \left(1 - \sum_{i=1}^{m} \lambda_i^0\right) \mu, \tag{6.2.42}$$

where $\lambda_i^0 = \lambda_i(\mathbf{x}_0),\ i = 1, 2, \cdots, m,$ is a set of OK weights to be determined. We also impose the following constraint on the OK weights so that the unknown mean can be removed

$$\sum_{i=1}^{m} \lambda_i^0 = 1. \tag{6.2.43}$$

To solve for λ_i^0, we again minimize the MSE

$$MSE = E\{(\theta_0 - \hat{\theta}_0)^2\}. \tag{6.2.44}$$

Similar to the SK, we can expand the MSE using (6.2.42)

$$E[(\hat{\theta}_0 - \theta_0)^2] = \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda_i^0 \lambda_j^0 E[(\theta_i - \theta_0)(\theta_j - \theta_0)]. \tag{6.2.45}$$

Because

$$\begin{aligned} E[(\theta_i - \theta_0)(\theta_j - \theta_0)] &= \frac{1}{2} E[(\theta_i - \theta_0)^2 + (\theta_j - \theta_0)^2 - (\theta_i - \theta_j)^2] \\ &= \gamma_{i0} + \gamma_{j0} - \gamma_{ij}, \end{aligned}$$

where $\gamma_{i0} = \gamma(\mathbf{x}_i - \mathbf{x}_0)$, $\gamma_{j0} = \gamma(\mathbf{x}_j - \mathbf{x}_0)$, and $\gamma_{ij} = \gamma(\mathbf{x}_i - \mathbf{x}_j)$, (6.2.45) can be rewritten as

$$E[(\hat{\theta}_0 - \theta_0)^2] = -\sum_{i=1}^{m} \sum_{j=1}^{m} \gamma_{ij} \lambda_i^0 \lambda_j^0 + 2 \sum_{i=1}^{m} \gamma_{i0} \lambda_i^0. \tag{6.2.46}$$

The advantage of (6.2.46) is that spatial data can be more readily fitted to a variogram model than a covariance model. Minimization of (6.2.46) with the linear constraint (6.2.43) is equivalent to minimizing the following Lagrange function

$$L = -\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \gamma_{ij} \lambda_i^0 \lambda_j^0 + \sum_{i=1}^{m} \gamma_{i0} \lambda_i^0 - \eta \left[\sum_{i=1}^{m} \lambda_i^0 - 1 \right], \tag{6.2.47}$$

where η is a Lagrange multiplier. Using the necessary condition of minimization $\partial L / \partial \lambda_i^0 = 0$ $(i = 1, 2, \cdots, m)$ and $\partial L / \partial \eta = 0$, we obtain the OK system of equations

$$\begin{aligned} &\sum_{j=1}^{m} \gamma_{ij} \lambda_j^0 + \eta = \gamma_{i0}, \ i = 1, 2 \cdots, m \\ &\sum_{i=1}^{m} \lambda_i^0 = 1, \end{aligned} \tag{6.2.48}$$

from which the $m+1$ unknowns can be determined. Substituting the solution λ_i^0 $(i = 1, 2, \cdots, m)$ into (6.2.42), we obtain the OK estimator for $\mathbf{x}_0$. OK is a best linear unbiased estimator (BLUE) and has the minimum variance of estimation given by

$$Var(\hat{\theta}_0 - \theta_0) = \sum_{i=1}^{m} \gamma_{i0} \lambda_i^0 + \eta. \tag{6.2.49}$$

Note that we use the same notations to denote the weights of SK and the weights of OK, but their values actually are different. OK follows naturally from variogram analysis and its weights always sum up to one, while the SK weights do not satisfy such constraint.

Kriging assigns larger weights to nearby measurements because variogram increases with the separation distance. Nevertheless, kriging is a global interpolation method, and all measurements are used in determining the interpolated value of a point. When $\theta(\mathbf{x})$ is not an intrinsically stationary random field in the region $\mathbf{\Omega}$, we cannot obtain the ordinary kriging estimates globally unless the region can be divided into several zones such that $\theta(\mathbf{x})$ satisfies the intrinsic stationarity condition in each zone. For example, the region $\mathbf{\Omega}$ may represent an aquifer with a layered structure and $\theta(\mathbf{x})$ is weakly stationary in each layer. In this case, we can build separate kriging estimators for different layers.

6.2.4.3 Regression Kriging

For a general nonstationary random field, we may use universal kriging (Cressie 1993; Marsily 1986) to filter out its spatially varying mean. But, regression kriging (RK) (Hengl et al. 2004; Odeh et al. 1995; Kitanidis 1997) is not only a more elegant method to deal with variable mean, but also an appropriate parameterization method for inverse solution.

A nonstationary random field, $\theta(\mathbf{x})$, can be decomposed into two parts, a deterministic trend (or drift) $\mu(\mathbf{x})$ describing the large-scale variability of the field, and a random fluctuation part $\varepsilon(\mathbf{x})$ describing the small-scale variability of the field, viz.

$$\theta(\mathbf{x}) = \mu(\mathbf{x}) + \varepsilon(\mathbf{x}). \tag{6.2.50}$$

The RK estimation of $\theta(\mathbf{x})$ is conducted in two stages. First, the trend is estimated by using a deterministic regression method. Second, the residuals are estimated by a statistical method. Assume that the trend is approximated by a parametric model $\hat{\mu}(\mathbf{x}, \boldsymbol{\beta})$, where $\boldsymbol{\beta}$ is a set of model parameter, and the fluctuation can be estimated by kriging $\hat{\varepsilon}(\mathbf{x}, \boldsymbol{\psi})$, where $\boldsymbol{\psi}$ is a set of hyperparameters appearing in the selected variogram model. We have the following algorithm for RK:

1. Use GLS to identify model parameters $\hat{\boldsymbol{\beta}}$ through fitting measurements $\{\theta_i = \theta(\mathbf{x}_i) \mid i = 1, 2, \cdots, m\}$ using a deterministic trend model $\hat{\mu}(\mathbf{x}, \boldsymbol{\beta})$.
2. Calculate the residuals by removing the trend, $\varepsilon_i = \theta_i - \hat{\mu}(\mathbf{x}_i, \hat{\boldsymbol{\beta}})$ for $i = 1, 2, \cdots, m$.
3. If the residual distribution is nearly normal with zero mean, calculate the sample variogram of the residuals, fit it to a selected variogram model, and identify the variogram parameters $\hat{\boldsymbol{\psi}}$.
4. Find SK or OK estimation of $\hat{\varepsilon}(\mathbf{x}, \hat{\boldsymbol{\psi}})$.
5. The estimated RF is given finally by $\hat{\theta}(\mathbf{x}) = \hat{\mu}(\mathbf{x}, \hat{\boldsymbol{\beta}}) + \hat{\varepsilon}(\mathbf{x}, \hat{\boldsymbol{\psi}})$.
6. The variance of estimation is given by summation.

$$Var(\hat{\theta}) = Var(\hat{\mu}) + Var(\hat{\varepsilon}). \tag{6.2.51}$$

For Step 1, either OLS or GLS may be used to perform linear regression. If the latter is used and the residuals are correlated, it may be necessary to perform Steps 1−4 in an iterative manner because estimation of $\hat{\beta}$ in Step 1 may require knowledge of the covariance of the residuals, which can change after Step 3. For more details and available software packages on RK, readers may refer to Hengl et al. (2004).

6.2.4.4 Indicator Kriging

Indicator kriging (IK) is used for interpolating categorical variables defined in Sect. 6.2.3. In particular, IK estimates the conditional probability that a category occurs at an observed location for the given information. Let $I_k(\mathbf{x})$ be an indicator random function of an attribute defined in (6.2.26) and its measured values at $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ be $\{I_{k,i} = I_k(\mathbf{x}_i), i = 1, 2, \cdots m\}$. Then at an unobserved site $\mathbf{x}_0$, $I_k(\mathbf{x}_0)$ can be estimated by OK

$$\hat{I}_k(\mathbf{x}_0) = \sum_{i=1}^{m} \lambda_i^0 I_{k,i}, \tag{6.2.52}$$

where OK weights $\{\lambda_i^0\}$ depend on the semivariogram of $I_k(\mathbf{x})$ defined in (6.2.28), which is identified by fitting measured values to a variogram model. The value of $\hat{I}_k(\mathbf{x}_0)$ is between 0 and 1, corresponding to the probability of the attribute at $\mathbf{x}_0$ falling into the category *k*.

Indicator kriging can also be used to estimate the probability that the value of a continuous distribution $\theta(\mathbf{x})$ (e.g., concentration of a contaminant) at a target point $\mathbf{x}_0$ exceed a threshold value θ_c (e.g., maximum contaminant level), given measured values at m sites $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ around $\mathbf{x}_0$. In this case, we define an indicator random function $I_1(\mathbf{x})$ for $\theta(\mathbf{x})$ such that

$$I_1(x) = \begin{cases} 1, & \theta(\mathbf{x}) > \theta_c \\ 0, & \text{otherwise} \end{cases}$$

Measured values $\{\theta(\mathbf{x}_1), \theta(\mathbf{x}_2), \cdots, \theta(\mathbf{x}_m)\}$ of $\theta(\mathbf{x})$ are then transferred into "measured values" $\{I_1(\mathbf{x}_1), I_1(\mathbf{x}_2), \cdots, I_1(\mathbf{x}_m)\}$ of the indicator random function $I_1(\mathbf{x})$, and the value at a target point $\mathbf{x}_0$, $\hat{I}_1(\mathbf{x}_0) = p[\theta(\mathbf{x}_0) > \theta_0]$, can be estimated by the indicator kriging (6.2.52) for $k = 1$.

Besides indicator kriging, there are other variants of kriging, such as disjunctive kriging, multi-Gaussian kriging, lognormal kriging, and so forth. These methods first transfer the measured data of a non-Gaussian distribution into Gaussian distribution and then use the OK to find the interpolation results. Note that in our discussion so far we have assumed that all measurements are used. To make the calculations faster, practical algorithms often only consider observations located in a local neighborhood of each estimation site such that the size of kriging system of equations is significantly reduced.

Detailed discussion of the theory and available software packages on various kriging algorithms can be found in Goovaerts (1997), Deutsch and Journel (1998) (GSLIB), Webster and Oliver (2001), Bivand et al. (2008) (GSTAT in R), and Remy et al. (2009) (Stanford Geostatistical Modeling Software or SGeMS). As an interpolation method, kriging has found interesting applications in many fields, including EWR (Knotters et al. 2010).

6.2.4.5 Kriging for Parameterization

The main purpose of introducing kriging in this chapter is to use it as a technique to parameterize a random field for inversion. In its general form, kriging parameterizes a random function as a linear combination of its measured values

$$\hat{\theta}(\mathbf{x}) = \sum_{i=1}^{m} \theta(\mathbf{x}_i)\lambda_i(\mathbf{x}, \mathbf{v}, \boldsymbol{\psi}), \quad \mathbf{x} \in \Omega, \tag{6.2.53}$$

where the kriging weights $\{\lambda_i(\mathbf{x}, \mathbf{v}, \boldsymbol{\psi})\}$ are obtained by solving one of the kriging systems introduced before, such as the OK in (6.2.48). Equation (6.2.53) suggests that the kriging coefficients play the role of basis functions, in which the shape parameters $\mathbf{v} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ are the basis points of interpolation, and $\boldsymbol{\psi}$ is a set of statistical parameters defining the random field, such as parameters defining a variogram model. To obtain an estimate of $\hat{\theta}(\mathbf{x})$, we need not only measurements $\{\theta(\mathbf{x}_1), \theta(\mathbf{x}_2), \cdots, \theta(\mathbf{x}_m)\}$, but also the statistical parameters $\boldsymbol{\psi}$.

In Sect. 6.2.2, we presented three cases of using interpolation methods for parameter estimation. Here, we describe how kriging fits into three cases. For Case 1, the unknown parameter is estimated with its real measurements $\{\theta(\mathbf{x}_1), \theta(\mathbf{x}_2), \cdots, \theta(\mathbf{x}_m)\}$. As an interpolation method, (6.2.53) can be used directly for this case, where $\boldsymbol{\psi}$ is estimated from the same set of parameter measurements. In fact, we have learned in the last section about fitting a variogram model using measurements. Because kriging satisfies the requirements presented in Sect. 6.2.1, it is qualified as a parameterization method for inversion. For Case 2 of Sect. 6.2.2, $\{\theta(\mathbf{x}_1), \theta(\mathbf{x}_2), \cdots, \theta(\mathbf{x}_m)\}$ in (6.2.53) are considered as hypothetical measurements which, together with statistical parameters, are estimated indirectly from the measurements of state variables by inverse solution. While for Case 3 of Sect. 6.2.2, hypothetical measurements and statistical parameters are estimated when both parameter measurements and state observations are available. We will return to these topics in the next chapter when introducing the cokriging and pilot point methods.

6.2.5 Multipoint Statistics

Parameterization of spatial patterns is often needed in many EWR applications. A primary reason is to better characterize the spatial distribution of attributes (e.g., rock facies and soil types) which, in turn, can help improve the accuracy of a distributed model. In geoscience, the observations that are available for parameterization and simulation are often classified as either "hard data" or "soft data." The former refer to direct observations such as borehole logs and outcrop analyses, whereas the latter refer to indirect observations obtained via noninvasive technologies such as geophysical surveys or x-ray computed tomography. All this prior information can be used to develop a conceptual model of material pattern distributions, as well as to derive statistical controls, which are statistics of geometric measures such as volumetric proportions, shape lengths, and juxtapositional relationships of different material types (see also our discussion related to transition probability in Sect. 6.2.3). Now the question is how to factor the derived statistical controls and conceptualization of pattern distributions into parameterization.

Many of the traditional geostatistical descriptors introduced in previous sections are insufficient for creating realistic spatial patterns. Variograms explore two-point statistics of variates and covariates (e.g., geophysical data) via covariance or cross-covariance. Therefore, by way of construction, variograms do not contain much geometrical information and will have difficulty in simulating complex topological patterns (i.e., curvilinear fluvial channels or micropore structures of materials). Although the indicator variograms make an extra effort toward this direction by providing two-point statistics of categorical variables, they cannot easily fuse other types of statistic controls and, thus, tend to yield unrealistic or overly smooth spatial structures. It is also known that two structures having distinctively different geometries may have similar variograms. A prerequisite for structure imitation is to honor sample statistics of geometric measures, which can serve as both guidance and constraints during simulations. For example, the correct modeling of juxtapositional relationship is crucial for honoring the long-range connectivity of spatial structures. In the geostatistics literature, object-based algorithms have long been used for structure imitation (Koltermann and Gorelick 1996); however, object-based models suffer from not being able to condition to local data of different volumes. Two statistical methodologies have been independently developed for circumventing these problems by combining the strengths of variogram- and object-based techniques: the Markov chain transition probability approach (Carle and Fogg 1996, 1997) and the pixel-based, multipoint statistical approach (Journel 1992; Strebelle 2000), both are rooted in MCMC. The former is parametric, whereas the latter is nonparametric. We will describe the pixel-based multipoint statistical approach below.

As its name suggests, the multipoint statistical approach parameterizes joint correlations among multiple points or patterns. The application of multipoint statistical approach follows three main steps. The first step is development of training image(s), which can be in the form of either continuous or categorical variables. *Training images* are conceptual assemblages of patterns that reflect the general aspects of spatial structures or textures expected to be present in actual objects and need not have any local accuracy (Caers and Zhang 2004). For example, a training image can be drawn by a geologist based on his/her interpretation of field observations or generated using an object-based modeling technique. In the Bayesian sense, the role of training images is equivalent to prior PDFs for generating random samples which are, in the current context, the local patterns. The training image concept implicitly assumes stationarity and ergodicity, meaning different training images need to be developed for different depositional environments. An inherent difficulty of using multipoint statistics is related to the parameterization of the training image. Mariethoz and Kelly (2011) presented a parameterization method based on random transformation. It enables the generation of complex geological images whose spatial structure can be parameterized by adjusting the statistics of the random transformations. Michael et al. (2010) combined multipoint statistics with geologic-processing models to construct realistic 3-D realizations conditioned to data. Note that the local accuracy is enforced during the simulation stage through conditioning patterns to data, as we will see later,

In the second step, the local patterns manifested by a training image are extracted using a template consisting of $n+1$ pixels, where the location of the center of the template is denoted by $\mathbf{x}_0$ and the locations of all other n offset pixels (or voxels if 3D) are denoted by $\mathbf{x}_j$ ($j = 1,\cdots,n$). A *template* can be defined as

$$\mathcal{T} = \{\mathbf{x}_j : \mathbf{x}_0 + \mathbf{r}_j \big| \mathbf{x}_0, j = 1,\cdots,n\}, \tag{6.2.54}$$

where $\mathbf{r}_j$ is the separation vector. Figure 6.12 illustrates a 2D binary-phase training image (left) and a local pattern captured by a 7×7 square template (right). Templates of different shapes and sizes can be used, depending on the problem at hand. Because a template includes the relative positions of all neighbors of $\mathbf{x}_0$ and their values, it is very rich in terms of information content. Scanning of the training image using a template $\mathcal{T}$ results in a pattern set (or database) $\Omega_{\mathcal{T}}$

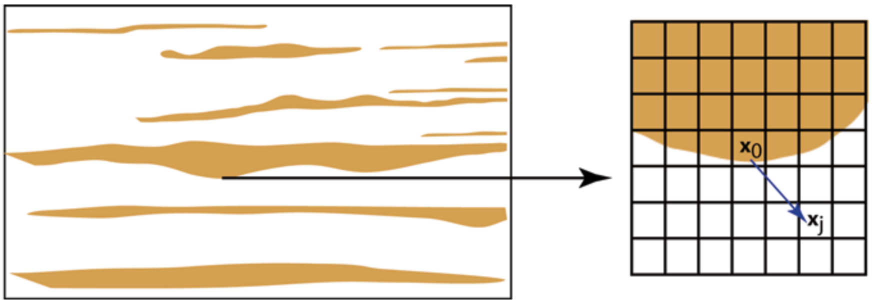


Fig. 6.12 Illustration of **a** a binary training image and **b** a 7 × 7 template used to scan it

$$\Omega_{\mathcal{T}} = \{pat(\mathbf{x}_j), j = 1, \cdots, N\}, \tag{6.2.55}$$

where N is the number of central locations in the training image. Given $\Omega_{\mathcal{T}}$, the probability of occurrence of a particular local pattern, as seen by the template $\mathcal{T}$, can be determined by

$$P(pat_n) = c(pat_n) / N, \tag{6.2.56}$$

where $c(pat_n)$ is the number of replicates of a local pattern, pat_n, in $\Omega_{\mathcal{T}}$. It is important that $\Omega_{\mathcal{T}}$ consists of enough replicates of different local patterns so that the calculated probabilities are not biased. Here replicate patterns refer to all patterns having the same geometry and values. Equation (6.2.56) can be extended to the general case of K different categories, where $c_k(pat_n)$ $(k = 1, \cdots, K)$ can be defined to count the number of replicates of pat_n whose central pixel is category k. To be more precise, the training image is first converted to K binary maps, each original pattern is represented by K sets of binary patterns, and from which the probability in (6.2.56) can be calculated. This completes the second step of the multipoint statistical approach and, also, the parameterization stage.

The third step of multipoint statistical approach is sequential simulation, where at each pixel a pattern is looked up and extracted from $\Omega_{\mathcal{T}}$ to match the existing pattern in a neighborhood, which is determined by previously simulated pixels and/or conditioning data.

So far we have talked about classifying and storing distinct local patterns in a pattern database without specifying how. Although a pixel-wise comparison can be used to determine the similarity between two patterns, it is not very efficient when the number of patterns is large. Alternatively, a scoring system may be used

$$S = \sum_{j=1}^{n} w(\mathbf{r}_j) T(\mathbf{x}_0 + \mathbf{r}_j \,|\, \mathbf{x}_0), \tag{6.2.57}$$

where $w(\mathbf{r}_j)$ are weighting coefficients and $T(\mathbf{x}_0 + \mathbf{r}_j \,|\, \mathbf{x}_0)$ are values of the pixels. Equation (6.2.57) can be regarded as an image filter and the role of $w(\mathbf{r}_j)$ in

(6.2.57) is similar to that of the weighting matrix in MRF. A single score may not be sufficient to identify all geometrical characteristics of a pattern. Multipoint statistical algorithms typically employ more than one filter. This is a rather unique aspect of the multipoint statistical approach because the filters act as dimension reduction techniques: an *n*-dimensional pattern is translated to only a small number of filter scores. Zhang (2006) defines three directional filters:

- The average filter

$$f_1 = 1 - \frac{|i|}{m}, \quad i = -m, \ldots + m, \tag{6.2.58}$$

where m is the half-width of the template (number of pixels) and f_1 ranges from 0 to 1.

- The gradient filter,

$$f_2 = i / m, \quad i = -m, \ldots, +m, \tag{6.2.59}$$

which ranges from −1 to 1.

- The curvature filter,

$$\frac{2|i|}{m} - 1, \tag{6.2.60}$$

which ranges from −1 to 1.

For a 2D template with two principal directions (e.g., N-S and E-W), (6.2.58)–(6.2.60) define six filters; hence, the dimension of a template is reduced from n to 6. Data clustering algorithms (e.g., k-means) can then be used to group similar patterns into classes and store them in a pattern database for use during simulation. This filter-based algorithm has been implemented in the multipoint statistical simulator, FILTERSIM, which is included in the open-source geostatistical simulation package, Stanford Geostatistical Modeling Software (SGeMS) (Zhang 2006; Wu et al. 2008; Remy et al. 2009). Complications arise when correlation exists at multiple scales. If a single template is used and the size of the template size is small, the long-range continuity is not reproduced well; on the other hand, if the template size is too large, the number of "orphans" in Ω_T (i.e., local patterns without replicates) increases dramatically. A practical trick is to apply multiple nested templates to capture pattern structures at different scales (Zhang 2006).

Scanning and storing patterns can be computationally demanding when the size of a training image is large. Mariethoz et al. (2010) proposed to directly sample the training image (in a local window) for a given pattern, instead of storing all training pattern probability values in a database a priori. The "on-demand" direct sampling method is more efficient than the database approach and can be used to simulate both categorical and continuous random variables.

The multipoint statistical methods have been used in a large number of EWR applications, including subsurface flow and transport (Sun et al. 2008; Weissmann et al. 2002; Doughty and Pruess 2004; Caers 2003; Feyen and Caers 2006; Sun et al. 2009), groundwater and surface water interaction (Zhou et al. 2013; Fleckenstein et al. 2006), and remote sensing (Cao et al. 2011; Jha et al. 2013).

The multipoint statistical approach represents a good example of the increasing use of dimension-reduction techniques in parameterization of distributed variables. Unlike scalar statistical parameters in the variogram case, local patterns are vector data and the classification of these patterns becomes a key part of the parameterization technique. Like the variograms, the multipoint statistical parameterization does not provide a differentiable relationship between the input (i.e., training image) and the output (i.e., realizations of the random field). Such relationship, if exists, can improve the efficiency of gradient-based parameter estimation algorithms. We will revisit this issue in Sect. 6.4. In the next section, we formally introduce dimension reduction techniques.

6.3 Linear Dimension Reduction

The process of dimension reduction is closely related to finding the intrinsic dimension of a high-dimensional dataset, such as the pattern database in the previous section. Loosely speaking, the *intrinsic dimension* of a dataset corresponds to the minimum number of features (variables) that are necessary to explain the observed high-dimensional data or to represent a high-dimensional variable. A key performance criterion of dimension reduction techniques is how well the information content and/or geometry of the original dataset can be retained or how to identify the intrinsic dimension. In this section, we introduce several commonly used linear dimension reduction techniques. For detailed discussions on these techniques, readers may refer to Jolliffe (2002) and Härdle and Simar (2012).

6.3.1 Principal Component Analysis

Principal component analysis (PCA) is one of the oldest and, yet, still the most widely used linear dimension reduction techniques (Hotelling 1933). Historically it has been reinvented a number of times and is also known under names such as empirical orthogonal function (EOF), the discrete Karhunen–Loève (KL) transform, and proper orthogonal decomposition (POD) in different application fields.

Let us consider an m-dimensional random variable $\mathbf{X}$. In a model reduction scenario, $\mathbf{X}$ may consist of m state variables (e.g., streamflow from the outlets of m subbasins or simulated groundwater water levels at m locations). In a data reduction scenario, $\mathbf{X}$ may be a large sample dataset of multiple variables taken at different measurement times. Because of correlation among variables, the information in $\mathbf{X}$ may be redundant. The purpose of PCA is to find a small number of *principal com-*

ponents (PCs) (see text below for definition) to account for most of the total variance in the original $\mathbf{X}$. Mathematically, the process of finding PCs is equivalent to transforming $\mathbf{X}$ into another random variable $\mathbf{Z}$ such that (i) the dimension of $\mathbf{Z}$ is lower than the dimension of $\mathbf{X}$, (ii) the components of $\mathbf{Z}$ are independent of each other (i.e., the covariance matrix of $\mathbf{Z}$ is diagonal), and (iii) the components of $\mathbf{Z}$ are ordered according to their variances from the largest to smallest.

6.3.1.1 Finding the Principal Components

Let us assume we have *n* samples of an *m*-dimensional random variable $\mathbf{X}$ and store them as an $m \times n$ sample matrix $\mathbf{X}_s$. Without loss of generality, we assume that the sampling mean of each component has already been removed from the data. Thus, the $m \times m$ sample covariance matrix is given by

$$\mathbf{C}_\mathbf{x} = \frac{1}{n-1}\mathbf{X}_s\mathbf{X}_s^T. \tag{6.3.1}$$

Our purpose is to find an orthogonal transformation matrix **P** such that the sample covariance $\mathbf{C}_\mathbf{z}$ for dataset $\mathbf{Z}_s = \mathbf{P}\mathbf{X}_s$ will become a diagonal matrix

$$\mathbf{C}_\mathbf{z} = \frac{1}{n-1}\mathbf{Z}_s\mathbf{Z}_s^T = \frac{1}{n-1}(\mathbf{P}\mathbf{X}_s)(\mathbf{P}\mathbf{X}_s)^T = \frac{1}{n-1}\mathbf{P}(\mathbf{X}_s\mathbf{X}_s^T)\mathbf{P}^T. \tag{6.3.2}$$

Because $\mathbf{X}_s\mathbf{X}_s^T$ is a symmetric square matrix, its *eigenvalue decomposition* (see Appendix C) is

$$\mathbf{X}_s\mathbf{X}_s^T = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T, \tag{6.3.3}$$

where $\mathbf{Q}$ is an orthogonal matrix with the eigenvectors of $\mathbf{X}_s\mathbf{X}_s^T$ as its columns, and $\boldsymbol{\Lambda}$ is a diagonal matrix with the eigenvalues ($\lambda_1, \lambda_2, \cdots, \lambda_m$) of $\mathbf{X}_s\mathbf{X}_s^T$ as its entries. The eigenvalue decomposition problem can be performed by using any linear algebra package, such as ARPACK (Lehoucq et al. 1998). The Matlab's `eigs` function provides an implementation of the ARPACK, and R's `eigen` function offers similar capability. By default, the eigenvalues are ordered from the largest to smallest.

Setting the transformation matrix $\mathbf{P} = \mathbf{Q}^T$ and substituting $\mathbf{X}_s\mathbf{X}_s^T = \mathbf{P}^T\boldsymbol{\Lambda}\mathbf{P}$ into (6.3.2), we obtain a diagonal matrix

$$\mathbf{C}_\mathbf{z} = \frac{1}{n-1}\mathbf{P}(\mathbf{P}^T\boldsymbol{\Lambda}\mathbf{P})\mathbf{P}^T = \frac{1}{n-1}(\mathbf{P}\mathbf{P}^{-1})\boldsymbol{\Lambda}(\mathbf{P}\mathbf{P}^{-1}) = \frac{1}{n-1}\boldsymbol{\Lambda}. \tag{6.3.4}$$

In the above equation, $\mathbf{P}^T = \mathbf{P}^{-1}$ is used because $\mathbf{P}$ is an orthogonal matrix (see Appendix C). The eigenvectors of $\mathbf{X}_s\mathbf{X}_s^T$ are called PCs of $\mathbf{X}_s$. The above derivation shows that when transformation matrix $\mathbf{P}$ is formed with the PCs as its rows, the covariance matrix of dataset $\mathbf{Z}_s = \mathbf{P}\mathbf{X}_s$ becomes a diagonal matrix, and the elements on the diagonal are the variances of $\mathbf{X}_s$ associated with the PCs ordered from the largest to the smallest.

PCA provides a representation of correlated data using uncorrelated orthogonal PCs. If we consider PCs as orthogonal basis functions, then PCA provides a linear parameterization method in the form of (6.1.3).

6.3.1.2 Connection with SVD

SVD was introduced in Chap. 2 in the context of regularization of a linear system. There is a close relationship between PCA and SVD. Given a sample matrix $\mathbf{X}_s$, let us define a new $n \times m$ matrix $\mathbf{Y}$ such that

$$\mathbf{Y} = \frac{1}{\sqrt{n-1}}\mathbf{X}_s^T. \tag{6.3.5}$$

It is easy to show that $\mathbf{Y}^T\mathbf{Y} = \frac{1}{n-1}\mathbf{X}_s\mathbf{X}_s^T = \mathbf{C}_\mathbf{x}$. Let the SVD of $\mathbf{Y}$ be $\mathbf{U\Sigma V}^T$, then $\mathbf{Y}^T\mathbf{Y} = \mathbf{V\Sigma}^2\mathbf{V}^T$ or $\mathbf{X}_s\mathbf{X}_s^T = (n-1)\mathbf{V\Sigma}^2\mathbf{V}^T$. Therefore, $\mathbf{V}$ is equal to $\mathbf{Q}$ in (6.3.3), and the columns of $\mathbf{V}$ are the PCs of $\mathbf{X}_s$. Because SVD is more general than eigenvalue decomposition (e.g., SVD exists for any matrix, not just square matrices; singular vectors are always orthogonal) and is easier to calculate, SVD is usually used behind the PCA routines.

6.3.1.3 Dimension Reduction

As mentioned before, the main use of PCA is for dimension reduction. After performing PCA on a sample dataset of $\mathbf{X}$, we may find that a large number of eigenvalues have close-to-zero values, indicating information redundancy. Thus, PCs corresponding to small eigenvalues can be removed, which means that the dimension of $\mathbf{X}$ can be reduced from m to an intrinsic dimension k without losing major features of the dataset. During this reduction process, all components that are non-independent or caused by noises are eliminated. Recall that the same concept is behind TSVD introduced in Chap. 2, where the purpose there was to improve stability of inverse solutions by truncating small singular values.

One method for determining the intrinsic dimension is to use scree plot, which is simply a plot of all eigenvalues in a decreasing order. If the scree plot reveals an L-shaped curve, the eigenvalue corresponding to the turning point of the curve can be chosen for the intrinsic dimension, such that only PCs corresponding to eigenvalues greater or equal to the turning point are kept. Another estimation method is to choose k PCs such that

$$\sum_{i=1}^{k} \lambda_i \Big/ \sum_{i=1}^{m} \lambda_i \geq 100\alpha\%, \tag{6.3.6}$$

where α is a threshold specified by the user and λ_i are eigenvalues. There are other more involved methods for estimating the intrinsic dimension of a high dimensional dataset, including the geometric methods (Kégl 2003) and the probability-based methods (Levina and Bickel 2004) that usually require a large number of data.

In summary, PCA is a nonparametric linear dimension reduction technique whose effectiveness is tied to the validity of its main assumptions: (i) linearity, only linear transformation are involved; (ii) second-order statistics, the sample covariance sufficiently captures the probability distribution of $\mathbf{X}$; and (iii) the major features of the physical problem can be captured by a small number of PCs corresponding to large variances (eigenvalues). PCA cannot incorporate prior structure information. PCA is available in many high-level linear algebra or statistical packages such as Matlab (`pca`) and R (`princomp`).

6.3.2 Factor Analysis

Many environmental datasets, such as surface water monitoring sets, often include samples of multivariates from multiple sites. The goal of factor analysis (FA) is to find a small set of latent variables to explain patterns of a large set of measured variables without losing important information. In statistics, the term *latent variable* is used to refer to an inferred rather than a directly measured variable.

FA was originally introduced in psychology in connection with human intelligence assessment (Spearman 1904). Like PCA, FA is a dimension-reduction technique based on second-order statistics. Unlike PCA, which finds PCs that are not necessarily interpretable, FA aims to find a small number of factors that are physically interpretable and meaningful. FA focuses on discovering the underlying structure of a dataset and on revealing any latent variables that cause the observed variables to co-vary, rather than simply maximizing the variance. Therefore, FA can be especially useful in helping design new experiments or conduct sensitivity analysis (see also Chap. 10).

FA assumes that a linear model exists between an *m*-dimensional random variable $\mathbf{X}$ and a *k*-dimensional factor vector **f**,

$$\mathbf{X} - \boldsymbol{\mu} = \mathbf{L}\mathbf{f} + \boldsymbol{\varepsilon}, \tag{6.3.7}$$

where $\boldsymbol{\mu}$ is the mean of $\mathbf{X}$, the $m \times k$ matrix $\mathbf{L}$ is called the loadings matrix. In the literature, $\mathbf{f}$ is often referred to as *common factors* and $\boldsymbol{\varepsilon}$ the *specific factors*. FA assumes the following for $\mathbf{f}$ and $\boldsymbol{\varepsilon}$:

- $\boldsymbol{\varepsilon}$ has zero-mean and its diagonal covariance matrix is denoted by $\boldsymbol{\Psi}$;
- $\mathbf{f}$ has zero-mean and its covariance matrix is the identity matrix $\mathbf{I}$; and
- $\mathbf{f}$ and $\boldsymbol{\varepsilon}$ are independent.

The first two bullets in the above basically require the components of $\mathbf{f}$ and $\boldsymbol{\varepsilon}$ be uncorrelated. Using (6.3.7) and these assumptions, we can derive the covariance matrix

$$\mathbf{C}_{\mathbf{X}} = E[(\mathbf{X}-\boldsymbol{\mu})(\mathbf{X}-\boldsymbol{\mu})^T] = \mathbf{L}E(\mathbf{f}\mathbf{f}^T)\mathbf{L}^T + E(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T) = \mathbf{L}\mathbf{L}^T + \boldsymbol{\Psi}, \tag{6.3.8}$$

where $\mathbf{L}\mathbf{L}^T$ is called *common variance*, which represents the part of data variance that is accounted for by the common factors $\mathbf{f}$; $\boldsymbol{\Psi}$ is called *specific variance* and it represents the variance that is not shared by other variables. The greater the specific variance of a variable, the less is its relevance in the factor model. In particular, the diagonal elements in the above equation give variances

$$\sigma_{ii}^2 = \sum_{j=1}^{k} l_{ij}^2 + \psi_{ii},\ i = 1, 2, \cdots, m, \tag{6.3.9}$$

where σ_{ij} and l_{ij} are elements of $\mathbf{C}_{\mathbf{X}}$ and $\mathbf{L}$, respectively; ψ_{ii} is the variances of $\boldsymbol{\varepsilon}$. The summation term on the right-hand side of the above equation is called the *communality* and is denoted by h_i^2. Communalities measure the variances due to all common factors. The remaining problem is how to determine the loadings matrix $\mathbf{L}$ and estimate the specific variance $\boldsymbol{\Psi}$ through sample data.

Let us denote the sample matrix by $\mathbf{X}_s$. We can use PCA to find the decomposition $\mathbf{X}_s\mathbf{X}_s^T = \mathbf{P}^T\boldsymbol{\Lambda}\mathbf{P}$, where the rows of $\mathbf{P}$ consists of PCs. By reducing the number of PCs to *k*, we obtain an approximation

$$\mathbf{X}_s\mathbf{X}_s^T \approx \mathbf{P}_k^T\boldsymbol{\Lambda}_k\mathbf{P}_k = \mathbf{P}_k^T\boldsymbol{\Lambda}_k^{1/2}\boldsymbol{\Lambda}_k^{1/2}\mathbf{P}_k = (\mathbf{P}_k^T\boldsymbol{\Lambda}_k^{1/2})(\mathbf{P}_k^T\boldsymbol{\Lambda}_k^{1/2})^T, \tag{6.3.10}$$

where $\mathbf{P}_k$ is a $k \times m$ matrix with the first *k* PCs of $\mathbf{X}_s$, $\{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_k\}$, in its rows; and $\boldsymbol{\Lambda}_k$ is a $k \times k$ diagonal matrix with the first largest eigenvalues, $\{\lambda_1, \lambda_2, \cdots, \lambda_k\}$, as its entries. Now the FA equation (6.3.7) can be completely determined. Let the loadings matrix be $\mathbf{L} \triangleq \mathbf{P}_k^T\boldsymbol{\Lambda}_k^{1/2}$, we then use (6.3.9) to estimate the variance of the specified factors

$$\psi_{ii} = \hat{\sigma}_{ii} - \hat{h}_i^2,\ i = 1, 2, \cdots, m \tag{6.3.11}$$

where $\hat{\sigma}_{ii}$ and $\hat{h}_i^2 = \sum_{j=1}^{k} \lambda_j e_{ji}^2$ are the sampling variance and the estimated communality of the ith component, respectively.

From the above discussion we can see the differences between PCA and FA. In PCA, the number *k* of PCs is determined objectively by analyzing the sample covariance, whereas in FA the number *k* of factors is determined somewhat by prior knowledge. Due to the existence of specific factors, (6.3.11) shows that FA does not discriminate the variance caused by common and specific factors. Therefore, the most important factor in PCA may not coincide with the most important factor in FA. Moreover, FA can be used to uncover the underlying structure of a model (Exploratory FA) or to confirm a proposed structure of a model (Confirmatory FA) (Raykov and Marcoulides 2008).

Another difference between PCA and FA is that PCs are uniquely determined, while factor loadings are not unique. In fact, for any orthogonal matrix **R**, (6.3.7) can be replaced by

$$\mathbf{X} - \boldsymbol{\mu} = (\mathbf{LR})(\mathbf{R}^T \mathbf{f}) + \boldsymbol{\varepsilon}. \tag{6.3.12}$$

Nonuniqueness is both a disadvantage and an advantage of FA. For example, we can use a rotation transform **R** to simplify the structure in the loadings matrix, such that each factor has a small number of large loadings and a large number of small (or zero) loadings (Raykov and Marcoulides 2008). *Varimax*, developed by Kaiser (1958), is a commonly used criterion for orthogonal rotation. It maximizes the variance of the squared loadings of a factor on all the variables in a factor matrix. After a varimax rotation, each component of $\mathbf{X}$ is associated with only one or few factors, and each factor represents only one or few components. Note that orthogonal rotations require factors to be uncorrelated. If the factors are correlated, using orthogonal rotation may result in a loss of valuable information. In this case, oblique rotations can provide a more accurate solution (Costello and Osborne 2005).

FA is available in most statistical analysis packages, such as Matlab (`factoran`) and R (`factanal`).

Example 6.4 *Surface Water Quality Data Analysis Using FA*
Boyacioglu et al. (2005) presented surface water quality data collected from 17 monitoring stations located in the Buyuk Menderes River Basin in western Turkey. The river basin had been subject to increased pollution due to various anthropogenic activities. The Buyuk Menderes River Basin dataset consists of measurements of a total of nine variables collected during the high- and low-flow periods of the Buyuk Menderes River, namely, electrical conductivity, total organic content, biological oxygen demand, chemical oxygen demand, sodium, total coliform, total dissolved solids, sulfate, and total Kjeldahl nitrogen. We now use the low-flow dataset to demonstrate how FA can be used to reduce the dimension.

The FA proceeded in three steps:

1. First, the raw data were converted to standard normal random variates and then a sample covariance (or correlation) matrix of all variables was calculated. The raw data should be relatively normally distributed, and the correlation matrix should be full rank and positive definite. This requires elimination of dependent variables before conducting the FA. If all nine variables were used, the determinant of the correlation matrix would be negative, suggesting linear dependency between some variables. While there are several options, we dropped sulfate from the analysis. The resulting correlation matrix is tabulated in Table 6.1.
2. In the second step, factors from the correlation matrix were extracted. To determine the number of necessary factors, the eigenvalues of the correlation matrix was calculated (see Fig. 6.13a) and the number of latent factors was set to the number of eigenvalues that are greater than one. This is the so-called Kaiser criterion in the literature (Simeonov et al. 2003; Raykov and Marcoulides 2008). In the current case, the first two eigenvalues account for about 85 % of the total

variance. The number of factors is set to two. Note that instead of the Kaiser criterion, it is always a good idea to use scree plot to assess the number of factors to retain (Costello and Osborne 2005).

3. Finally, the factors were rotated using the varimax method to enhance interpretability.

Steps 2 and 3 above were carried out using the Matlab function `factoran`.

Figure 6.13b plots the two factors for all water quality parameters. Liu et al. (2003) classified the significant loadings as (i) strong (>0.75), (ii) moderate (0.50 to 0.75), and (iii) weak (0.30 to 0.50) based on absolute loading values. Costello and Osborne (2005) suggested that a factor with fewer than three items is generally weak and unstable; five or more strongly loading items (0.50 or better) are desirable and indicate a solid factor. Using 0.5 as a threshold, we see that the first loading factor is significant for electric conductivity, sodium, total dissolved solids, and total Kjeldahl nitrogen, whereas the second factor is significant for the rest of the parameters. This result has good physical meanings. The first group of parameters

Table 6.1 Correlation matrix for water quality data, adapted from Table 3 of Boyacioglu et al. (2005)

Parameter	EC	TOC	BOD	COD	Sodium	T. Coli	TDS	Total *N*
EC[a]	1.000	0.422	0.540	0.277	0.864	0.536	0.989	0.798
TOC[b]	0.422	1.000	0.697	0.350	0.584	0.428	0.430	0.586
BOD[c]	0.540	0.697	1.000	0.799	0.583	0.779	0.523	0.696
COD[d]	0.277	0.350	0.799	1.000	0.233	0.786	0.277	0.554
Sodium	0.864	0.584	0.583	0.233	1.000	0.376	0.890	0.858
T. Coli.[e]	0.536	0.428	0.779	0.786	0.376	1.000	0.505	0.585
TDS[f]	0.989	0.430	0.523	0.277	0.890	0.505	1.000	0.845
Total N	0.798	0.586	0.696	0.554	0.858	0.585	0.845	1.000

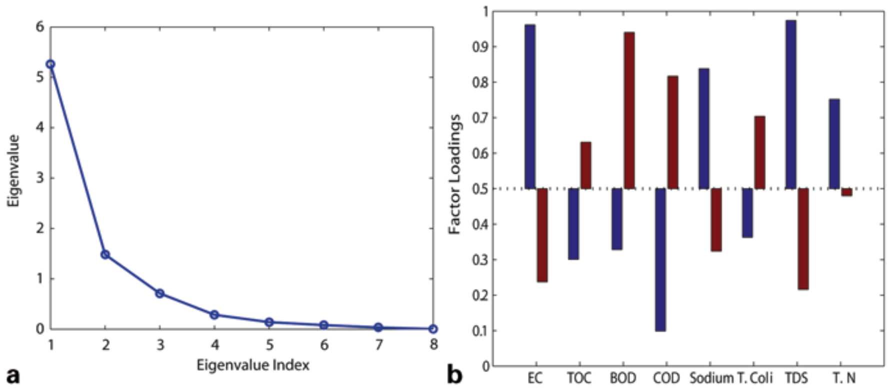


Fig. 6.13 **a** Eigenvalues of the correlation matrix used for FA. In this case, the number of factors is equal to the number of eigenvalues > 1. **b** The factor loadings **L**. A baseline is added to highlight the loading values > 0.5 for both factors

can be considered as indicators of the inorganic contaminants, whereas the second group is representative of organic contaminants. The findings here also have economic implications: instead of monitoring all pollutants, one can set up continuous monitoring of a representative parameter from each group as indicators of water quality and, therefore, achieve sample reduction and cost savings. ■

The discussion so far may suggest that FA is more of a data modeling approach than a method for parameterizing a single random variable per se. However, FA is a method of reducing model structures because it manifests the intricate relationship among samples of different random variables in terms of a small set of latent variables.

Note that FA is meant for exploratory analyses and the meanings of factors may not always be obvious in some applications. The analyst should also keep in mind some main limitations of FA, namely, its linearity, potential identification ambiguity, and reliance on distribution normality. The linearity assumption may be lifted by replacing the linear model used in (6.3.7) with a nonlinear model, provided that the associated identification and distribution normality are handled correspondingly. This leads to the general topic of nonlinear parameterization and dimension-reduction methods.

6.3.3 *Dimension Reduction for Inversion*

So far we have described PCA and FA, which are common tools of linear dimension reduction and have found a wide range of applications in optimal design and control, imaging processing, and also in the solution of inverse problems. When modeling a large-scale distributed system, the inverse solution could become extremely difficult because of (1) the high dimensionality of the estimated parameter and (2) the high dimensionality of the forward model. The former may cause the inverse solution to be ill-posed in nature, while the latter may cause the inverse solution to be computationally prohibitive. For these cases, it is critical to choose an appropriate parameterization and dimension reduction method, as we have alluded to at the beginning of this chapter. In this section, we show how the SVD-based PCA (or POD) may help to achieve such a goal during inversion.

6.3.3.1 Reducing the Dimension of a Parameter Vector

Let us start with the linear inverse problem introduced in Sect. 2.2

$$\mathbf{G}\boldsymbol{\theta} = \mathbf{d}, \tag{6.3.13}$$

where $\mathbf{G}$ is an $n \times m$ matrix, $\boldsymbol{\theta}$ is an m-dimensional parameter vector to be estimated, and $\mathbf{d}$ is an n-dimensional data vector. Using TSVD introduced in Sect. 2.2.3, we can approximate (6.3.13) as

$$\mathbf{U}_k \boldsymbol{\Sigma}_k \mathbf{V}_k^T \hat{\boldsymbol{\theta}} = \mathbf{d}, \tag{6.3.14}$$

where $\boldsymbol{\Sigma}_k$ is a $k \times k$ diagonal matrix that contains only the k largest singular values $s_1 \geq s_2 \geq \cdots \geq s_k$ and all other small singular values starting from s_{k+1} are truncated; $\mathbf{V}_k$ is an $m \times k$ matrix and its rows are the first k orthogonal basis vectors (PCs) spanning the parameter space; $\mathbf{U}_k$ is an $n \times k$ matrix and its columns are the first k orthogonal basis vectors spanning the data space; and the m-dimensional vector $\hat{\boldsymbol{\theta}}$ is a pseudo-inverse solution when the first k singular values are retained. Equation (6.3.14) can be rewritten as

$$\mathbf{U}_k \boldsymbol{\Sigma}_k \boldsymbol{\xi} = \mathbf{d}, \tag{6.3.15}$$

where $\boldsymbol{\xi} = \mathbf{V}_k^T \hat{\boldsymbol{\theta}}$ is the projection of $\hat{\boldsymbol{\theta}}$ onto a subspace spanned by the first k PCs. The problem of estimating the m-dimensional parameter $\boldsymbol{\theta}$ now is reduced to the estimation of k-dimensional parameter $\boldsymbol{\xi}$, and the solution of $\boldsymbol{\xi}$ is more stable than the solution of $\boldsymbol{\theta}$ for reasons mentioned under Sect. 2.2. After $\boldsymbol{\xi}$ is solved from (6.3.15), we have $\hat{\boldsymbol{\theta}} = \mathbf{V}_k \boldsymbol{\xi}$. Note that $\boldsymbol{\xi}$ no longer has the original physical meaning of $\boldsymbol{\theta}$, and $\hat{\boldsymbol{\theta}}$ is only an approximation of $\boldsymbol{\theta}$ because of the truncation.

If a nonlinear inverse problem can be linearized (see Sect. 2.2.4), the matrix $\mathbf{G}$ in (6.3.13) becomes $\mathbf{J}_D^T \mathbf{J}_D$, where $\mathbf{J}_D$ is the Jacobian of the nonlinear model. The above SVD-based PCA then can be used to reduce the dimension of the estimated parameter. But, the linearization error may be amplified after dimension reduction.

6.3.3.2 Reducing the Dimension of Model States

Considering a nonlinear dynamic model governed by a set of ODEs

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, \boldsymbol{\theta}), \tag{6.3.16}$$

where $\mathbf{u}(t) \in \mathbb{R}^N$ is the model outputs and $\boldsymbol{\theta}$ is the model parameters. This equation can also be seen as a semidiscrete form of transient PDEs after spatial discretization, in which N is the number of grid nodes (see Chap. 1). Now, we want to reduce the dimension of model output from N to $k \ll N$ such that the forward solution of the model can be obtained by less computational effort and the efficiency of the inverse solution can be improved.

By solving the forward problem (6.3.16), we collect the model state outputs (called snapshots) at L different times and form an $L \times N$ data matrix, with each row representing a snapshot of the model

$$\mathbf{X} = \left[\mathbf{u}(t_1), \mathbf{u}(t_2), \cdots, \mathbf{u}(t_L)\right]^T. \tag{6.3.17}$$

Applying TSVD to **X**, we have

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \approx \mathbf{U}_k\boldsymbol{\Sigma}_k\mathbf{V}_k^T, (k << N). \tag{6.3.18}$$

Let $\hat{\mathbf{u}}(t) = \mathbf{V}_k\boldsymbol{\xi}(t)$, $\boldsymbol{\xi} \in \mathbb{R}^k$, be an approximate solution of the model and substitute it into (6.3.16), we obtain a reduced set of model equations

$$\frac{d\boldsymbol{\xi}}{dt} = \mathbf{V}_k^T\mathbf{f}(\mathbf{V}_k\boldsymbol{\xi}, \boldsymbol{\theta}). \tag{6.3.19}$$

After $\boldsymbol{\xi}$ is solved from this smaller set of equations, we can use $\hat{\mathbf{u}}(t)$ as the model outputs to calculating the fitting residuals for inversion. However, the efficiency of the approach is a problem: the above model reduction process must be repeated each time after the unknown parameter $\boldsymbol{\theta}$ is updated because the state matrix $\mathbf{X}$ is a function of $\boldsymbol{\theta}$. The TSVD solution in (6.3.18) may be sensitive to the change of $\boldsymbol{\theta}$.

In order to find a robust reduced-dimensional model for all possible parameter values in an admissible region, the snapshots should be generated with different parameter samples $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_S)$ taken from the region, leading to an $LS \times N$ matrix

$$\mathbf{X} = \left[\mathbf{u}(t_1, \boldsymbol{\theta}_1), \cdots, \mathbf{u}(t_L, \boldsymbol{\theta}_1), \cdots, \cdots, \cdots, \mathbf{u}(t_1, \boldsymbol{\theta}_S), \cdots, \mathbf{u}(t_L, \boldsymbol{\theta}_S)\right]^T. \tag{6.3.20}$$

TSVD is then used to find the PCs of this dataset. Detailed descriptions on POD-based inversion process can be found in (Cardoso et al. 2009; Kaleta et al. 2011; Burkardt et al. 2006). Effective algorithms of reducing the forward solution dimension for inversion and other applications in groundwater hydrology can be found in (McPhee and Yeh 2008; Pasetto et al. 2013).

6.3.3.3 Reducing the Dimension of a Random Field

Let $\theta(\mathbf{x})$ be a random function defined in a spatial region (Ω). Its Karhunen–Loève (KL) expansion is given by (Ghanem and Spanos 1991)

$$\theta(\mathbf{x}) = \bar{\theta}(\mathbf{x}) + \sum_{i=1}^{\infty} \xi_i \sqrt{\lambda_i}\phi_i(\mathbf{x}), \tag{6.3.21}$$

where $\bar{\theta}(\mathbf{x})$ is the mean, λ_i and $\phi_i(\mathbf{x})$ are the eigenvalue and eigenfunctions of the covariance function of $\theta(\mathbf{x})$, and the set $\{\xi_i\}$ are orthonormal random variables with zero mean and unit covariance

$$\langle \xi_i \rangle = 0, \langle \xi_i \xi_j \rangle = \delta_{ij} \tag{6.3.22}$$

where δ_{ij} is Kronecker delta function. Comparing to (6.1.1), we see that (6.3.21) is a special series expansion of a distributed function. The significance of the KL expansion is that it expresses a random function using two orthogonal sets, one consisting of random variables $\{\xi_i\}$ and the other of deterministic basis functions $\{\phi_i(\mathbf{x})\}$.

KL expansion (6.3.21) can be obtained from the covariance function. The covariance of $\theta(\mathbf{x})$ between any two points $\mathbf{x}$ and $\mathbf{x}'$, $C(\mathbf{x},\mathbf{x}')$, has the following spectral or eigendecomposition

$$C(\mathbf{x},\mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x})\phi_i(\mathbf{x}') \tag{6.3.23}$$

and λ_i and $\phi_i(\mathbf{x})$ are the solution of the Fredholm integral equation of the second kind given by

$$\int_{\Omega} C(\mathbf{x},\mathbf{x}')\phi_i(\mathbf{x}')d\mathbf{x}' = \lambda_i \phi_i(\mathbf{x}),\ i = 1,2,\cdots \tag{6.3.24}$$

After $C(\mathbf{x},\mathbf{x}')$ is given, (6.3.24) can be solved either analytically or numerically (see Example 6.5), and the KL expansion of $\theta(\mathbf{x})$ is thus obtained.

We are especially interested in the truncated form of KL expansion because it provides a reduced-dimension parameterization. Let $\boldsymbol{\theta} \in \mathbb{R}^N$ be the discrete expression of $\theta(\mathbf{x})$ associated with a grid of *N* nodes. We want to represent the random field using the first *k* terms of KL expansion. The above equations, (6.3.21), (6.3.23), and (6.3.24), can be rewritten, respectively, as

$$\boldsymbol{\theta} \approx \overline{\boldsymbol{\theta}} + \mathbf{Q}_k \boldsymbol{\Lambda}_k^{1/2} \boldsymbol{\xi} \tag{6.3.25}$$

$$\mathbf{C} \approx \mathbf{Q}_k \boldsymbol{\Lambda}_k \mathbf{Q}_k^T \tag{6.3.26}$$

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{6.3.27}$$

where $\mathbf{v}_i = [\phi_i(\mathbf{x}_1),\phi_i(\mathbf{x}_2),\cdots,\phi_i(\mathbf{x}_N)]^T$ is the ith eigenvector of covariance matrix $\mathbf{C}$ and λ_i is the corresponding eigenvalue. Equation (6.3.26) is the truncated eigenvalue decomposition of $\mathbf{C}$, where the $N \times k$ matrix $\mathbf{Q}_k$ has *k* eigenvectors $\{\mathbf{v}_i\}$ (PCs) as its columns. In other words, we obtain the result of PCA again. As in PCA, by taking *n* samples of the random vector $\boldsymbol{\theta}$ and making the data centered (i.e., removing the mean), all PCs of the covariance matrix can be estimated by TSVD, and (6.3.25) gives a parameterization or a dimension-reduced approximation

$$\boldsymbol{\theta} \approx \hat{\boldsymbol{\theta}} \equiv \mathbf{Q}_k \boldsymbol{\Lambda}_k^{1/2} \boldsymbol{\xi}, \tag{6.3.28}$$

where $\hat{\boldsymbol{\theta}} \in \mathbb{R}^N$ but $\boldsymbol{\xi} \in \mathbb{R}^k$. Equation (6.3.28) can be considered as a random field generator: we can generate a vector of standard normal random variates $\boldsymbol{\xi}$ and use it to generate a realization of random field $\hat{\boldsymbol{\theta}}$, which can then be used as model input for inverse solution or other purposes. For example, such application of KL expansion is behind stochastic ensemble methods to be introduced in Chap. 9.

There are, however, two issues associated with this KL expansion method. First, the number of nodes (N) may be very large; as a result, performing SVD to find the PCs of an $N \times N$ sample covariance matrix may become too expensive, if not impossible. Second, the distribution of $\theta(\mathbf{x})$ may not be Gaussian, but the distribution of $\hat{\boldsymbol{\theta}}$ is always Gaussian; therefore, the approximated random fields are overly smooth. To circumvent these issues, we need the Kernel PCA (KPCA) and nonlinear dimension reduction techniques to be introduced in the next section.

Example 6.5 *KL Expansion*
For certain covariance function families, the solution to the eigenvalue problem (6.3.24) can be obtained analytically. For example, let us consider a one-dimensional random process with exponential covariance

$$C_{\Phi}(x,x') = \sigma^2 \int_L \exp\left(-\frac{|x-x'|}{I}\right)\phi(x')dx', \tag{6.3.29}$$

where σ^2 and I are variance and integral scale, respectively; $\phi(x)$ is eigenfunction; and L is the domain length. The eigenvalues and eigenfunctions can be found from the following two equations (Zhang and Lu 2004)

$$\lambda_i = \frac{2I\sigma^2}{I^2 w_i^2 + 1}, \tag{6.3.30}$$

and

$$\phi_i(x) = \frac{1}{\sqrt{(I^2 w_i^2 + 1)L/2 + I}}[Iw_i \cos(w_i x) + \sin(w_i x)], \tag{6.3.31}$$

where w_i are roots of the characteristic equation

$$(I^2 w^2 - 1)\sin(wL) = 2Iw\cos(wL). \tag{6.3.32}$$

Equation (6.3.32) has an infinite number of positive roots, which can be sorted in ascending order to get eigenvalues in the descending order.

The results are illustrated numerically using $L = 10$, $\sigma^2 = 1$, and $I = 1$. In this case, the first four roots of the characteristic equation are 0.2628, 0.5307, 0.8067, and 1.0909, respectively, in ascending order. Figure 6.14a shows the corresponding eigenfunctions, and Fig. 6.14b compares the covariance function approximated using the first 4, 10, and 50 terms of the KL expansion. The true covariance function is shown with the solid line. The long-range portion of the covariance curve (i.e., the low-frequency part in the frequency domain) is better approximated than the short-range portion. As the number of KL expansion terms increases, the approximation of the short-range part gets better, and the corresponding RMSE of the approximation is 0.15, 0.058, and 0.009, respectively.

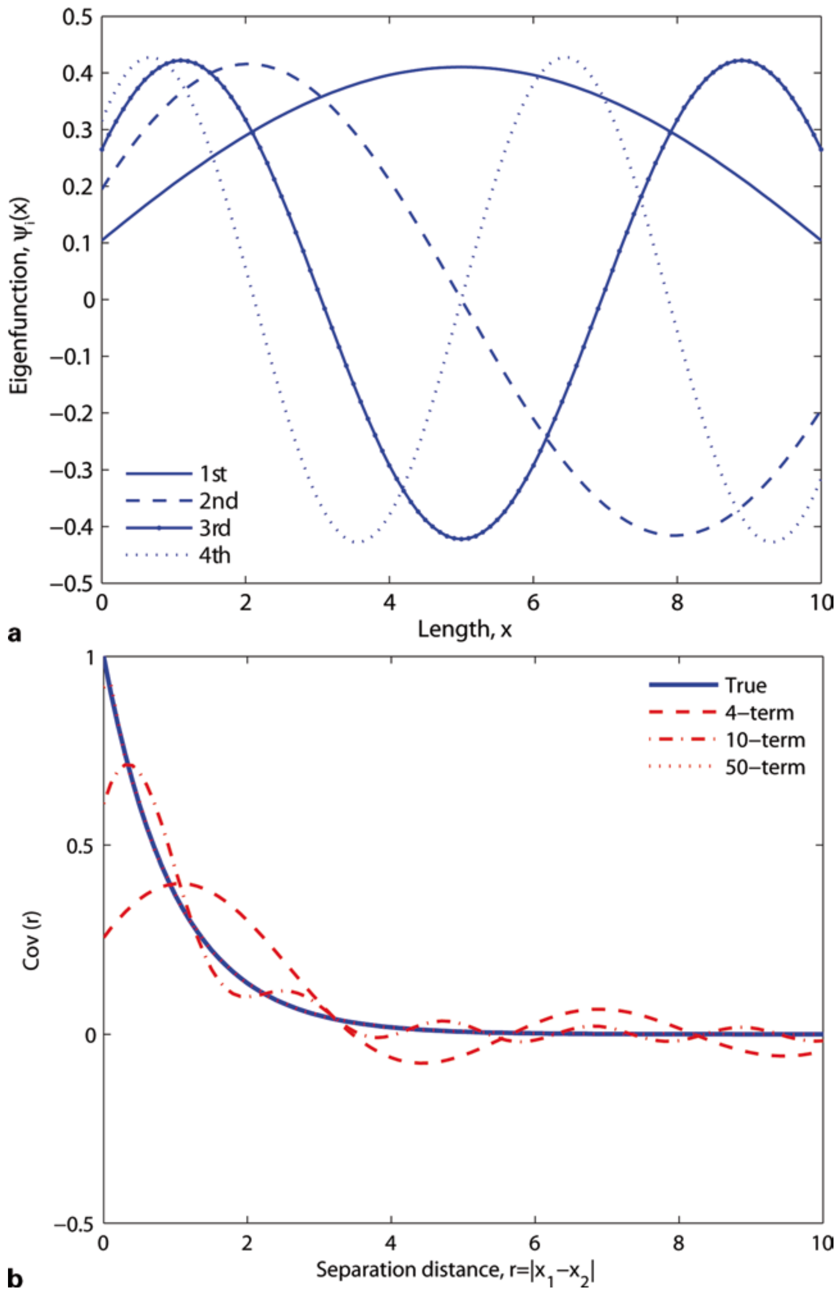


Fig. 6.14 **a** The first four eigenfunctions corresponding to a one-dimensional random process with exponential covariance function, $L = 10$, $\sigma^2 = 1$, and $I = 1$. **b** The covariance function approximated by the first 4, 10, and 50 terms of KL expansion

6.4 Nonlinear Dimension Reduction

The dimension reduction methods (i.e., PCA) described in the previous section all assume a linear mapping between the sample space and that of the lower-dimensional parameterization. This is not always true. There are situations in which such a linearity assumption can lead to inaccurate representation of the raw data and/or ineffective reduction in data dimension. One example that PCA does not work is parameterization of multimodal permeability random fields (Sarma et al. 2008). Another example is the reduction of remotely sensed data, such as hyperspectral images. Parameterization of these problems requires not only finding a low-dimensional representation of high-dimensional data, but also preserving certain characteristics of the original data (e.g., geometry of facies distributions) which are not linear. The significance of such low-dimensional space can be rather great in the context of inverse problems because we no longer need to be limited by the apparent high-dimensionality of many problems. In mathematics, a *manifold* is a topological space that resembles the Euclidean space of a specific dimension on a small enough scale. The identification of a small set of parameters that uniquely define a manifold embedded in a high-dimensional space is called *manifold learning*.

Many algorithms have been developed for nonlinear manifold learning, including kernel PCA, isomap, locally linear embedding, Laplacian Eigenmaps, Sammon mapping, and multilayer autoencoders (van der Maaten 2007). Not all of the algorithms are designed equal and their performance heavily depends on the underlying problem. It remains an active research area to choose the most effective algorithm for nonlinear dimension reduction in different EWR applications. In this section, we shall focus on several commonly used nonlinear dimension reduction techniques.

6.4.1 Kernel PCA

KPCA is a nonlinear version of PCA. It was originally introduced in the field of machine learning (Schölkopf et al. 1998) and later used as a parameterization method for inverse solution (Sarma et al. 2008). KPCA assumes the existence of a mapping that transforms the data from its original space to a high- or even infinite-dimensional space called the *feature space*. Through this projection, the hope is to "unfold" the otherwise nonlinear data into linear ones such that PCA can be applied.

Let $\boldsymbol{\theta}$ be an m-dimensional random vector, where m is a very large number, and $[\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_n]$ be the data matrix formed by n samples of $\boldsymbol{\theta}$, where $n \ll m$. As usual, we assume that the sample mean has been removed from the data and the $m \times m$ sample covariance matrix is given by

$$\mathbf{C} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{\theta}_j \boldsymbol{\theta}_j^{\mathbf{T}}. \tag{6.4.1}$$

In PCA, all eigenvectors $\mathbf{v}$ (PCs) are solved from the eigenvalue equation

$$\lambda \mathbf{v} = \mathbf{C}\mathbf{v}. \tag{6.4.2}$$

This equation is too expensive or even impossible to solve when the size of $\mathbf{C}$ ($m \times m$) is too large. As mentioned in Sect. 6.3.3, this is the first difficulty of using PCA to parameterize a random field. In what follows, we will see that with KPCA the solution of (6.4.2) is reduced to a low-dimensional eigenvalue problem.

Because the data matrix $[\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_n]$ is an $m \times n$ rectangular matrix, the number of its singular values is at most $\min(n, m) = n$. In other words, the maximum number of nonzero eigenvectors of matrix (6.4.1) is *n*. This fact has two nice consequences: first, all eigenvectors $\mathbf{v}$ with nonzero eigenvalues must lie in the span of $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_n\}$, namely, there exists a set of coefficients $\{\alpha_j\}$ such that each eigenvector can be expressed as a linear combination of the sample vectors

$$\mathbf{v} = \sum_{j=1}^{n} \alpha_j \boldsymbol{\theta}_j. \tag{6.4.3}$$

Second, (6.4.2) can be represented by the following inner product form

$$\lambda(\boldsymbol{\theta}_l \cdot \mathbf{v}) = (\boldsymbol{\theta}_l \cdot \mathbf{C}\mathbf{v}), \quad l = 1, 2, \cdots, n \tag{6.4.4}$$

where $\mathbf{C}\mathbf{v} = (1/n)\sum_{j=1}^{n} (\boldsymbol{\theta}_j \cdot \mathbf{v})\boldsymbol{\theta}_j$. Substituting (6.4.3) into (6.4.4) and after some manipulations (Schölkopf et al. 1998), we get a so-called kernel eigenvalue problem

$$n\lambda \boldsymbol{\alpha} = \mathbf{K}\boldsymbol{\alpha}, \tag{6.4.5}$$

where $\mathbf{K}$ is an $n \times n$ kernel matrix with elements $K_{ij} = (\boldsymbol{\theta}_i \cdot \boldsymbol{\theta}_j)$, vector $\boldsymbol{\alpha}$ is an eigenvector of $\mathbf{K}$ and its components α_j are used in the expansion (6.4.3), and $\delta = n\lambda$ is the associated eigenvalue. Once $\boldsymbol{\alpha}$ and δ are obtained by solving the reduced dimensional problem (6.4.5), the original eigenvector $\mathbf{v}$ of $\mathbf{C}$ can be obtained by substituting $\boldsymbol{\alpha}$ into (6.4.3), and its associated eigenvalue is equal to $\lambda = \delta / n$. Solving the kernel eigenvalue equation (6.4.5) is much easier than solving the original eigenvalue equation (6.4.2). For example, in an example given by Sarma et al. (2008), the hydraulic conductivity is distributed over 2500 nodes. The size of matrix $\mathbf{C}$ in (6.4.2) is 2500×2500, but the covariance can be recovered with no more than 300 samples, and thus, the size of matrix $\mathbf{K}$ in (6.4.5) is less than 300×300. The computational effort of PCA is decreased significantly.

Next, we will derive KPCA and use it to parameterize random fields characterized by multipoint statistics. As mentioned at the beginning of this section, KPCA assumes that a nonlinear mapping $\boldsymbol{\Phi} = \boldsymbol{\Phi}(\boldsymbol{\theta})$ exists that transforms $\boldsymbol{\theta} \in \mathbb{R}^N$ to $\boldsymbol{\Phi} \in \mathbb{F}$, where $\mathbb{F}$ is a high-dimensional feature space.

With the inner product form introduced herein, it is straightforward to derive PCA for the nonlinear mapping $\boldsymbol{\Phi}$ in $\mathbb{F}$. Let the dataset $\{\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, \cdots, \boldsymbol{\Phi}_n\}$ be the low-dimensional projection of the centered sample dataset $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_n\}$, where $\boldsymbol{\Phi}_i = \boldsymbol{\Phi}(\boldsymbol{\theta}_i)$ for $i = 1, 2, \cdots, n$. The sample covariance matrix of $\boldsymbol{\Phi}$ is then given by

$$\mathbf{C}_{\boldsymbol{\Phi}} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{\Phi}_j \boldsymbol{\Phi}_j^T. \tag{6.4.6}$$

Repeat the derivation of equations (6.4.1) to (6.4.5) but replacing $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_n\}$ with $\{\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, \cdots, \boldsymbol{\Phi}_n\}$, we arrive at another eigenvalue equation

$$\delta \boldsymbol{\alpha} = \mathbf{K}\boldsymbol{\alpha} \tag{6.4.7}$$

where the $n \times n$ kernel matrix $\mathbf{K}$ has elements $K_{ij} = (\boldsymbol{\Phi}_i \cdot \boldsymbol{\Phi}_j)$, $\boldsymbol{\alpha}$ is an eigenvector of $\mathbf{K}$, and δ is its corresponding eigenvalue. After all eigenvectors $\{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \cdots, \boldsymbol{\alpha}_n\}$ and eigenvalues $\{\delta_1, \delta_2, \cdots, \delta_n\}$ of $\mathbf{K}$ are solved from this equation, the eigenvectors (PCs) and corresponding eigenvalues of $\mathbf{C}_{\boldsymbol{\Phi}}$ are given, respectively, by

$$\mathbf{v}_l = \sum_{j=1}^{n} \alpha_{l,j} \boldsymbol{\Phi}_j \text{ and } \lambda_l = \frac{\delta_l}{n} \quad l = 1, 2, \cdots, n. \tag{6.4.8}$$

We now have completed the derivation of PCA in the feature space on $\boldsymbol{\Phi}(\boldsymbol{\theta})$. The following parameterization of $\boldsymbol{\Phi}(\boldsymbol{\theta})$ is obtained

$$\hat{\boldsymbol{\Phi}} = \mathbf{Q}\boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi}, \tag{6.4.9}$$

where matrix $\mathbf{Q}$ has $\mathbf{v}_l$ given in (6.4.8) as its columns, the diagonal matrix $\boldsymbol{\Lambda}$ has λ_l given in (6.4.8) as its entries, and the *n*-dimensional vector $\boldsymbol{\xi}$ is the reduced-dimension parameter. Note that in the above derivation of KPCA, the actual form of $\boldsymbol{\Phi}$ never needs to be known explicitly because all that is required is the functional form of its inner product. This is known as the *kernel trick* in machine learning and is the building block of several machine learning algorithms, as we shall see in Chap. 8.

Using (6.4.8), parameterization (6.4.9) can be rewritten as

$$\hat{\boldsymbol{\Phi}} = \sum_{i=1}^{n} \beta_i \boldsymbol{\Phi}_i, \text{ where } \beta_i = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} \alpha_{ij} \delta_j^{1/2} \xi_j. \tag{6.4.10}$$

As we have explained in Sect. 6.3.3, $\hat{\boldsymbol{\Phi}}$ depends only on the second-order moment of the data statistics. In order to make it contain the information of higher order moments, we have to change the structure of the kernel matrix $\mathbf{K}$ by introducing a kernel function $\kappa(\mathbf{x}, \mathbf{y})$ and define the form inner product in $\mathbb{F}$ as $(\boldsymbol{\Phi}_i \cdot \boldsymbol{\Phi}_j) = \kappa(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$. As a result, the solution $\boldsymbol{\alpha}$ and δ of (6.4.7) will change accordingly. Finally, (6.4.9) becomes a parameterization of $\boldsymbol{\Phi}(\boldsymbol{\theta})$ generated by the KPCA.

Example 6.6 *Polynomial Kernels*
One of the most commonly used kernel functions is the polynomial kernel $\kappa(\mathbf{x},\mathbf{y}) = (\mathbf{x}^T\mathbf{y}+1)^p$, where $\mathbf{x}$ and $\mathbf{y}$ are vectors in the same input space and p is the order of polynomial kernel. Consider the following second-order polynomial kernel in $\mathbb{R}^2$

$$\kappa(\mathbf{x},\mathbf{y}) = (\mathbf{x}^T\mathbf{y}+1)^2$$

where $\mathbf{x} = (x_1, x_2)^T$ and $\mathbf{y} = (y_1, y_2)^T$. The kernel function can be decomposed as the inner product of two mappings

$$\begin{aligned}\kappa(\mathbf{x},\mathbf{y}) &= (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)(1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, \sqrt{2}y_1y_2, y_2^2)^T \\ &= \mathbf{\Phi}(\mathbf{x})^T\mathbf{\Phi}(\mathbf{y})\end{aligned}$$

in which the mapping $\mathbf{\Phi}$ transforms data from the two-dimensional input space into a 6-dimensional feature space. It is not hard to imagine that such decomposition quickly becomes infeasible as the order of polynomial or the dimension of input data grows higher. The kernel trick makes it possible to solve problems in the six-dimensional space without explicitly transforming the input data into that space.

More complex kernels can be constructed out of simple kernels. Assuming we have valid kernels $\kappa_1(\mathbf{x},\mathbf{y})$ and $\kappa_2(\mathbf{x},\mathbf{y})$, then the following operations are valid and can create a new kernel κ out of κ_1 and κ_2

$$\begin{aligned}\kappa(\mathbf{x},\mathbf{y}) &= c\kappa_1(\mathbf{x},\mathbf{y}) \\ \kappa(\mathbf{x},\mathbf{y}) &= \kappa_1(\mathbf{x},\mathbf{y}) + \kappa_2(\mathbf{x},\mathbf{y}) \\ \kappa(\mathbf{x},\mathbf{y}) &= \kappa_1(\mathbf{x},\mathbf{y})\kappa_2(\mathbf{x},\mathbf{y}) \\ \kappa(\mathbf{x},\mathbf{y}) &= \exp(\kappa_1(\mathbf{x},\mathbf{y}))\end{aligned}$$

in which c is a constant. More examples of valid kernel operations can be found in Bishop (2006). In addition to polynomial kernel, Gaussian kernel introduced in RBF (Sect. 6.1.5) is also widely used in KPCA. Other examples of the kernel functions can be found in (Sarma et al. 2008; Schölkopf et al. 1998).

For inverse solution, our purpose is to find a parameterization of the original parameter $\boldsymbol{\theta}$. Finding $\boldsymbol{\theta}$ from $\mathbf{\Phi}(\boldsymbol{\theta})$ (called the pre-image problem) is an ill-posed inverse problem, because a unique inverse mapping $\mathbf{\Phi}^{-1}$ may not exist. In this case, we can only find a quasisolution by solving a least-squares problem

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \rho(\boldsymbol{\theta}), \text{ where } \rho(\boldsymbol{\theta}) = \left\|\mathbf{\Phi}(\boldsymbol{\theta}) - \hat{\mathbf{\Phi}}\right\|^2. \qquad (6.4.11)$$

Expanding the norm in (6.4.11) and using (6.4.10), we have

$$\begin{aligned}\rho(\boldsymbol{\theta}) &= \boldsymbol{\Phi}(\boldsymbol{\theta})\cdot\boldsymbol{\Phi}(\boldsymbol{\theta}) - 2\boldsymbol{\Phi}(\boldsymbol{\theta})\cdot\hat{\boldsymbol{\Phi}} + \hat{\boldsymbol{\Phi}}\cdot\hat{\boldsymbol{\Phi}} \\ &= \kappa(\boldsymbol{\theta},\boldsymbol{\theta}) - 2\sum_{i=1}^{n}\beta_i\kappa(\boldsymbol{\theta}_i,\boldsymbol{\theta}) + \sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\beta_j\kappa(\boldsymbol{\theta}_i,\boldsymbol{\theta}_j).\end{aligned} \tag{6.4.12}$$

The optimal solution $\hat{\boldsymbol{\theta}}$ must satisfy the necessary condition of minimum. When the kernel function is given by the *p*-order inner product, the following equation can be derived from $\partial\rho/\partial\boldsymbol{\theta} = \mathbf{0}$ (Schölkopf et al. 1998):

$$\sum_{i=1}^{n}\beta_i\sum_{j=1}^{p} j(\boldsymbol{\theta}_i\cdot\hat{\boldsymbol{\theta}})^{j-1}\hat{\boldsymbol{\theta}} - \sum_{i=1}^{n}\beta_i\sum_{j=1}^{p} j(\boldsymbol{\theta}_i\cdot\hat{\boldsymbol{\theta}})^{j-1}\boldsymbol{\theta}_i = \mathbf{0}. \tag{6.4.13}$$

Prior information on the parameter structure can be incorporated into the samples and the iteration process of solution. The steps of KPCA parameterization process can be summarized as follows:

- Generate *n* realizations (or samples) $\{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\cdots,\boldsymbol{\theta}_n\}$ of the random field $\theta(\mathbf{x})$
- Select a kernel function $\kappa(\cdot,\cdot)$ and form the kernel matrix **K**
- Solving eigenvalue equation (6.4.7) for $\{\boldsymbol{\alpha}_i\}$ and $\{\delta_i\}$ ($i = 1,2,\cdots,n$)
- Calculate $\{\beta_i\}$ ($i = 1,2,\cdots,n$) by (6.4.10)
- Draw an *n*-dimensional vector $\boldsymbol{\xi}$
- Solve (6.4.13) to obtain $\hat{\boldsymbol{\theta}} = f(\boldsymbol{\xi})$ when $\kappa(\cdot,\cdot)$ is a *p*-order inner product

Note that we can further decrease the dimension of parameterization from *n* to *k* by truncating the eigenvectors associated with small eigenvalues. The above KPCA nonlinear parameterization technique was used by Sarma et al. (2008) for modeling permeability fields characterized by multipoint statistics.

The performance of KPCA is affected by the a priori specified kernel functions. Maximum variance unfolding (MVU) or semidefinite embedding attempts to overcome this issue by "learning" kernel matrices, in which a semidefinite programming problem is solved to yield elements of kernel matrix (Weinberger and Saul 2006).

6.4.2 Laplacian Eigenmaps

If PCA and KPCA are full spectral methods that involve eigendecomposition of a full covariance matrix, other algorithms seek to construct sparse matrices and, therefore, are easier to scale to large datasets. These include local linear embedding (Roweis and Saul 2000) and Laplacian Eigenmaps (Belkin and Niyogi 2003). Because of the similarity between the two algorithms, we will only describe the Laplacian Eigenmap algorithm, which constructs a low-dimensional representation of the input data by minimizing the distance between data points and their neighbors. Thus, Laplacian Eigenmaps may be viewed as a local method that attempts to preserve the neighborhood information during dimension reduction.

Given the input set $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_n\}$ in $\mathbb{R}^m$, the Laplacian Eigenmap algorithm finds a reduced-dimension set. Laplacian Eigenmap proceeds in three steps. First, a neighborhood is defined for each input data point, which can be based on either geometric distance or simply the l nearest neighbors. Here, the size of neighborhood is specified by the user.

Second, a weight matrix $\mathbf{W}$ of dimensions $n \times n$ is defined on the input data using the neighborhoods identified in the previous step. Two popular choices for $\mathbf{W}$ are:

- Nearest neighbor

$$w_{ij} = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are neighbors} \\ 0, & \text{otherwise} \end{cases}. \tag{6.4.14}$$

- Gaussian kernel

$$w_{ij} = \begin{cases} \mathrm{e}^{-\left\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\right\|^2 / 2\sigma^2}, & \text{if } i \text{ and } j \text{ are neighbors} \\ 0, & \text{otherwise} \end{cases}. \tag{6.4.15}$$

Thus, by way of construction, $\mathbf{W}$ measures local similarity among data points and is usually highly sparse. For example, if $\mathbf{W}$ is defined via the nearest neighbor method, each row of $\mathbf{W}$ only contains l nonzero elements and l is usually smaller than the number of data, n. The Gaussian kernel in (6.4.15) plays the same role as the kernel functions used in RBF and KPCA do.

In the third step, a reduced-dimension set is constructed. Let $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$ denote a vector and we want to project the original dataset onto $\mathbf{y}$ such that the points in $\mathbf{y}$ stay as close as possible. This is equivalent to solving the following minimization problem to minimize distances between neighboring nodes

$$\min \frac{1}{2} \sum_{i,j} (y_i - y_j)^T w_{ij} (y_i - y_j), \text{ or } \min_{\mathbf{y}} \mathbf{y}^T \mathbf{L} \mathbf{y} \tag{6.4.16}$$

in which $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the so-called Laplacian matrix and $\mathbf{D}$ is a diagonal matrix with elements $D_{ii} = \sum_j w_{ij}$. It can be shown that the solution can be obtained by solving the following generalized eigenvalue problem (Belkin and Niyogi 2003)

$$\mathbf{L}\mathbf{y} = \lambda \mathbf{D}\mathbf{y}. \tag{6.4.17}$$

Let the smallest k nonzero eigenvalues obtained from (6.4.17) be

$$\lambda_1 < \cdots < \lambda_k,$$

and denote the corresponding eigenvectors as $\{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_k\}$. This set of eigenvectors represents the reduced-dimension dataset found by Laplacian Eigenmaps. Like in the case of KPCA (see (6.4.7)), only an $n \times n$ eigenvalue problem needs to be solved, which is much easier than dealing with the eigenvalue problem in original, high-dimensional space. In (6.4.17), $\mathbf{L}$ is sparse and $\mathbf{D}$ is diagonal. The computational complexity of Laplacian Eigenmap is $O((l+k)n^2)$, which is generally much less than the full spectral method such as KPCA. From parameterization perspective, any new point in the reduced-dimension space can be expressed in terms of $\{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_k\}$ and the weight matrix; however, projecting the new point to the original space requires solving an ill-posed, pre-image problem as it is done for KPCA. Laplacian Eigenmap has been used for information extraction from hyperspectral remote sensing data (Qian and Chen 2007).

6.5 Review Questions

1. What is the dimension of a model? Give examples of finite- and infinite-dimensional models. Why do we need model dimension reduction?
2. According to the criteria presented at the end of Sect. 6.1.2, is the IDW method suitable for parameterizing a distributed function to be identified? Explain how it is used to the three cases described in Sect. 6.1.2.
3. According to the general expression of function approximation given in equation (6.1.3), show the basis functions $\{\phi_j\}$ and coefficients $\{c_j\}$ for the three methods, IDW, nearest-neighbor, and RBF.
4. What are the differences between parameterizing a deterministic function and parameterizing a random field?
5. What are the dimension (the number of DOF) of a GRF model and the dimension of a MRF model under different assumptions?
6. How is the variogram of a random field defined and estimated? How do we choose a variogram model? And, how do we find whether or not a random field can be regarded as an intrinsically stationary one?
7. Give the details of deriving simple kriging equations (6.2.38) and (6.2.39).
8. Design an example from your study area, and get familiar with a geostatistical software package to perform ordinary kriging and regression kriging. Can we use the ordinary kriging to estimate a nonintrinsically stationary random field?
9. Express PCA by the general form of parameterization given in equation (6.1.3).
10. What are the advantages and disadvantages of using SVD-based PCA to reduce the dimension of the unknown parameter vector for model inversion?
11. Explain how KL expansion can be used to approximate a random field.

Chapter 7
Model Structure Identification

In the CIP, it is assumed that model errors can be neglected and only model parameters need to be estimated from observed data. In practice, determination of the model structure is the first problem that a modeler has to deal with when modeling a complicated physical system. This problem is referred to as "system identification" in control theory. It requires estimation of the model structure and model parameters simultaneously from input–output data. In EWR modeling, a conceptual model with given structure is often provided by an expert based on his/her knowledge. The unknown parameters of the model are then estimated by inverse solution. If model outputs do not fit observed data well, the model structure is "tweaked" by trial and error. However, such traditional model construction process suffers from the following issues:

- The fitting residual cannot be reduced to a satisfactory level.
- Only very limited model structures can be considered, and the true structure is usually beyond reach because of its complexity.
- It is difficult to determine an appropriate level of model complexity. A model complexity that is appropriate for fitting data may not always be appropriate for other intended model applications such as prediction.
- The problem of identifying model structure and model parameters simultaneously is usually ill-posed, namely, the same dataset can be fit equally well by different model structures with different model parameters.

Because of these issues, model structure error is often the major cause of model failure (Sun et al. 1998). Even a carefully calibrated model may produce unreliable results when used for prediction and decision-making purposes. Bredehoeft (2005) pointed out that "surprises" occur in 20–30 % of groundwater model analyses, in which he defined a surprise as new data or evidence that invalidates a prevailing conceptual model. In a recent white paper published by the US National Science Foundation, model structure identification was listed as one of the grand challenges of future environmental modeling (Beck et al. 2009).

We have shown in Chap. 1 that parametric EWR models are derived from conservative laws and represented by certain types of mathematical equations. Because of a myriad of information is required to build a EWR model and limitations of data in practice, the developed EWR model can be considered as a “gray box” that has uncertainties in its dimensions, internal structures, and external conditions (Refsgaard et al. 2007; Sun 1994; Beven 2009; Chou and Voit 2009). In this chapter, a model will be denoted by $(\mathbf{S}, \boldsymbol{\theta})$, where $\mathbf{S}$ is the model structure and $\boldsymbol{\theta}$ denotes the set of parameters associated with the structure. We will show how both $\mathbf{S}$ and $\boldsymbol{\theta}$ can be parameterized and optimized using existing knowledge and data.

In Sect. 7.1, the model structure $\mathbf{S}$ is parameterized, an *extended inverse problem* (EIP) is formulated for identifying both $\mathbf{S}$ and $\boldsymbol{\theta}$ with state observations and prior information, and the model complexity problem is considered. The EIP is a global optimization problem and is difficult to solve in general. For a distributed parameter, the structure identification problem becomes how to find the optimal parameterization. In Sect. 7.2, techniques and algorithms of adaptive parameterization, such as Voronoi nearest neighbor algorithm, zone refinement, and level set methods, are reviewed. We will show how shape parameters of a structure are defined and identified. The multiscale inversion method introduced in Sect. 7.3 may avoid the difficulty of structure identification and give higher-resolution results, but generally requires more data support.

In the statistical framework, structure identification requires estimation of both shape parameters and statistical parameters. Statistical parameters are hyperparameters that do not appear directly in the model but must be estimated during the inversion process. In Sect. 7.4, the *statistical EIP* is formulated, in which model parameters and hyperparameters are estimated either simultaneously or iteratively by MLE or by hierarchical Bayesian inversion. Geostatistical inversion uses kriging and cokriging as parameterization for estimating a distributed parameter, but is subject to the problem of plausibility. The pilot-point method introduced in that section is a flexible parameterization method for inversion that can decrease the model structure error of a geostatistical model effectively, especially when the number and locations of pilot points (as shape parameters) are optimized.

7.1 Model Structure Parameterization

7.1.1 Model Structure Representation

A model of a system (or *system model*) is characterized by a structure and a set of parameters associated with that structure. All candidate models to be identified for a real system form a set. We assume that the real system can be considered as a member of the set approximately, at least for certain aspects of model applications when the model structure is made sufficiently complex. Intuitively, we would like to find a model that has the most parsimonious structure and yet is reasonably close to the real system under consideration. A major question is then how we can measure the “distance” between two models having different structures. To answer this question, we have to define a *Banach space* (called a *system model space*, or *system space* in short) that contains all feasible system models, as well as the “real system” as its members. For numerical models with space/time discretization, however, we can limit our discussion to the Euclidean space.

7.1.1.1 Define a System Model Space

Assume that the properties of a spatially distributed system are characterized by r unknown or incompletely known parameters $\{\theta_1(\mathbf{x}), \theta_2(\mathbf{x}), \cdots, \theta_r(\mathbf{x})\}$. The problem of system identification then becomes identification of these parameters. If the small-scale variability of a distributed parameter can be neglected, its number of degrees of freedom (DOF) is equal to or less than the node number (N) of the finest grid used for simulating the system. In this case, the system space $\mathbb{M}$ is defined as r stacking N-dimensional spaces, where the ith layer is the parameter space of discretized $\theta_i(\mathbf{x})$ ($i = 1, 2, \cdots, r$). Thus, the DOF of the most complex model structure in $\mathbb{M}$ is $r \times N$. The real system after discretization is an element of $\mathbb{M}$, and all candidate models to be identified for the system consist of a subset $\mathfrak{M}_{ad} \subset \mathbb{M}$.

The difference between two models $\mathcal{M}_A$ and $\mathcal{M}_B$ in system space $\mathbb{M}$ is measured by $\|\mathcal{M}_A - \mathcal{M}_B\|$. In the observation space $\mathbb{F}$ of a state variable u, the difference between the two models is measured by $\|u_D(\mathcal{M}_A) - u_D(\mathcal{M}_B)\|$, where D is an observation design stipulating where and when the state variable u is observed. Therefore, after the norms in space $\mathbb{M}$ and $\mathbb{F}$ are defined, models with different structures can be compared to each other. For example, we can compare a two-dimensional model having a continuous structure with a three-dimensional model having a discontinuous structure.

7.1.1.2 Parameterization of a System Model

The dimension of space $\mathbb{M}$ is often too high for identifying a system model. To make a complex model structure identifiable with limited data, parameterization becomes necessary. From Chap. 6, we have learned that a distributed function can be parameterized by truncating its expansion. When a system model is characterized by r distributed parameters $\{\theta_1(\mathbf{x}), \theta_2(\mathbf{x}), \cdots, \theta_r(\mathbf{x})\}$, we have the following truncated expansion:

$$\theta_i(\mathbf{x}) \approx \hat{\theta}_i(\mathbf{x}) = \sum_{j=1}^{m_i} \theta_{ij}\phi_{ij}(\mathbf{x}, \boldsymbol{\nu}_i), i = 1, 2, \cdots, r \tag{7.1.1}$$

where m_i is the dimension of parameterization of $\theta_i(\mathbf{x})$, $\boldsymbol{\phi}_i = \{\phi_{ij} \mid j = 1, 2, \cdots, m_i\}$ are corresponding basis functions, and $\boldsymbol{\nu}_i$ is a set of shape parameters associated with basis functions $\boldsymbol{\phi}_i$. Let $\mathbf{S}_i = \{m_i, \boldsymbol{\phi}_i, \boldsymbol{\nu}_i\}$ be a parameterization structure and $\boldsymbol{\theta}_i = \{\theta_{ij} \mid j = 1, 2, \cdots, m_i\}$ be the parameter values associated with the structure $\mathbf{S}_i$. If we use the shorthand notation $(\mathbf{S}_i, \boldsymbol{\theta}_i)$ to denote a single parameterization $\hat{\theta}_i(\mathbf{x})$ in (7.1.1), then a parameterization of the whole system model is

$$(\mathbf{S}, \boldsymbol{\theta}) = \{\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_r; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_r\}. \tag{7.1.2}$$

By systematically varying the model structure and associated parameter values, we can generate a set of candidate models. We would like to find a model from the set that can best represent the real system. An EIP is formulated for this purpose.

7.1.2 Extended Inverse Problem

As mentioned in Chap. 1, CIP identifies model parameters under the assumption that the model structure is known, while EIP aims to identify the model structure and model parameters simultaneously. Based on criteria (C-1) and (C-2) of inverse problem formulation (i.e., fitting observation data and using prior information), EIP is defined by (Sun and Sun 2002)

$$\begin{aligned} &(\mathbf{S}^*, \boldsymbol{\theta}^*) = \arg\min_{(\mathbf{S},\boldsymbol{\theta})} S(\mathbf{S}, \boldsymbol{\theta}),\ (\mathbf{S}, \boldsymbol{\theta}) \in \mathfrak{M}_{ad} \subset \mathbb{M}, \\ &S(\mathbf{S}, \boldsymbol{\theta}) = \left\| \mathbf{u}_D(\mathbf{S}, \boldsymbol{\theta}) - \mathbf{u}_D^{obs} \right\| + \alpha R(\mathbf{S}, \boldsymbol{\theta}) \end{aligned} \tag{7.1.3}$$

where $\alpha R(\mathbf{S}, \boldsymbol{\theta})$ is a regularization term. When an initial guess $(\mathbf{S}^0, \boldsymbol{\theta}^0)$ is available, we may set $R(\mathbf{S}, \boldsymbol{\theta}) = \left\| (\mathbf{S}, \boldsymbol{\theta}) - (\mathbf{S}^{\mathbf{0}}, \boldsymbol{\theta}^{\mathbf{0}}) \right\|$.

EIP can be defined similarly in the statistical framework. Rewriting the Bayes' theorem for a system model $\mathcal{M}=(\mathbf{S}, \boldsymbol{\theta})$, we have

$$p_*(\mathbf{S}, \boldsymbol{\theta}) = \mathrm{c}L(\mathbf{S}, \boldsymbol{\theta}) p_{\mathbf{0}}(\mathbf{S}, \boldsymbol{\theta}), \tag{7.1.4}$$

where $p_*(\mathbf{S}, \boldsymbol{\theta})$, $p_0(\mathbf{S}, \boldsymbol{\theta})$, and $L(\mathbf{S}, \boldsymbol{\theta})$ are posterior distribution, prior distribution, and likelihood function of the model, respectively, and c is a normalization constant. The MAP estimate of the model is thus given by

$$(\mathbf{S}^*, \boldsymbol{\theta}^*)_{MAP} = \arg\min_{(\mathbf{S},\boldsymbol{\theta})} \{-\log L(\mathbf{S}, \boldsymbol{\theta}) - \log p_{\mathbf{0}}(\mathbf{S}, \boldsymbol{\theta})\}. \tag{7.1.5}$$

When both observation error distribution and prior distribution are Gaussian (Sect. 4.2.3), the MAP estimate (7.1.5) is identical to (7.1.3).

Two approaches are available for solving the EIP problem (7.1.3), (i) using a global optimization method to search for $\mathbf{S}^*$ and $\boldsymbol{\theta}^*$ simultaneously or (ii) using a min-min optimization method to search for $\mathbf{S}^*$ and $\boldsymbol{\theta}^*$ in a hierarchical manner because (7.1.3) can be replaced by

$$(\mathbf{S}^*, \boldsymbol{\theta}^*) = \arg\min_{\mathbf{S}} \min_{\boldsymbol{\theta}} S(\mathbf{S}, \boldsymbol{\theta}). \tag{7.1.6}$$

When searching for the optimal structure (the outer min problem in (7.1.6)), the optimal parameter values associated with the current structure are identified by solving a CIP (the inner min problem in (7.1.6)). In practice, however, both approaches for EIP solution are very hard to implement when the structures of candidate models are complex. Another major difficulty of solving EIP is its nonunique nature. Because the errors in model structure and parameter values can compensate each other when minimizing the fitting residual, different combinations of model structures and parameter values could fit the same observation data equally well, especially when the model structure is complicated and/or the data are insufficient. In this case, incorporating prior information related to both model structure and parameter values into the regularization term of EIP problem (7.1.3) is helpful.

7.1.2.1 EIP for a Single Distributed Parameter

Let us start from a simple problem, for which the system model to be identified is a single distributed parameter $\theta(\mathbf{x})$. In this case, EIP reduces to identification of the following form:

$$\hat{\theta}(\mathbf{x}) = (\mathbf{S}, \boldsymbol{\theta}) = \sum_{j=1}^{m} \theta_j \phi_j(\mathbf{x}, \boldsymbol{\nu}), \tag{7.1.7}$$

where $\mathbf{S}=\{m, \boldsymbol{\phi}, \boldsymbol{\nu}\}$ is its structure, m is the dimension of parameterization, $\boldsymbol{\phi}$ is a set of basis functions, $\boldsymbol{\nu}$ is a set of shape parameters, and $\boldsymbol{\theta}$ is the parameter values associated with the structure. Eq. (7.1.7) is a special case of (7.1.1) when i in the latter equation is set to 1. In CIP, $\mathbf{S}$ is given and only $\boldsymbol{\theta}$ needs to be identified, whereas in EIP, both $\mathbf{S}$ and $\boldsymbol{\theta}$ (i.e., all four components $\{m, \boldsymbol{\phi}, \boldsymbol{\nu}, \boldsymbol{\theta}\}$) need to be identified.

Example 7.1 *Parameter Structure Identification Using Zonation Parameterization* Let us consider the identification of a distributed parameter $\theta(\mathbf{x})$ defined in a two-dimensional domain, Ω. As we mentioned in Chap. 2, zonation is one of the simplest and most often used parameterization methods. It partitions Ω into zones $\{\Omega_j \mid j = 1, 2, \cdots, m\}$. Figure 7.1a assumes that $\theta(\mathbf{x})$ is constant in each zone (i.e., a

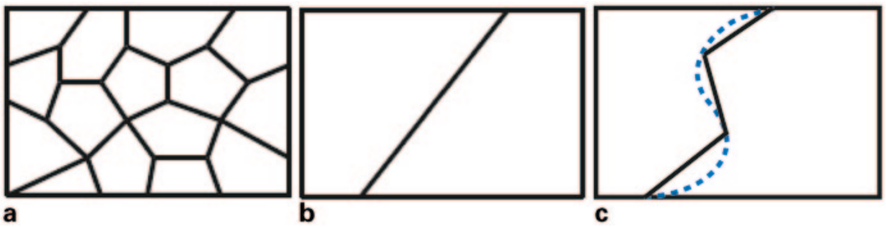


Fig. 7.1 Illustration of **a** a zonation partition; **b** a two-zone partition with a straight line boundary; and **c** a two-zone partition with a curve boundary, which is approximated by a 3-segment zigzag line

piecewise constant function $\hat{\theta}(\mathbf{x})$ is used as a model for identification). In this case, $\{m, \boldsymbol{\phi}, \boldsymbol{\nu}, \boldsymbol{\theta}\}$ have the following meanings:

- The dimension m of parameterization is the number of zones.
- The j-th basis function is defined by

$$\phi_j(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in j\text{-th zone} \\ 0, & \text{otherwise} \end{cases}$$

- The shape parameter $\boldsymbol{\nu}$ determines the zonation pattern. Let us consider a simple two-zone case. In Fig. 7.1b, the boundary between the two zones is a straight line. Because the number of DOF of such a boundary is two (its two ends must be on the boundary of the region), the shape parameter $\boldsymbol{\nu}$ has two components. In Fig. 7.1c, the boundary between the two zones is a curve. If we use a zigzag line (piecewise constant) with three segments to replace the curve approximately, the number of DOF of the line is six, or the shape parameter has six components. In general, the dimension of the shape parameter is several times larger than the number of zones.
- Component θ_j of $\boldsymbol{\theta}$ is the parameter value of the jth zone.

This example clearly shows that even when basis functions are given, the DOF of structure parameters could still be large because of the variability in structure patterns.

7.1.3 Model Complexity Selection

The complexity of a model is characterized by its DOF. Two major difficulties exist in the identification of a complex model. First, the more complex a model is, the more data are required. When data are insufficient, the EIP will be ill-posed. Second, the optimization problems associated with EIPs are combinatorial in nature and the computational effort is generally prohibitive. In fact, a model can be successfully identified by solving an EIP only when its structure is relatively simple or the number of candidate structures is small.

In the traditional process of model construction, a conceptual model is used first to fit existing data. If the fitting residual is large after manually or automatically adjusting model parameters, the modeler would need to modify the structure of the conceptual model and introduce more adjustable parameters (i.e., increasing the DOF) to reduce the fitting residual. During this process, the model complexity is gradually increased and the fitting residual is gradually decreased; the model calibration process is terminated when the fitting residual reaches a satisfactory level. Unfortunately, a model developed by following such a process often gives erroneous prediction results. This problem is caused by model structure error and data insufficiency.

Let the real system be $\mathcal{M}^t = (\mathbf{S}^t, \boldsymbol{\theta}^t)$, the identified model be $\mathcal{M}^* = (\mathbf{S}^*, \boldsymbol{\theta}^*)$, and the observed data be generated by the real system (i.e., $\mathbf{u}_D^{obs}$ are equal to $\mathbf{u}_D(\mathbf{S}^t, \boldsymbol{\theta}^t)$ plus observation error). Because of data insufficiency, $\mathbf{u}_D(\mathbf{S}^*, \boldsymbol{\theta}^*)$ may be very

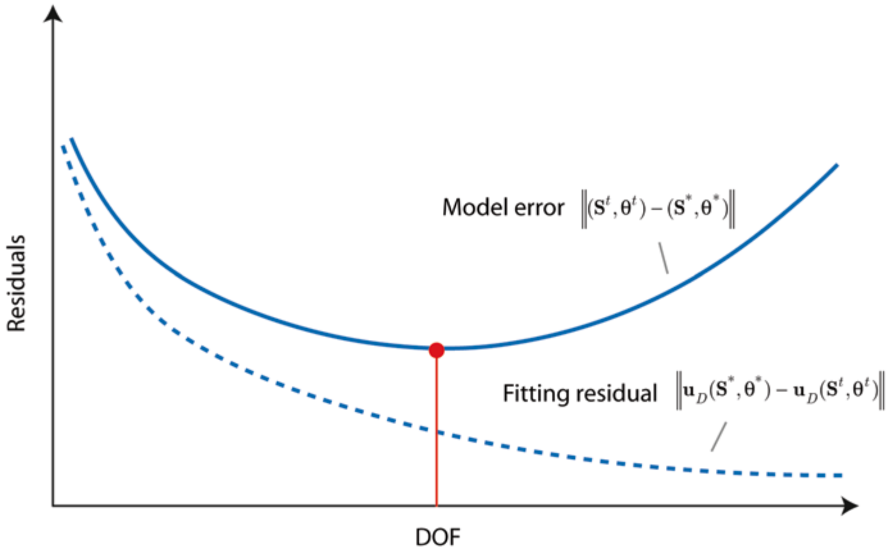


Fig. 7.2 Effect of increasing model complexity. The *solid line* shows the changes of model error, while the *dashed line* shows the fitting residual. The *arrow* indicates the most appropriate model complexity

close to $\mathbf{u}_D(\mathbf{S}^t,\boldsymbol{\theta}^t)$, even when $\mathbf{S}^*$ is significantly different from $\mathbf{S}^t$. As we have explained before, the effect of incorrect model structure is compensated automatically by the effect of incorrect parameter values during the data-fitting process. Paradoxically, making $\mathbf{u}_D(\mathbf{S}^*,\boldsymbol{\theta}^*)$ and $\mathbf{u}_D(\mathbf{S}^t,\boldsymbol{\theta}^t)$ closer in observation space will increase the difference between $\boldsymbol{\theta}^*$ and $\boldsymbol{\theta}^t$ in parameter space. As a result, incorrect parameter values are always obtained when an incorrect model structure is used for inversion (Sun and Yeh 1985).

As a hypothetical example, the true system $(\mathbf{S}^t,\boldsymbol{\theta}^t)$ is assumed to be known. Figure 7.2 shows a typical process of model identification by increasing model complexity. With the increase of DOF, the fitting residual (shown by the dash line) keeps decreasing, but the difference between the identified model and the true system, $\|(\mathbf{S}^t,\boldsymbol{\theta}^t)-(\mathbf{S}^*,\boldsymbol{\theta}^*)\|$, decreases at first and then starts to increase after a certain DOF. The model then becomes "overparameterized," and the data are "overfitted."

Overparameterization is caused by introducing too many unknowns for inversion, more than what can be supported by data; thus, a direct consequence of overparameterization is the increase of model variability. An overparameterized model is unreliable for model prediction, just as an under-parameterized model is. In Fig. 7.2, the most appropriate model complexity corresponds to the minimum of the discrepancy curve. However, this curve is dependent on the true system and is thus unknown in practice.

7.1.3.1 Model Complexity Selection with SVD

In Sect. 2.2.3, we showed that the inverse solution of a linear model $\mathbf{G}\boldsymbol{\theta}=\mathbf{d}$ becomes unstable when the matrix condition number $\kappa=s_1 / s_r$ is large, where s_1 and s_r are the largest and smallest nonzero singular values of the coefficient matrix $\mathbf{G}$. As mentioned in Sect. 2.2.3, the smallest singular value s_k that can be retained in the truncated SVD must satisfy

$$s_k \geq s_1 r_d / r_\theta, \tag{7.1.8}$$

where $r_d = \|\Delta\mathbf{d}\| / \|\mathbf{d}\|$ and $r_\theta = \left\|\Delta\boldsymbol{\theta}^\dagger\right\| / \left\|\boldsymbol{\theta}^\dagger\right\|$ are the relative uncertainty of data and the relative uncertainty of the identified parameter, respectively. From (7.1.8), only the first k components of $\boldsymbol{\theta}^\dagger$ can be identified from the given dataset. For a nonlinear model, we could use SVD to find the largest DOF to avoid overparameterization, in which the coefficient matrix $\mathbf{G}$ is replaced by the Jacobian matrix $\mathbf{J}_D$ after linearization (see Sect. 2.2.4). However, there are two inherent difficulties associated with this approach. First, the TSVD solution does not have a physical meaning, and second, the computational cost would be expensive because $\mathbf{J}_D$ must be updated each time the identified parameters are updated during the EIP solution process.

7.1.3.2 Model Complexity Selection with Statistical Criteria

In statistics, the model selection problem has been studied extensively. Statistical criteria for model selection attempt to balance the goodness of fit and the complexity of model structure so that both over- and underparameterization can be avoided. Assume that we have the following information:

- A set of n observation data,
- A set of K candidate models $\mathfrak{M} = \{\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_K\}$, and
- The number of DOF of model $\mathcal{M}_k$ is p_k ($k = 1, 2, \ldots, K$).

The problem is how to pick an appropriate model from $\mathfrak{M}$ for modeling the dataset. According to the Akaike Information Criterion (AIC) for model selection (Akaike 1974), model $\mathcal{M}_{k^*}$ should be selected from $\mathfrak{M}$ such that its AIC value is the smallest among all,

$$k^* = \arg\min_k \left\{-2\log L(\mathcal{M}_k) + 2p_k\right\}, \tag{7.1.9}$$

where $L(\mathcal{M}_k)$ is the likelihood of model $\mathcal{M}_k$. Problem (7.1.9) can be seen as a MLE problem with a penalty term proportional to the number of DOF (the model complexity) or can be seen as a two-criterion optimization problem that gives a trade-off solution between the goodness of data fit and the complexity of model structure.

Another commonly used model selection criterion is the *Bayesian Information Criterion* (BIC) (Schwarz 1978). According to BIC, k^* is calculated by

$$k^* = \arg\min_k \left\{-2\log L(\mathcal{M}_k) + p_k \log n\right\}. \tag{7.1.10}$$

Comparing (7.1.9) to (7.1.10), we see that AIC penalizes the DOF less strongly than BIC does. AIC and BIC have been widely used for linear regression and time series modeling. In principle, they can be applied to selection of any maximum likelihood-based models. The actual performance of AIC and BIC will depend on the nature of the data-generation model and sample sizes (Burnham and Anderson 2002). AIC and BIC are mainly applied to model structure selection for regression models, which will be covered in details in Chap. 8.

Besides AIC and BIC, other statistical criteria for model selection exist, such as deviance information criterion (DIC) and Kullback information criterion (KIC). Detailed discussion on these criteria and their modifications can be found in (Ye et al. 2008; Burnham and Anderson 2002; Claeskens and Hjort 2008). The performance of different model selection criteria has been studied for groundwater modeling (e.g., Ye et al. 2008; Poeter and Hill 2007; Refsgaard et al. 2012; Rojas et al. 2008) and hydrological modeling (e.g., Wagener et al. 2004; Dawson and Wilby 2001; Schoups et al. 2008).

Practical challenges exist when applying the aforementioned statistical model selection criteria to identifying system models involving distributed parameters. First, there may be an infinite number of models that have the same number of DOF but different structure patterns. For example, the number of possible two-zone structures in the region given in Example 7.1 is infinite and uncountable. Second, the computational effort of finding an appropriate model by calculating (7.1.9) or (7.1.10) is essentially the same as that of obtaining the MAP estimate (7.1.5). When the DOF of a parameterized model is large, the curse of dimensionality principle (see Chap. 6) dictates that the computational effort will become prohibitive.

7.2 Adaptive Parameterization

7.2.1 General Algorithm

In general, an adaptive parameterization algorithm attempts to construct a series of models sequentially

$$\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_m, \mathcal{M}_{m+1}, \cdots \tag{7.2.1}$$

During the adaptive parameterization process, the complexity of these models is gradually increased, the structure pattern is optimized after each increase of complexity, and parameter values associated with the optimal structure pattern are estimated. The process is repeated until no more information can be extracted from the data to support identification of a more complex model structure. With adaptive parameterization, the model complexity is determined automatically by the quantity and quality of available data.

7.2.1.1 Discretized Form

As in the last section, the grid used for solving the forward problem will be considered as the finest structure or the most complex structure. After discretization, the maximum number of DOF of a distributed parameter is equal to the total number of nodes (N) of the grid (or the total number of elements when a block-centered numerical method is used). We will use $\boldsymbol{\theta}_N$ to denote the N-dimensional input parameter vector. After discretization, parameterization (7.1.7) can be represented by

$$\boldsymbol{\theta}_N = \mathbf{G}\boldsymbol{\theta}_G, \tag{7.2.2}$$

where $\mathbf{G}$ is an $N \times m$ matrix called the *structure matrix* of parameterization and $\boldsymbol{\theta}_G$ is the parameter vector associated with the structure. The elements of matrix $\mathbf{G}$ are $g_{ij} = \phi_j(\mathbf{x}_i, \mathbf{v}_G)$ ($i = 1, 2, \cdots, N$; $j = 1, 2, \cdots, m$), where $\mathbf{x}_i$ represents the coordinates of node i. For example, when the zonation method is used, we have $g_{ij} = 1$ when node i is located in zone j; otherwise, $g_{ij} = 0$. In this case, the pattern of nonzero elements in the structure matrix follows exactly the zonation pattern.

Before parameterization, the objective function for identifying $\boldsymbol{\theta}_N$ is given by

$$S(\boldsymbol{\theta}_N) = \left\| \mathbf{u}_D(\boldsymbol{\theta}_N) - \mathbf{u}_D^{obs} \right\| + \alpha R(\boldsymbol{\theta}_N). \tag{7.2.3}$$

Equation (7.2.3) is a CIP problem. However, minimizing $S(\boldsymbol{\theta}_N)$ is generally infeasible because of the high dimensionality of $\boldsymbol{\theta}_N$. After parameterization, (7.2.3) becomes

$$S(\mathbf{G}\boldsymbol{\theta}_G) = \left\| \mathbf{u}_D(\mathbf{G}\boldsymbol{\theta}_G) - \mathbf{u}_D^{obs} \right\| + \alpha R(\mathbf{G}\boldsymbol{\theta}_G). \tag{7.2.4}$$

Equation (7.2.4) is an EIP problem because both structure matrix $\mathbf{G}$ and value vector $\boldsymbol{\theta}_G$ are unknown. Comparing (7.2.4) with (7.1.3), we see that the generic $(\mathbf{S}, \boldsymbol{\theta})$ pair is replaced by $\mathbf{G}\boldsymbol{\theta}_G$. Adaptive parameterization attempts to solve the EIP by the following general steps:

1. Select a set of basis functions $\{\boldsymbol{\phi}(\mathbf{x}, \mathbf{v})\}$.
2. Design an initial parameterization model with a relatively low dimension m and a simple structure pattern (i.e., the dimension of shape parameter $\mathbf{v}$ is low) on the basis of prior information.
3. Because m and $\boldsymbol{\phi}$ are fixed, the structure matrix $\mathbf{G}$ depends only on the shape parameter $\mathbf{v}_G$. From (7.2.4), the EIP (7.1.3) reduces to identification of the optimal $(\mathbf{v}_\mathbf{G}^*, \boldsymbol{\theta}_\mathbf{G}^*)$ such that

$$(\mathbf{v}_G^*, \boldsymbol{\theta}_G^*) = \arg \min_{(\mathbf{v}_G, \boldsymbol{\theta}_G)} S(\mathbf{v}_G, \boldsymbol{\theta}_G)$$

where (7.2.5)

$$S(\mathbf{v}_G, \boldsymbol{\theta}_G) = \left\| \mathbf{u}_D(\mathbf{G}(\mathbf{v}_G)\boldsymbol{\theta}_G) - \mathbf{u}_D^{obs} \right\| + \alpha R(\mathbf{G}(\mathbf{v}_G)\boldsymbol{\theta}_G).$$

4. Check whether one or more predetermined stopping criteria are satisfied. If not, increase the complexity of the model by increasing the dimension of parameterization and/or the dimension of the shape vector, and then return to step 3.

Thus, a sequence of nested optimization problems needs to be solved. Different adaptive parameterization methods use different basis functions, shape parameters, and optimization algorithms. The adaptive parameterization scheme delineated herein also shares common features with adaptive mesh refinement (AMR) commonly used in FEM (Lee et al. 1998; Berger and Oliger 1984).

7.2.1.2 Sensitivity Analysis

Regardless of the approach used for parameter structure identification, it is important to be able to find the sensitivity of objective function with respect to $\mathbf{v}_G$ and $\boldsymbol{\theta}_G$ effectively. Before parameterization, the objective function is $S(\boldsymbol{\theta}_N)$ in (7.2.3). In Chap. 5, we learned that when the adjoint-state method is used, the gradient components $\partial S / \partial \theta_i$ ($i = 1, 2, \cdots, N$) for all nodes of the finest grid can be obtained by only solving the forward problem once and the adjoint problem once. After parameterization, the objective function becomes

$$S(\mathbf{v}_G, \boldsymbol{\theta}_G) = S(v_{G,1}, v_{G,2}, \cdots, v_{G,k}, \theta_{G,1}, \theta_{G,2}, \cdots, \theta_{G,m}), \tag{7.2.6}$$

and its gradient components can be calculated by the chain rule as

$$\frac{\partial S}{\partial \theta_{G,j}} = \sum_{i=1}^{N} \frac{\partial S}{\partial \theta_i} \frac{\partial \theta_i}{\partial \theta_{G,j}}, j = 1, 2, \cdots, m, \tag{7.2.7}$$

$$\frac{\partial S}{\partial v_{G,j}} = \sum_{i=1}^{N} \frac{\partial S}{\partial \theta_i} \frac{\partial \theta_i}{\partial v_{G,j}}, \; j = 1, 2, \cdots, k. \tag{7.2.8}$$

After $\partial S / \partial \theta_i$ are obtained by the adjoint-state method (or by the reverse model of numerical differentiation) for all nodes $i = 1, 2, \cdots, N$, the cost of calculating these gradient components is minimal because $\partial \theta_i / \partial \theta_{G,j}$ and $\partial \theta_i / \partial v_{G,j}$ can be obtained directly from the structure matrix $\mathbf{G}$.

After parameterization, the sensitivity coefficients of model output $u_{D,l}$ with respect to the components of $\mathbf{v}_G$ and $\boldsymbol{\theta}_G$ can also be obtained via the chain rule as

$$\frac{\partial u_{D,l}}{\partial \theta_{G,j}} = \sum_{i=1}^{N} \frac{\partial u_{D,l}}{\partial \theta_i} \frac{\partial \theta_i}{\partial \theta_{G,j}}, \; l = 1, 2, \cdots, n; j = 1, 2, \cdots, m \tag{7.2.9}$$

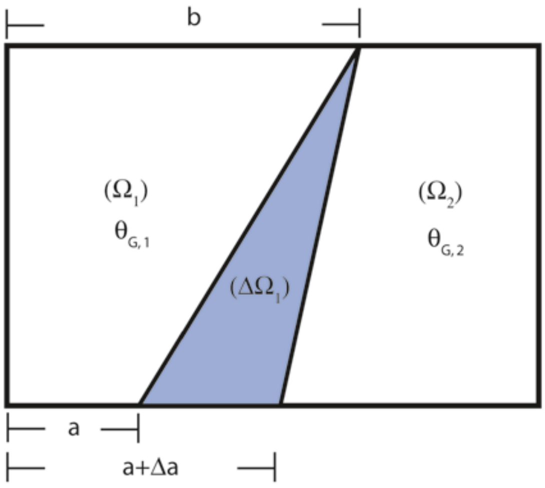


Fig. 7.3 Two zone structure with a moving cutting line

$$\frac{\partial u_{D,l}}{\partial v_{G,j}} = \sum_{i=1}^{N} \frac{\partial u_{D,l}}{\partial \theta_i} \frac{\partial \theta_i}{\partial v_{G,j}}, \quad l = 1,2,\cdots,n; j = 1,2,\cdots,k \tag{7.2.10}$$

where n is the number of observation data. All sensitivity coefficients $\partial u_{D,l} / \partial \theta_i$ ($i = 1,2,\cdots,N$) for the finest grid can be obtained effectively by the adjoint-state method. As in (7.2.7) and (7.2.8), $\partial \theta_i / \partial \theta_{G,j}$ and $\partial \theta_i / \partial v_{G,j}$ can be obtained from the structure matrix. Therefore, the sensitivity analysis problem for solving EIP (7.2.5) is not difficult.

Example 7.2 *Calculating the Gradient of a Zonation Structure*
A line in Fig. 7.3 cuts the region into two zones, Ω_1 and Ω_2. If the two ends of the cutting line are located at the two horizontal sides, then the shape parameters of the structure are $v_{G,1} = a$ and $v_{G,2} = b$, respectively (i.e., the distances of the two ends measured from the left side of the region). The parameter values associated with the two zones are $\theta_{G,1}$ and $\theta_{G,2}$, respectively. When node $i \in \Omega_1$, let $\theta_i = \theta_{G,1}$, and when node $i \in \Omega_2$, let $\theta_i = \theta_{G,2}$. Therefore, $\partial \theta_i / \partial \theta_{G,j} = 1$ when node $i \in \Omega_j$, and $\partial \theta_i / \partial \theta_{G,j} = 0$ when node $i \notin \Omega_j$, where $j = 1, 2$. From (7.2.7),

$$\frac{\partial S}{\partial \theta_{G,1}} = \sum_{i \in \Omega_1} \frac{\partial S}{\partial \theta_i} \text{ and } \frac{\partial S}{\partial \theta_{G,2}} = \sum_{i \in \Omega_2} \frac{\partial S}{\partial \theta_i}. \tag{7.2.11}$$

When $v_{G,1}$ has an increment $\Delta v_{G,1}$, a subregion $\Delta\Omega_1$ is generated because of this change (the shaded area in Fig. 7.3), in which the node value θ_i changes from $\theta_{G,1}$ to $\theta_{G,2}$, or $\Delta\theta_i = \theta_{G,2} - \theta_{G,1}$. Therefore, $\partial \theta_i / \partial v_{G,1} \approx \Delta \theta_i / \Delta v_{G,1}$. From (7.2.8),

$$\frac{\partial S}{\partial v_{G,1}} = \sum_{i \in \Delta\Omega_1} \frac{\partial S}{\partial \theta_i} \frac{\Delta \theta_i}{\Delta v_{G,1}} \text{ and } \frac{\partial S}{\partial v_{G,2}} = \sum_{i \in \Delta\Omega_2} \frac{\partial S}{\partial \theta_i} \frac{\Delta \theta_i}{\Delta v_{G,2}}, \tag{7.2.12}$$

where $i \in \Delta\Omega_1$ in the first equation means the summation is over all node i that are located in $\Delta\Omega_1$. The second equation is obtained similarly.

7.2.1.3 Solution of EIP

Local Optimization Once the gradient of objective function $S(\mathbf{v}_G, \boldsymbol{\theta}_G)$ is calculated by (7.2.7) and (7.2.8), a gradient-based local optimization method can be used to solve the EIP in (7.2.5). Unfortunately, the objective function formulated in this way is usually non-convex because different pairs of $(\mathbf{v}_G, \boldsymbol{\theta}_G)$ (i.e., different structures combined with different parameter values) may produce very close model outputs. As a result, the solution of a gradient-based optimization method is often trapped at local minima.

Global Optimization A global optimization method can be used to find the global minimum of objective function (7.2.6). For example, Zheng and Wang (1996) used tabu search and simulated annealing (SA), Tsai et al. (2003) used generic algorithm (GA), and Tan et al. (2008) combine tabu search with the adjoint-state method to increase the efficiency of the search process. However, as the dimension of parameterization increases, the computational effort of global optimization quickly becomes infeasible because convergence of a global optimization algorithm is usually very slow and because during each iteration a high-dimensional CIP must be solved.

Min-Min Optimization The EIP in (7.2.5) can be solved by a min-min optimization process

$$(\mathbf{v}_G^*, \boldsymbol{\theta}_G^*) = \arg\min_{\mathbf{v}_G} \min_{\boldsymbol{\theta}_G} S(\mathbf{v}_G, \boldsymbol{\theta}_G), \mathbf{v}_G \in V_G, \boldsymbol{\theta}_G \in P_G \tag{7.2.13}$$

where V_G and P_G are admissible regions of $\mathbf{v}_G$ and $\boldsymbol{\theta}_G$, respectively. Furthermore, the min-min problem can be solved iteratively by

$$\min_{\mathbf{v}_G} F(\mathbf{v}_G), \text{ for } \mathbf{v}_G \in V_G, \text{ where} \tag{7.2.14}$$

$$F(\mathbf{v}_G) = \min_{\boldsymbol{\theta}_G} S(\mathbf{v}_G, \boldsymbol{\theta}_G), \text{ for } \boldsymbol{\theta}_G \in P_G \tag{7.2.15}$$

The inner optimization problem (7.2.15) is a CIP because $\mathbf{v}_G$ is fixed and the outer optimization problem (7.2.14) contains only shape parameters. The high-dimensional EIP is thus decomposed into two relatively low-dimensional optimization problems. The efficiency of the min-min algorithm can be improved significantly by the following structure selection method.

7.2.1.4 Structure Selection

In one iteration cycle of min-min optimization, the outer min generates structure candidates and the inner min solves the CIP to obtain the fitting residual for each candidate structure. A structure associated with the minimum fitting residual is then selected as the best structure to be used for the next iteration. Most computation effort during this process is spent on solving the costly inner CIP for all candidate structures. Keep in mind that the main goal of solving these CIPs is to facilitate structure selection rather than parameter identification. Based on the adjoint-state method and linearization, Sun and Yeh (1985) presented an effective method for structure selection. When the GLS norm is used (see Sect. 4.2), the objective function is given by

$$S(\boldsymbol{\theta}_N) = [\mathbf{u}_D(\mathbf{x},\boldsymbol{\theta}_N) - \mathbf{u}_D^{obs}]^T \mathbf{W}[\mathbf{u}_D(\mathbf{x},\boldsymbol{\theta}_N) - \mathbf{u}_D^{obs}],\ \boldsymbol{\theta}_N \in P_{ad} \tag{7.2.16}$$

where $\mathbf{W}$ is a weight matrix. Around an estimated $\boldsymbol{\theta}_N^0$ of $\boldsymbol{\theta}_N$, the first-order approximation of the model is $\mathbf{u}_D(\mathbf{x},\boldsymbol{\theta}_N) \approx \mathbf{u}_D(\mathbf{x},\boldsymbol{\theta}_N^0) + \mathbf{J}_D(\boldsymbol{\theta}_N - \boldsymbol{\theta}_N^0)$, where the Jacobian $\mathbf{J}_D$ is evaluated at $\boldsymbol{\theta}_N^0$ and is usually obtained by the adjoint-state method. Using parameterization $\boldsymbol{\theta}_N = \mathbf{G}\boldsymbol{\theta}_G$ and first-order approximation, the objective function (7.2.16) becomes

$$S(\boldsymbol{\theta}_N) \approx (\mathbf{J}_D\mathbf{G}\boldsymbol{\theta}_G - \mathbf{a}_0)^T \mathbf{W}(\mathbf{J}_D\mathbf{G}\boldsymbol{\theta}_G - \mathbf{a}_0), \tag{7.2.17}$$

where $\mathbf{a}_0 = \mathbf{u}_D^{obs} - \mathbf{u}_D(\mathbf{x},\boldsymbol{\theta}_N^0) + \mathbf{J}_D\boldsymbol{\theta}_N^0$. Minimization of (7.2.17) gives

$$(\mathbf{G}^T\mathbf{J}_D^T\mathbf{W}\mathbf{J}_D\mathbf{G})\boldsymbol{\theta}_G = \mathbf{G}^T\mathbf{J}_D^T\mathbf{W}\mathbf{a}_0. \tag{7.2.18}$$

Let the solution to the linear system in (7.2.18) be $\hat{\boldsymbol{\theta}}_G$, the minimum of the objective function can be written as

$$S(\hat{\boldsymbol{\theta}}_N) \approx S(\mathbf{G}\hat{\boldsymbol{\theta}}_G) = (\mathbf{J}_D\mathbf{G}\hat{\boldsymbol{\theta}}_G - \mathbf{a}_0)^T \mathbf{W}(\mathbf{J}_D\mathbf{G}\hat{\boldsymbol{\theta}}_G - \mathbf{a}_0). \tag{7.2.19}$$

When we have a set of structures, we can find the solutions of (7.2.18) for each of these structures by only changing $\mathbf{G}$ in the equation because $\mathbf{J}_D$ and $\mathbf{a}_0$ are independent of structures. Next, we can use (7.2.19) to estimate the fitting residual associated with each of these structures without solving CIP. A structure that gives the smallest fitting residual is then selected as the best structure among these structures. Without solving CIP, the cost of completing such a structure selection process for the min-min optimization problem in (7.2.14) and (7.2.15) is not expensive.

7.2.2 Adding Basis Points

In this method, a set of basis points is used as the shape parameter of parameterization. New basis points are added gradually to increase the structure complexity, and their locations are identified for optimizing each structure. The original form of the method was presented in Sun and Yeh (1985) for groundwater modeling. In principle, any interpolation method that involves basis points, such as IDW, Voronoi diagram, kriging, and others introduced in Chap. 6, can be used here for adaptive parameterization.

By using different basis functions in (7.1.7), the identified representative parameter can be either discrete or continuous. One example is Voronoi zonation. When $\mathbf{v}_G = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ is used as generators, all Voronoi cells $\{V(\mathbf{x}_j) \mid j=1,2,\cdots,m\}$ form a partition of the definition region (Sect. 6.1.4). If elements of the structure matrix $\mathbf{G}(\mathbf{v}_G)$ in (7.2.2) are defined by $g_{ij} = 1$ when node $i \in V(\mathbf{x}_j)$ and $g_{ij} = 0$ otherwise, the parameterization gives a zonation structure with Voronoi cells as its zones and a constant parameter value is associated with each zone. Another simple example is the use of IDW that can produce continuous and smooth interpolation functions by adding additional shape parameters (Sect. 6.1.4).

Each time new basis points are added, their locations need to be optimized by solving a global minimization problem. During this process, we have to find and compare the fitting residuals for all candidate structure patterns. If the fitting residual of each candidate is obtained by solving a CIP, the total computational effort for pattern optimization would become unaffordable. Fortunately, we can use the adjoint-state method described in the last subsection to expedite structure selection. To evaluate the fitting residual of a structure pattern, we need only to recalculate the structure matrix $\mathbf{G}$ in (7.2.19), rather than resolve CIP.

The major steps of solving EIP by gradually adding basis points include:

1. Form an initial model structure Z_0. If there is no prior information, let Z_0 be a homogeneous structure with only one basis point. Then, solve CIP with the initial structure to find the inverse solution $\boldsymbol{\theta}_0^*$, its span $\boldsymbol{\theta}_{N,0}^* = \mathbf{G}(\mathbf{v}_G^*)\boldsymbol{\theta}_0^*$, and the fitting residual $S(\boldsymbol{\theta}_{N,0}^*)$.
2. Calculate Jacobian matrix $\mathbf{J}_D(\boldsymbol{\theta}_{N,0}^*)$ by the adjoint-state method.
3. Add one or several new basis points to increase the structure complexity. The starting positions of new basis points should include the previous basis points as a subset (i.e., all basis points from the previous iteration should be covered) so that the fitting residual can keep decreasing rather than increasing, after adding new basis points.
4. Find the optimal zonation pattern. Follow a global search strategy to move the basis points from their starting locations to new locations. There are two options: only optimizing the locations of newly added basis points or optimizing the locations of all basis points. The adjoint-based pattern search process is used here to search for the optimal zonation pattern until the fitting residual cannot be decreased further for any new movement. During this process, when the linear system (7.2.18) becomes ill-conditioned for a movement, skip it and move to the next one. After the optimal pattern $\mathbf{v}_G^*$ is found, a new model structure Z_1 is obtained. Then, solve CIP to find the optimal parameter vector $\boldsymbol{\theta}_1^*$, its span $\boldsymbol{\theta}_{N,1}^* = \mathbf{G}(\mathbf{v}_G^*)\boldsymbol{\theta}_1^*$, and the fitting residual $S(\boldsymbol{\theta}_{N,1}^*)$.
5. Choose one of the following stopping criteria during the solution process:

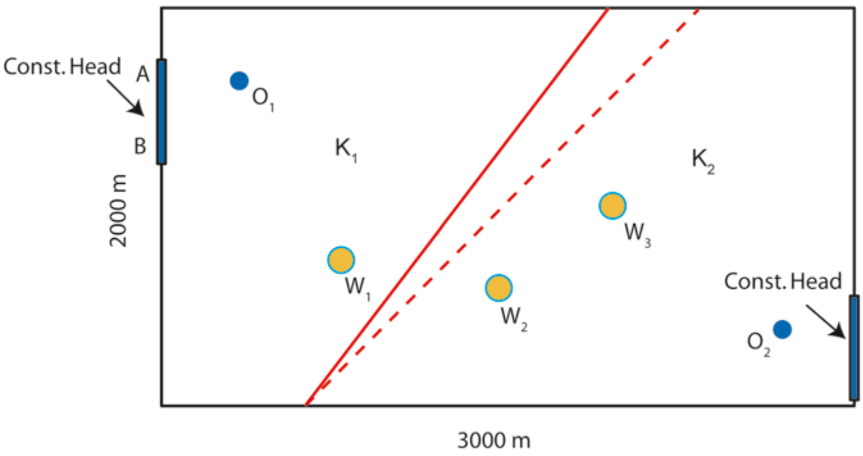


Fig. 7.4 Fault location identification by solving EIP. The *solid line* is the true location of the fault. In the absence of observation error, the true fault location is accurately recovered; in the presence of observation error, the identified fault location (*dashed line*) deviates slightly from its true location

(i) $\left\|\boldsymbol{\theta}^*_{N,1} - \boldsymbol{\theta}^*_{N,0}\right\|$ is less than a predetermined tolerance;

(ii) $S(\boldsymbol{\theta}^*_{N,1})$ is less than a predetermined tolerance;

(iii) $\left|S(\boldsymbol{\theta}^*_{N,1}) - S(\boldsymbol{\theta}^*_{N,0})\right|$ is less than a predetermined tolerance.

(iv) After adding new basis points, the linear system (7.2.18) becomes ill-conditioned for all movements.

If any of the above criteria is satisfied, the process can be stopped. If none of these criteria is met, replace Z_0 by Z_1, $\boldsymbol{\theta}^*_{N,0}$ by $\boldsymbol{\theta}^*_{N,1}$, $S(\boldsymbol{\theta}^*_{N,0})$ by $S(\theta^*_{N,1})$ and then go back to Step 2. Criterion (ii) may also be seen when data are limited (see Example 7.4 below). Criterion (iv) means the data cannot support identification of a more complex structure, and therefore, the search process should be terminated.

Example 7.3 *Optimal Partition of One Zone into Two Zones*
In this example, the hydraulic conductivity of a hypothetical aquifer described in Example 5.14 is identified by solving EIP. We now assume that a fault separates the aquifer into two homogeneous zones with true hydraulic conductivity $K_1 = 20$ m/day and $K_2 = 60$ m/day, respectively (see Fig. 7.4). A pumping test is conducted to collect data for solving the inverse problem. Wells W_1, W_2, and W_3 are pumped at constant rates of 500, 2000, and 1000 m^3/day, respectively, for 2 days. Head measurements are recorded at five wells O_1, W_1, W_2, W_3, and O_2 at times 0.01, 0.1, 0.5, 1.0, and 2.0 day. According to the basis-point EIP algorithm described in the above, we initially started from a homogeneous structure with a single Voronoi generator $\mathbf{x}_1$ located at the center of the region. Without observation error, the identified hydraulic conductivity is $K^* = 20$ m/day, and RMSE = 0.26 m. After adding the second Voronoi generator $\mathbf{x}_2$ and moving it to the optimal location, the RMSE

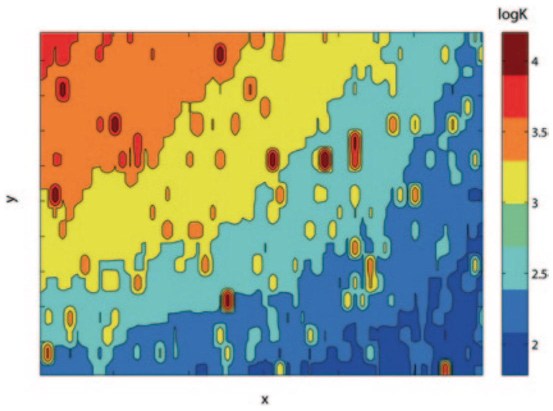


Fig. 7.5 Synthetic true hydraulic conductivity field used in Example 7.4

reduces to 0.05 m, and the true location of the fault and the true values of K in each zone are accurately recovered: $K_1^* = 19.96$ and $K_2^* = 60.16\text{m/day}$.

To test the stability of the inverse solution, random observation errors with norm equal to 0.05 m are added to the data. For the single basis point case, the identified K^* is 36.5 m/day and the RMSE is 0.29 m. After adding the second basis point, the identified fault location is somehow deviated from the true one (dashed line in Fig. 7.4). The identified parameter values associated with the structure are $K_1^* = 18.9$ and $K_2^* = 57.2\text{m/day}$ with RMSE = 0.014 m.

When the true parameter structure is complex and unknown, the EIP solution can provide a simplified representative model, as shown in the following example.

Example 7.4 *Find the Representative Parameter of a Random Field*
We revisit the hypothetical aquifer used in the previous example, but assume that the unknown hydraulic conductivity is spatially heterogeneous with a trend. The random component of hydraulic conductivity field has an exponential covariance structure with variance $\sigma^2 = 2.0$ and correlation distance $\ell = 500$ m; the trend part is a bilinear function determined by the parameter values at the four corners of the region. The resulting "true" hydraulic conductivity field (log-transformed) is shown in Fig. 7.5, which shows wide variations. The observation data were obtained by simulating the same pumping test described in the last example and adding random observation errors with norm equal to 0.05 m.

Using the same basis-point EIP algorithm as before, we start from the homogeneous structure and then add Voronoi generators one at a time until a stopping criterion is satisfied. When the number of zones is increased to four, the fitting residual (RMSE) is reduced to 0.04 m, which is already within the range of observation error. Thus, we terminate the EIP process according to Criterion (iii) mentioned in Step 5 of the basis-point EIP algorithm. The results are summarized in Table 7.1, and the identified four-zone structure is plotted in Fig. 7.6a. Because the EIP solution is nonunique in the presence of observation error, we can find many acceptable models that have similar fitting residuals. For the current case, an equally plausible model is shown in Fig. 7.6b. Of course, we may try to add more zones and search

Table 7.1 Summary of the EIP results of Example 7.4.

Number of zones	1	2	3	4
K values (m/day)	$K_1 = 17.8$	$K_1 = 14.2$ $K_2 = 30.6$	$K_1 = 14.2$ $K_2 = 32.2$ $K_3 = 17.3$	$K_1 = 14.1$ $K_2 = 46.6$ $K_3 = 18.4$ $K_4 = 5.7$
RMSE (m)	0.40	0.12	0.07	0.04

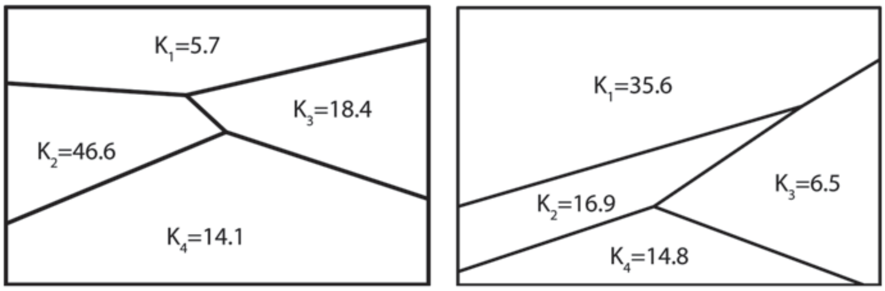


Fig. 7.6 **a** and **b**: Two sets of possible representative parameters identified by EIP solution for approximating the heterogeneous random field in Fig. 7.5

for more patterns to further decrease the fitting residual. However, simply reducing fitting residuals will not lead to better models and may run the risk of overfitting, as we have explained before (see Fig. 7.2 and discussion therein). ■

After Example 7.4, we can make the following observations on EIP solution:

- It is impossible in practice to accurately identify a system, such as the one shown in Fig. 7.5, because of the limitation of data. In this case, the best an inversion algorithm can do is to help find a *data-acceptable* model—a model that cannot be distinguished from the true system in the data space. In other words, we obtain a model that cannot be rejected by the existing observation data and prior information. A model found by EIP is a "data-acceptable" model, and its structure may be significantly different from that of the real system, as we have seen in Example 7.4. In a certain sense, EIP can be seen as a tool for global upscaling or model reduction.
- Accurate parameter values cannot be found by solving EIP, unless the true system structure is known, as we demonstrated in Example 7.3. Structure error is always automatically compensated by the error in parameter values in the EIP solution process.
- In the presence of observation errors, the EIP solution is always nonunique. For example, all models shown in Fig. 7.6 are EIP solutions. The number of "data-acceptable" models is actually infinite. They may have different structure complexities (e.g., number of zones) and different structure patterns (e.g., shape of Voronoi diagrams) and parameter values.

- Not all "data-acceptable" models are useful because they may give different results for a model application. Using a sampling-based approach to estimate reliability of an EIP solution is generally ineffective because of the variability in model structure. We will consider how to find a model that is both "data acceptable" and "application acceptable" in Chap. 12.

7.2.3 Zone Refinement

Zone refinement is another method for adaptive parameterization (Chavent and Bissell 1998; Ameur et al. 2002; Hayek et al. 2008). Starting from an initial zonation model, the method selects one or several zones to refine in order to increase the model complexity. The parameter values associated with the new zonation are then identified by inversion. This process is repeated until a stopping criterion is met. When deciding which zones are to be selected for refinement, a common criterion is that the refinement should lead to the largest reduction in the value of objective function.

7.2.3.1 Refinement Indicator

Let us assume that the current zonation consists of m mutually exclusive zones $\Omega = \Omega_1 \cap \Omega_2 \cap \ldots \cap \Omega_m$ and $\{\hat{\theta}^j \mid j = 1, 2, \cdots, m\}$ are the parameter values associated with this zonation, as identified by solving the CIP. Furthermore, assume that a current zone Ω_j is selected to be cut into two subzones $\Omega_{j,1}$ and $\Omega_{j,2}$. The interface between them is called a *cut* and denoted by *C*. The number of possible cuts, of course, is infinite. Our purpose is to find the optimal cut such that the objective function can be maximally decreased after the cut. Let us denote the unknown parameter values associated with two subzones as θ_1^j and θ_2^j, respectively. Let $\Delta S = S(\hat{\theta}^j, \hat{\theta}^j) - S(\theta_1^j, \theta_2^j)$ be the decrease of the objective function after θ_1^j and θ_2^j are identified by inversion. Using the first-order approximation to the objective function, we have

$$\Delta S \approx (\theta_1^j - \hat{\theta}^j)\sum_{\Omega_{j,1}} \frac{\partial S}{\partial \theta_i} + (\theta_2^j - \hat{\theta}^j)\sum_{\Omega_{j,2}} \frac{\partial S}{\partial \theta_i}, \tag{7.2.20}$$

where the summation $\Sigma_{\Omega_{j,1}}$ is over all nodes (i) within subzone $\Omega_{j,1}$, $\Sigma_{\Omega_{j,2}}$ has similar meaning, and all partial derivatives in the equation are evaluated at $\hat{\theta}^j$ (i.e., zone value before the cut). Because $\hat{\theta}^j$ is the inverse solution obtained by a gradient algorithm, we must have

$$\nabla S(\hat{\theta}^j) = \sum_{\Omega_{j,1}} \frac{\partial S}{\partial \theta_i} + \sum_{\Omega_{j,2}} \frac{\partial S}{\partial \theta_i} \approx 0 \text{ or } \sum_{\Omega_{j,1}} \frac{\partial S}{\partial \theta_i} \approx -\sum_{\Omega_{j,2}} \frac{\partial S}{\partial \theta_i}. \tag{7.2.21}$$

Substituting (7.2.21) into (7.2.20) gives

$$\Delta S \approx \lambda_j(\theta_2^j - \theta_1^j), \tag{7.2.22}$$

where

$$\lambda_j = \frac{1}{2}\left(\sum_{\Omega_{j,1}} \frac{\partial S}{\partial \theta_i} - \sum_{\Omega_{j,2}} \frac{\partial S}{\partial \theta_i}\right), \tag{7.2.23}$$

is called a *refinement indicator* of Ω_j when it is cut into two zones. We can always assume $\lambda_j > 0$; otherwise, we can change $(\theta_2^j - \theta_1^j)$ to $(\theta_1^j - \theta_2^j)$ on the right-hand side of (7.2.22). Using (7.2.21), λ_j can be defined equivalently by

$$\lambda_j = \left|\sum_{\Omega_{j,1}} \frac{\partial S}{\partial \theta_i}\right| \quad \text{or} \quad \lambda_j = \left|\sum_{\Omega_{j,2}} \frac{\partial S}{\partial \theta_i}\right|. \tag{7.2.24}$$

Because λ_j depends on the shape of the two subzones, it is a function of cut, namely, $\lambda_j = \lambda_j(C)$. According to (7.2.22), a larger λ_j will cause a larger decrease of the objective function. Therefore, a cut that corresponds to the maximum λ_j should be considered as the optimal one.

7.2.3.2 Finding the Optimal Cut

After the sensitivity coefficients $(\partial S / \partial \theta_i)$ are calculated for all nodes by the adjoint-state method, the refinement indicator $\lambda_j(C)$ associated with a cut can be obtained simply by summing the sensitivity coefficients of all nodes within subzone $\Omega_{j,1}$ or subzone $\Omega_{j,2}$ of the cut (see Example 7.2). The optimal cut C^* can then be obtained by solving the following maximization problem:

$$C^* = \arg\max_{C} \lambda_j(C), \text{ where } \lambda_j(C) = \left|\sum_{\Omega_{j,1}} \frac{\partial S}{\partial \theta_i}\right|. \tag{7.2.25}$$

To make this problem solvable, candidate cuts have to be parameterized, for example, by considering only horizontal and vertical grid lines as cuts (Figs. 7.7a and b). In this case, of course, only a nearly optimal cut can be found. Moreover, because (7.2.22) is based on the first-order approximation, there is no guarantee that the optimal cut obtained in such a way will lead to the maximum ΔS. As suggested in Ameur et al. (2002), it is better to keep several nearly optimal cuts and calculate ΔS for each of them to determine which one is the best.

After the maximum refinement indicators $\lambda_j(C^*)$ $(j = 1, 2, \cdots, m)$ are calculated for all current zones, we can select one or several zones that have the largest indicator to refine and finally generate a new zonation structure for EIP solution, as shown in Fig. 7.7c.

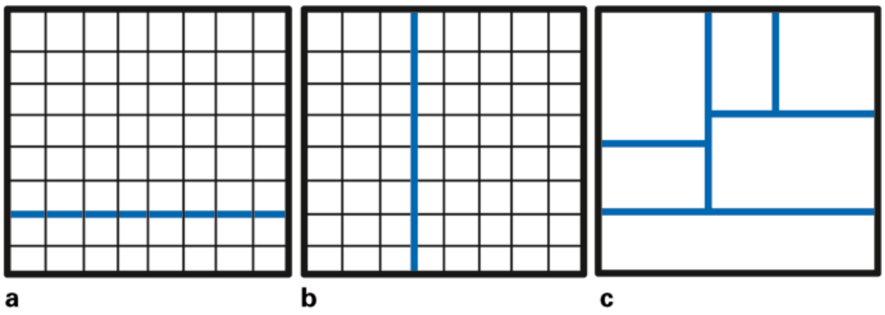


Fig. 7.7 Zone refinement. **a** A horizontal cut, **b** a vertical cut, and **c** an identified zonation

7.2.3.3 Algorithm

The zone refinement method consists of the following major steps:

1. Form an initial zonation structure Z_0 based on prior information. If there is no prior structure information, set Z_0 as a homogeneous (one zone) structure. Then, solve CIP to obtain the optimal parameter values $\boldsymbol{\theta}_0^*$ and its span $\boldsymbol{\theta}_{N,0}^*$ associated with Z_0.
2. Use the adjoint-state method to calculate the sensitivity coefficients evaluated at $\boldsymbol{\theta}_{N,0}^*$ for all nodes.
3. Find the best cut (or several nearly best cuts) and the maximum refinement indicator for each zone, select zones to refine, and finally form a new zonation structure Z_1.
4. Solve the CIP for structure Z_1 to obtain the optimal parameter $\boldsymbol{\theta}_1^*$ and its span $\boldsymbol{\theta}_{N,1}^*$ associated with Z_1.
5. Check the stopping criteria: (i) $\left\|\boldsymbol{\theta}_{N,1}^* - \boldsymbol{\theta}_{N,0}^*\right\|$ becomes small; (ii) $S(\boldsymbol{\theta}_{N,1}^*)$ becomes small; (iii) $\left|S(\boldsymbol{\theta}_{N,1}^*) - S(\boldsymbol{\theta}_{N,0}^*)\right|$ becomes small; and (iv) after adding new cuts, the inverse solution becomes overparameterized. If all of these criteria are not met, replace Z_0 by Z_1, $\boldsymbol{\theta}_{N,0}^*$ by $\boldsymbol{\theta}_{N,1}^*$, $S(\boldsymbol{\theta}_{N,0}^*)$ by $S(\boldsymbol{\theta}_{N,1}^*)$ and then go back to step 2.

Detailed discussions and numerical examples can be found in Ameur et al. (2002) and Hayek et al. (2008). Ameur et al. (2002) also defined refinement indicators for cutting one zone into more than two zones and coarsening indicators for reducing the number of zones. Ameur et al. (2008) extended the method for the identification of multiple distributed parameters of a system. Hayek et al. (2009) presented a second-order refinement indicator that is more efficient than the first-order one.

Both the basis point method and the zone refinement method can be used for adaptive parameterization. To increase the structure complexity, the former increases the number of basis points, while the latter increases the number of zones. Both methods use the adjoint-state sensitivity analysis for pattern optimization. But the former uses global optimization for pattern identification and it can use different basis functions to fit continuously and/or discontinuously varying parameters.

There are other methods for zone structure identification. For example, Eppstein and Dougherty (1996) used an extended Kalman filter (see Chap. 9) and partitional clustering algorithm, Ayvaz (2007) applied a fuzzy c-means clustering and meta-heuristic harmony search algorithm, and Lin et al. (2010) used a modified tabu search method.

7.2.4 The Level Set Method

The level set method was originally introduced by Osher and Sethian in the 1980s for tracking moving interfaces (Osher and Fedkiw 2003; Sethian 1996). It has been successfully applied to a number of disciplines including computational fluid dynamics, computer animations, combustion, and inverse problems for shape reconstruction and scattering (Santosa 1996; Burger and Osher 2005; Dorn and Lesselier 2009; Chan and Tai 2004). It is especially suitable for identifying interfaces between different materials or fluid phases. In this section, the level set method is used as a parameterization tool for structure pattern identification in the framework of EIP.

7.2.4.1 Level Set Parameterization

Assume that the unknown parameter $\theta(\mathbf{x})$ in a region Ω can be approximated by a multivalue function $\hat{\theta}(\mathbf{x})$ that has m different values at m mutually exclusive subregions or zones, viz.

$$\hat{\theta}(\mathbf{x}) = \left\{ \theta_j \middle| \mathbf{x} \in \Omega_j, j = 1, 2, \cdots, m;\ \Omega = \bigcap_{j=1}^{m} \Omega_j \right\}. \tag{7.2.26}$$

This representation is the same as the zonation method, but here a zone may have an arbitrary shape and can be either connected or disconnected, as shown in Fig. 7.8 for a two-zone case. Note that the interface (or hypersurface) Γ is a curve in two-dimensional and surface in three-dimensional space (Fig. 7.9).

As another example, consider an aquifer consisting of different geological materials (facies) with different hydraulic conductivity values. The space occupied by each material is defined as one zone (connected or disconnected), and the inverse problem is to identify the material interfaces and the hydraulic conductivity of each material. In the level set method, this kind of structure is represented and then identified by defining a set of *level set functions*.

Let us first consider the two-zone case:

$$\hat{\theta}(\mathbf{x}) = \begin{cases} \theta_1, & \mathbf{x} \in \Omega_1 \\ \theta_2, & \mathbf{x} \in \Omega_2 \end{cases}.$$

For this case, the EIP requires identifying $(\Gamma, \theta_1, \theta_2)$ based on state observations and prior information, where Γ is the interface between the two zones. We have seen in the last section that using an adaptive zonation method to identify the ge-

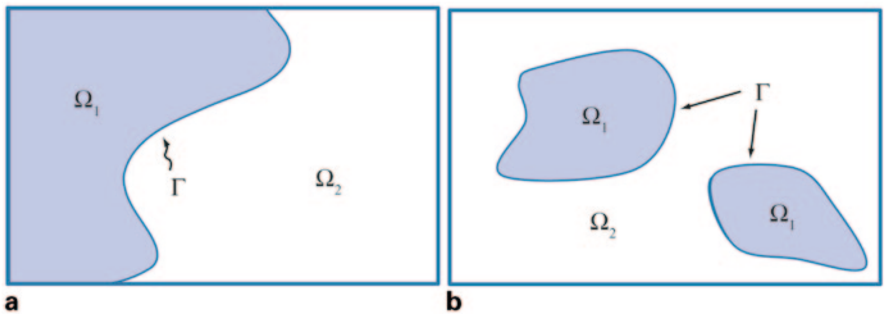


Fig. 7.8 A two-zone structure and its zonal interfaces (Γ): **a** connected and **b** disconnected zones

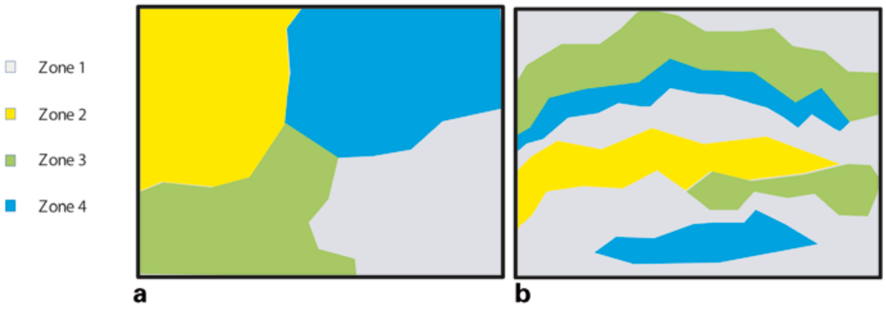


Fig. 7.9 A four-zone structure and its zonal interfaces: **a** connected and **b** disconnected zones

ometry of an irregular zone is not an easy task. In the level set method, a level set function $\varphi(\mathbf{x})$ is defined over the whole region (Ω) by

$$\begin{cases} \varphi(\mathbf{x}) > 0, \text{when } \mathbf{x} \in \Omega_1 \\ \varphi(\mathbf{x}) = 0, \text{when } \mathbf{x} \in \Gamma \\ \varphi(\mathbf{x}) < 0, \text{when } \mathbf{x} \in \Omega_2 \end{cases}. \tag{7.2.27}$$

The two-value function $\hat{\theta}(\mathbf{x})$ can then be represented by

$$\hat{\theta}(\mathbf{x}) = \theta_1 H(\varphi) + \theta_2 (1 - H(\varphi)), \tag{7.2.28}$$

where $H(\varphi)$ is the *Heaviside function* defined by $H(\varphi) = 1$, when $\varphi > 0$, and $H(\varphi) = 0$, when $\varphi \le 0$. Note that the level set function defined by (7.2.27) is nonunique, but the boundary Γ is determined uniquely once a level set function is found. Using the Heaviside function makes (7.2.28) more definite than using the level set function directly. The problem of zone structure identification now becomes identification of the level set function $\varphi(\mathbf{x})$.

A multivalue function $\hat{\theta}(\mathbf{x})$ defined in (7.2.26) can be represented similarly by multiple level set functions $\boldsymbol{\varphi}_k = \{\varphi_i(\mathbf{x}) \mid i = 1, 2, \cdots, k\}$, where k is the smallest integer satisfying $2^k \ge m$ (k level set functions can differentiate at most 2^k zones).

According to the multiple level set framework (Tai and Chan 2004; Vese and Chan 2002), we have

$$\hat{\theta}(\mathbf{x}, \boldsymbol{\theta}_m, \boldsymbol{\varphi}_k) = \sum_{j=1}^{m} \theta_j \phi_j(\mathbf{x}, \boldsymbol{\varphi}_k). \tag{7.2.29}$$

Here, $\boldsymbol{\theta}_m = \{\theta_1, \theta_2, \cdots, \theta_m\}$, $m = 2^k$, and $\phi_i(\mathbf{x}, \boldsymbol{\varphi})$ are basis functions defined by

$$\phi_j(\mathbf{x}, \boldsymbol{\varphi}_k) = \prod_{i=1}^{k} R_j(\varphi_i), \quad \text{where} \quad R_j(\varphi_i) = \begin{cases} H(\varphi_i), & \text{if } b_i^j = 0 \\ 1 - H(\varphi_i), & \text{if } b_i^j = 1 \end{cases} \tag{7.2.30}$$

where $(b_1^j, b_2^j, \cdots, b_k^j) = \text{bin}(j-1)$ is the binary representation of integer $(j-1)$. When the true number of zones m is less than 2^k, (7.2.29) can still be used by deleting all terms associated with empty zones.

Let us consider the case of using two level set functions ($k = 2$) to represent a four-zone structure ($m = 4$) shown in Fig. 7.6. From (7.2.29), we have

$$\begin{aligned} \Omega_1 &= \{\mathbf{x} \mid \varphi_1(\mathbf{x}) > 0,\ \varphi_2(\mathbf{x}) > 0\} \\ \Omega_2 &= \{\mathbf{x} \mid \varphi_1(\mathbf{x}) > 0,\ \varphi_2(\mathbf{x}) < 0\} \\ \Omega_3 &= \{\mathbf{x} \mid \varphi_1(\mathbf{x}) < 0,\ \varphi_2(\mathbf{x}) > 0\} \\ \Omega_4 &= \{\mathbf{x} \mid \varphi_1(\mathbf{x}) < 0,\ \varphi_2(\mathbf{x}) < 0\} \end{aligned} \tag{7.2.31}$$

Using (7.2.31), we can represent the four-value, piecewise-constant function $\hat{\theta}(\mathbf{x})$ by

$$\begin{aligned} \hat{\theta}(\mathbf{x}) = {} & \theta_1 H(\varphi_1) H(\varphi_2) + \theta_2 H(\varphi_1)(1 - H(\varphi_2)) \\ & + \theta_3 (1 - H(\varphi_1)) H(\varphi_2) + \theta_4 (1 - H(\varphi_1))(1 - H(\varphi_2)). \end{aligned} \tag{7.2.32}$$

There are two approaches of using the level set method for inversion: *the evolution approach* and the *optimization approach* (Santosa 1996; van Dijk et al. 2013). The first approach is a traditional one. It moves interfaces gradually from their initial locations toward optimal locations such that the fitting residual is gradually decreased until it is minimized. In the second approach, the level set functions in (7.2.29) are considered as shape parameters of parameterization and identified directly by solving the EIP. The following is a brief description of these two approaches.

7.2.4.2 The Evolution Approach for Level Set Inversion

We start from an initial guess of $\varphi(\mathbf{x})$ and then use an iterative process to modify it gradually based on the criterion that the objective function is gradually decreased

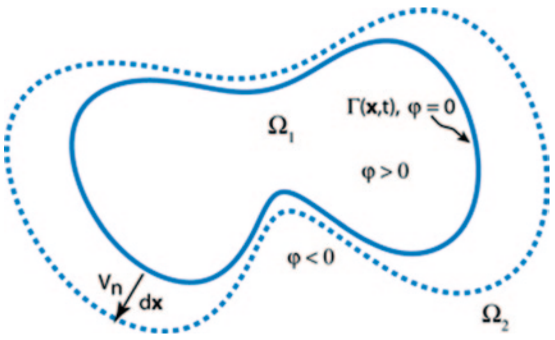


Fig. 7.10 Illustration of the level set function for a two-zone case

until the minimum is reached. During this process, the interface Γ moves and deforms until reaching the optimal location. In the traditional level set method, such an iterative process is simulated by an evolution process by introducing an artificial time variable t. Again, using the two-zone case as an example, we can redefine the level set function (7.2.27) as

$$\begin{cases} \varphi(\mathbf{x},t) > 0, \text{when } \mathbf{x} \in \Omega_1(t) \\ \varphi(\mathbf{x},t) = 0, \text{when } \mathbf{x} \in \Gamma(t) \\ \varphi(\mathbf{x},t) < 0, \text{when } \mathbf{x} \in \Omega_2(t) \end{cases}. \tag{7.2.33}$$

Figure 7.10 illustrates the movement of the interface for the two-zone case in two-dimensional. The initial guess is now the initial condition $\varphi(\mathbf{x},0)$, the result of the nth iteration is denoted by $\varphi(\mathbf{x},t_n)$, and the final result is represented by $\varphi(\mathbf{x},\infty)$. According to this definition, we always have zero level set $\varphi(\mathbf{x}(t),t) = 0$ for $\mathbf{x}(t)$ on $\Gamma(t)$, and the movement of $\Gamma(t)$ is described by

$$\frac{\partial \varphi}{\partial t} + \mathbf{v}\cdot\nabla\varphi = 0,\ \varphi(\mathbf{x},0) = \varphi_0(\mathbf{x}), \tag{7.2.34}$$

where $\mathbf{v}(\mathbf{x},t)$ is the moving velocity of $\mathbf{x}(t)$ on $\Gamma(t)$, $\varphi_0(\mathbf{x})$ is the initial level set function, and $\varphi_0(\mathbf{x})$=0 determines the initial location of Γ. Because the normal direction to $\Gamma(t)$ is $\mathbf{n} = \nabla\varphi / \left|\nabla\varphi\right|$, where $\left|\nabla\varphi\right| = \sqrt{\nabla\varphi\cdot\nabla\varphi}$, (7.2.34) can be rewritten as

$$\frac{\partial \varphi}{\partial t} + v_n \left|\nabla\varphi\right| = 0,\ \varphi(\mathbf{x},0) = \varphi_0(\mathbf{x}). \tag{7.2.35}$$

Equation (7.2.35) is a type of the Hamilton–Jacobi equation and called the *level set equation*, where $v_n(\mathbf{x},t) = \mathbf{v}\cdot\mathbf{n}$ is the moving velocity of $\mathbf{x} \in \Gamma$ at time t along the normal direction.

Our task now is to select an appropriate $v_n(\mathbf{x},t)$ such that the objective function will be decreased after each move. Consider the variation of objective function S due to a small normal movement of boundary Γ

$$\begin{aligned}\delta S &= \int_{\Omega} \frac{\partial S}{\partial \theta} \delta\theta(\mathbf{x}) d\Omega \\ &= \int_{\Gamma} \frac{\partial S}{\partial \theta} (\theta_2(\mathbf{x}) - \theta_1(\mathbf{x})) v_n(\mathbf{x}) dl,\end{aligned} \tag{7.2.36}$$

in which dl is an arc length along Γ and the variation of S is related to parameter variation $\delta\theta$. When a point $\mathbf{x} \in \Gamma$ moves to a new incremental location $\mathbf{x} + d\mathbf{x}$ along the normal direction of Γ, where $d\mathbf{x}=v_n\mathbf{n}$, the corresponding increment of $\theta(\mathbf{x})$ will be $\theta(\mathbf{x} + d\mathbf{x}) - \theta(\mathbf{x}) = \theta_2 - \theta_1$ in the current case. If we select $v_n(\mathbf{x}, t)$ as

$$v_n(\mathbf{x}, t) = -(\theta_2 - \theta_1)(\partial S / \partial \theta), \tag{7.2.37}$$

the variation δS will always be negative, and as a result, the objective function S will always be decreased by the movement of Γ. Substituting the so-determined $v_n(\mathbf{x}, t)$ into (7.2.35), we can obtain $\varphi(\mathbf{x}, t)$ and finally $\varphi(\mathbf{x}, \infty)$.

For detailed discussions on this traditional approach and its numerical solutions, readers may refer to Burger and Osher (2005), Santosa (1996), and Burger (2003). Extensions of the method to the multizone case with multiple level set functions can be found in Tai and Chan (2004) and DeCezaro et al. (2009). The gradient descent algorithm described above usually needs many iterations to converge, especially for low-sensitivity problems (Aghasi et al. 2011).

Example 7.5 *Identification of Zone Shapes Using Evolution Method*
In this example, the level set method is exemplified for identifying the shape of a low-permeability zone in an otherwise homogeneous aquifer. The size of the two-dimensional aquifer is 1000 m × 500 m and is subject to a constant head boundary condition of 10 m on the left-hand side, constant flux boundary condition at the lower-right segment, and no-flow boundary conditions on the rest of the boundary segments. The shape of the actual low-permeability zone is elliptical. Figure 7.11a shows the steady-state head distribution, as well as the actual location of the low-permeability zone. For illustration purposes, we assume that the hydraulic conductivity values of the background (K_1) and low-permeability zone (K_2) are given as 5 and 0.01 m/day, respectively. Thus, only the shape of the low-permeability zone needs to be identified. The steady-state solution is obtained using a finite element code. Head observations are taken from the monitoring network (white open circles) shown in Fig. 7.11a.

A level set toolbox developed by Sumengen (2004) is used to evolve the level set function. The toolbox implements an accurate, third-order essentially non-oscillatory (ENO3) finite difference scheme for solving the level set equation (7.2.35). The domain of the problem is discretized uniformly into 5 m × 5 m cells. Initially, the shape of the zero level set, $\varphi(\mathbf{x}, t = 0)$, is approximated by a square, as shown in Fig. 7.11b (the thick green line). After each iteration, the normal velocity $v_n(\mathbf{x}, t)$ at each cell is updated according to the following expression:

$$v_n(\mathbf{x}, t) = -\operatorname{sgn}(K_2 - K_1)\mathbf{J}^T[\mathbf{h}(\mathbf{x}_D) - \mathbf{h}_D^{obs}], \tag{7.2.38}$$

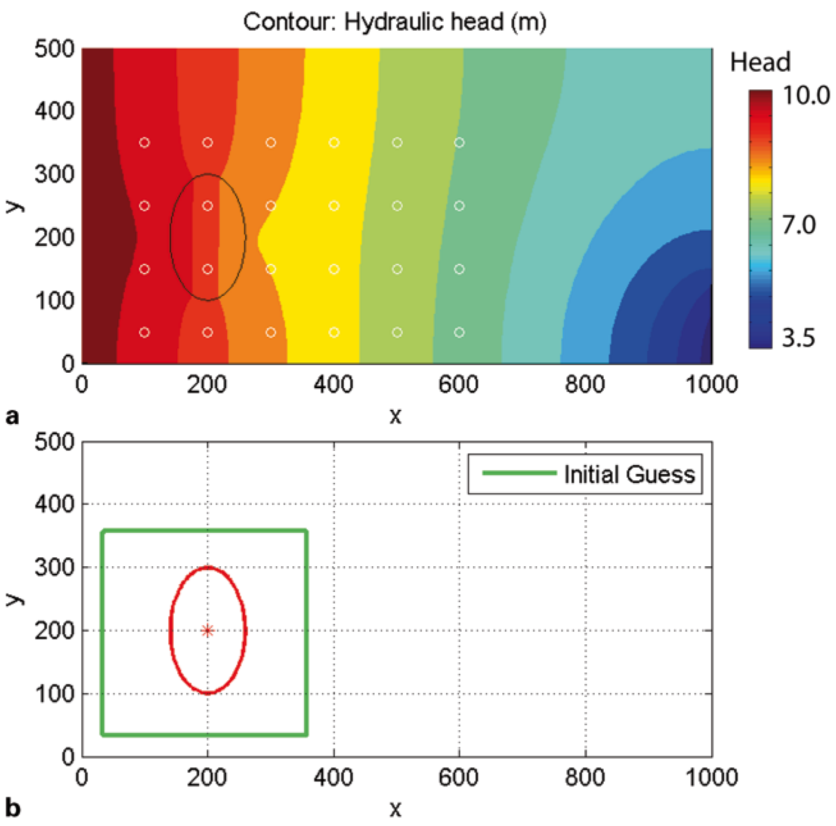


Fig. 7.11 **a** Contour of steady-state head distribution and locations of observation points (*white circles*) and **b** initial guess of level set function, $\varphi(\mathbf{x}, t = 0)$ (thick *green line*). The *ellipse* shape corresponds to the actual low-permeability zone. The domain size is 1000 m × 500 m

which is derived from (7.2.37) by expanding the derivative $\partial S / \partial \hat{\theta}$ with respective to the state variable, $\mathbf{h}$, using the chain rule. In (7.2.38), $\mathbf{h}_D^{obs}$ is a 24-element head observation vector (at steady state), $\mathbf{h}(\mathbf{x}_D)$ is the corresponding model predictions, and the Jacobian **J** is obtained by taking derivative of $\mathbf{h}(\mathbf{x}_D)$ with respect to the two hydraulic conductivity zones. After each evolution step, the zero level set is used to delineate an updated hydraulic conductivity field which, in turn, is used to obtain an updated head field by solving the flow problem. The stopping criterion used for this simple case is the RMSE between simulated and observed head values. Figure 7.12 shows the shape of zero level set at different times, which shows that the zero level set converges to the shape of the actual low-permeability zone.

If in addition to the zone geometry, the hydraulic conductivity values are also unknown, a CIP problem needs to be solved in each loop. Like in other aforementioned EIP methods, updating the Jacobian after each iteration is the most time-consuming part of the level set method.

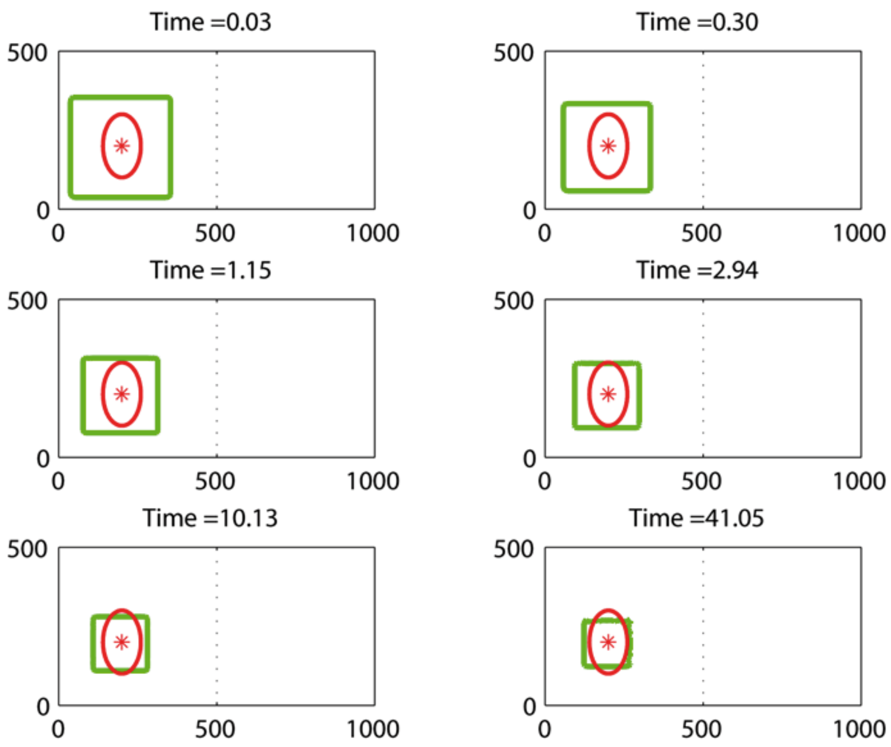


Fig. 7.12 Evolution of the zero level set

7.2.4.3 The Optimization Approach for Level Set Inversion

When (7.2.29) is used for parameterization, the level set inversion method turns into an adaptive parameterization method, in which the level set functions play the role of shape parameters. After the number of zones (m) and the number of level set functions (k) are determined, both $\boldsymbol{\theta}_m$ and $\boldsymbol{\varphi}_k$ can be identified by solving EIP. Following Sect. 7.2.1, the discrete representation of (7.2.29) is

$$\boldsymbol{\theta}_N = \mathbf{G}(\boldsymbol{\varphi}_k)\boldsymbol{\theta}_m. \tag{7.2.39}$$

Here, $\boldsymbol{\theta}_N$ is the model input parameter vector for all nodes, and $\mathbf{G}(\boldsymbol{\varphi}_k)$ is an $N \times m$ matrix with elements g_{ij}

$$g_{ij} = \begin{cases} 1, & i \in \text{ zone } j \\ 0, & \text{otherwise} \end{cases}.$$

The distribution pattern of nonzero elements in the matrix is determined by $\boldsymbol{\varphi}_k$. Using (7.2.5), we obtain the following objective function for EIP solution:

$$S(\boldsymbol{\varphi}_k, \boldsymbol{\theta}_m) = \left\| \mathbf{u}_D[\mathbf{G}(\boldsymbol{\varphi}_k)\boldsymbol{\theta}_m] - \mathbf{u}_D^{obs} \right\| + \alpha R(\boldsymbol{\varphi}_k, \boldsymbol{\theta}_m). \tag{7.2.40}$$

In level set inversion, more regularization terms may be added to the objective function, such as prior information, smoothness, and simple geometry of the zone boundaries (DeCezaro et al. 2009; Dorn and Lesselier 2009).

Note that because level set functions are distributed functions over the whole region Ω, they must be parameterized to make them identifiable. Using the general representation of parameterization, a level set function can be approximated by

$$\hat{\varphi}(\mathbf{x}, \mathbf{c}, \mathbf{a}) = \sum_{l=1}^{r} c_l \psi_l(\mathbf{x}, \mathbf{a}). \tag{7.2.41}$$

Here, r is the dimension of parameterization, $\mathbf{c} = \{c_1, c_2, \cdots, c_r\}$ are coefficients, $\{\psi_l\}$ are basis functions, and $\mathbf{a}$ is the shape vector. Linear basis functions are often used because of their simplicity and low DOF, but other basis functions are also suggested. For example, the RBF is introduced in Sect. 6.1, in which the shape vector $\mathbf{a}$ consists of the centers of the RBF and both $\mathbf{c}$ and $\mathbf{a}$ need to be identified (Aghasi et al. 2011; Gelas et al. 2007; van Dijk et al. 2013; Wang et al. 2003). Introducing appropriate shape parameters may increase the flexibility of $\hat{\varphi}(\mathbf{x}, \mathbf{c}, \mathbf{a})$ for fitting different shapes.

After all k level set functions in (7.2.29) are parameterized, we have $\hat{\boldsymbol{\varphi}}_k(\mathbf{x}, \boldsymbol{\mu})$, where $\boldsymbol{\mu} = \{\mathbf{c}_1, \mathbf{a}_1, \mathbf{c}_2, \mathbf{a}_2, \cdots, \mathbf{c}_k, \mathbf{a}_k\}$. The problem of level set function identification now becomes identification of a finite-dimensional shape vector $\boldsymbol{\mu}$, and the EIP can be solved by methods introduced in Sect. 7.2.1. Let us explain how to calculate the gradients $\partial S / \partial \boldsymbol{\theta}$ and $\partial S / \partial \boldsymbol{\mu}$ with the two-zone case. After parameterization, (7.2.28) can be rewritten as

$$\hat{\theta}(\mathbf{x}) = \theta_1 H(\hat{\varphi}(\mathbf{x}, \boldsymbol{\mu})) + \theta_2 [1 - H(\hat{\varphi}(\mathbf{x}, \boldsymbol{\mu}))]. \tag{7.2.42}$$

The derivative of Heaviside function $H(\varphi)$ is the Dirac-delta function $\delta(\varphi)$. For numerical calculation, it is often replaced by a smooth function $\delta_\varepsilon(\varphi) = \varepsilon / \pi(\varphi^2 + \varepsilon^2)$, where ε is a smoothing factor. Tai and Chan (2004) found that $\delta(\varphi)$ can be replaced simply by 1 without changing the gradient direction of the objective function in some applications. Now, as in the evolution level set inversion, we need to calculate the sensitivity matrix using methods introduced in Chap. 5,

$$\frac{\partial S}{\partial \theta_1} = \sum_{i=1}^{N} \frac{\partial S}{\partial \theta_i} H(\hat{\varphi}) = \sum_{i \in (\Omega_1)} \frac{\partial S}{\partial \theta_i}, \tag{7.2.43}$$

$$\frac{\partial S}{\partial \theta_2} = \sum_{i=1}^{N} \frac{\partial S}{\partial \theta_i} [1 - H(\hat{\varphi})] = \sum_{i \in (\Omega_2)} \frac{\partial S}{\partial \theta_i}. \tag{7.2.44}$$

In addition, for any component μ_j of the shape vector $\boldsymbol{\mu}$, we need to calculate

$$\frac{\partial S}{\partial \mu_j} = \sum_{i=1}^{N} \frac{\partial S}{\partial \theta_i} \frac{\partial \theta_i}{\partial H} \frac{\partial H}{\partial \hat{\varphi}} \frac{\partial \hat{\varphi}}{\partial \mu_j} = \sum_{i=1}^{N} \frac{\partial S}{\partial \theta_i} (\theta_1 - \theta_2) \delta_\varepsilon(\hat{\varphi}) \frac{\partial \hat{\varphi}}{\partial \mu_j}. \tag{7.2.45}$$

In the above equations, $\partial S / \partial \theta_i$ for all nodes ($i = 1, 2, \cdots, N$) are calculated by the adjoint-state method. When a Gauss–Newton method is used for optimization, the sensitivity matrix $\mathbf{J}_{\boldsymbol{\mu}} = [\partial u_i / \partial \mu_j]$ can be calculated similarly.

For more discussion on using the optimization method for level set inversion, readers may refer to, for example, Tai and Chan (2004), Dorn and Lesselier (2009), and Ahgasi et al. (2011). The level set inversion method has been used for identifying the hydraulic conductivity structure in aquifers and the permeability distribution in petroleum reservoirs (Berre et al. 2007; Iglesias and McLaughlin 2011; Dorn and Villegas 2008; Berre et al. 2009; Aghasi et al. 2013; Lien and Mannseth 2014). The level set method has been combined with Bayesian inversion (Cardif and Kitanidis 2009), ensemble Kalman filter (Chang et al. 2010), and the multiscale inversion method (Berre et al. 2007, 2009; Lien et al. 2005), which will be introduced in the next section.

7.3 Multiscale Inversion

7.3.1 Multiscale Refinement

Multiscale refinement, or multigrid refinement, is a special case of zone refinement, in which the zonation pattern is obtained by making the grid progressively finer. Algorithms for this method are very similar to those used in Sect. 7.2.2 and 7.2.3:

1. A homogeneous model is formed initially over a coarse grid.
2. A sensitivity-based selection method is used to determine which blocks of the coarse grid should be refined in order to decrease the value of the objective function as much as possible.
3. The selected blocks are replaced by finer-scale blocks to form a new structure.
4. The parameter values of all blocks of the new structure are identified by solving CIP.
5. The stopping criteria are checked for convergence. If not converged, return to Step 2.

Grimstad et al. (2003) used finite difference grids for refinement (Fig. 7.13a) and presented a stopping criterion

$$S(\boldsymbol{\theta}) < (n - m) + \sqrt{2(n - m)}, \tag{7.3.1}$$

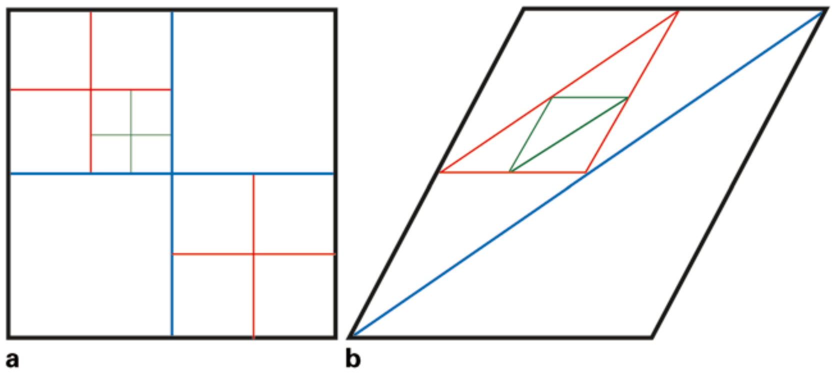


Fig. 7.13 Illustration of grid refinement for multiscale inversion. **a** Finite difference grid and **b** finite element mesh

where n is the number of observation data and m is the dimension of $\boldsymbol{\theta}$. Note that m is increased when the grid becomes finer. In Majdalani and Ackerer (2011), multiscale finite element meshes are used for refinement, in which a coarse triangle element is divided into four finer triangle elements (Fig. 7.13b). The authors applied their method to a real case study.

The multiscale refinement method avoids the difficulty of identifying shape parameters, but at the expense of losing flexibility in structure parameterization. Because structural patterns often cannot be well characterized by using a small number of regular blocks, an adaptive process of multiscale refinement is needed to generate high-resolution grids for identifying irregular zone boundaries. In this case, it is difficult to avoid the overparameterization problem—with the decrease of the block size, the identified block parameter values would become very insensitive to all observation data, and as a result, the CIP solution becomes unstable. Berre et al. (2007) attempted to overcome this problem by combining the adaptive multiscale estimation with the level set inversion, in which the former is used as a predictor, while the latter is used as a corrector. Berre et al. (2009) presented another way to combine the two methods, in which the multiscale refinement algorithm is applied to the level set functions, instead of the unknown parameter itself. In their method, bilinear interpolation is used to determine the basis functions in (7.2.41) without using any shape parameter. After substituting (7.2.41) into (7.2.29), the unknown parameter is determined completely by coefficients $\{\boldsymbol{\theta}_m, \mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_k\}$, which can be identified by minimizing the objective function. This process is repeated for progressively finer grids until the stopping criteria are satisfied. The adaptive multiscale refinement process can also be combined with Bayesian inversion for the estimation of statistical parameters (Wan and Zabaras 2011).

7.3.2 Multiscale Modeling

If a distributed parameter can be identified over a fine-resolution grid, the problem of identifying its structure becomes insignificant because the structure error becomes small. In other words, we may use the increase of parameterization dimension to compensate for the reduction in the number of shape parameters. Unfortunately, in reality, we cannot identify a distributed parameter directly at a very fine grid because overparameterization will cause the inverse solution unstable. The multiscale inversion approach provides another way to deal with this problem. After a model is identified at a coarse scale (a low-resolution grid), it is used as a high-level regularization mechanism to stabilize the inverse solution at a finer scale. This approach is different from the multiscale refinement method discussed in the previous subsection, where only some selected blocks of a grid are refined and the forward problem is solved in a fixed fine grid independent of the inverse solution. In multiscale inversion, a series of models are identified adaptively at a series of nested coarsening grids and both forward and inverse problems are solved at the same scale. In other words, different models are constructed for different scales. The following is a brief introduction of this approach.

7.3.2.1 Upscaling and Downscaling

Let us consider two scales, a coarse scale (c) and a fine scale (f), which are characterized by two nested grids with N_c and N_f nodes, respectively (Fig. 7.14). The models associated with the two scales are denoted by $\mathcal{M}(\mathbf{u}_c, \boldsymbol{\theta}_c)$ and $\mathcal{M}(\mathbf{u}_f, \boldsymbol{\theta}_f)$, in which $\mathbf{u}_c$ and $\mathbf{u}_f$ are the forward solutions and $\boldsymbol{\theta}_c$ and $\boldsymbol{\theta}_f$ are the respective parameterizations of the unknown parameter. The relation between $\boldsymbol{\theta}_c$ and $\boldsymbol{\theta}_f$ can be

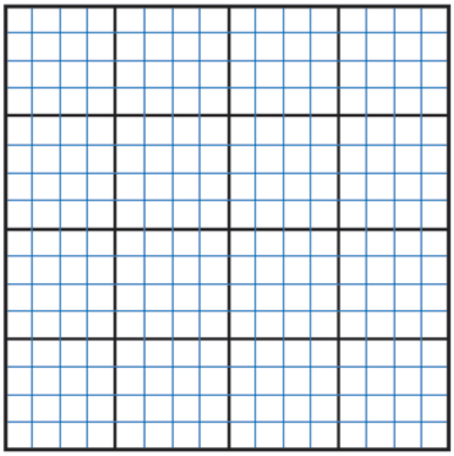


Fig. 7.14 Two nested grids, in which the coarse grid is shown by *thick dark lines*, whereas the fine grid is shown by *thin blue lines*

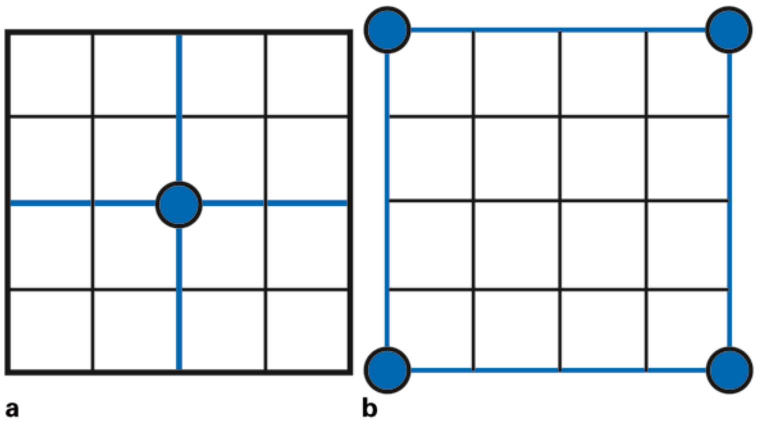


Fig. 7.15 Information exchange between scales. **a** Upscaling, in which the coarse-scale parameter value $\boldsymbol{\theta}_c$ (at the center circle) is obtained from surrounding fine-scale $\boldsymbol{\theta}_f$ values by averaging and **b** downscaling, in which all fine-scale $\boldsymbol{\theta}_f$ values within a block are obtained from the corresponding coarse-scale $\boldsymbol{\theta}_c$ (*corner circles*) through interpolation

represented by two transformations: an upscaling transformation $\boldsymbol{\theta}_c = U(\boldsymbol{\theta}_f)$ and a downscaling transformation $\boldsymbol{\theta}_f = D(\boldsymbol{\theta}_c)$. Figure 7.15a–b show how local nodal values are used to generate upscaled and downscaled parameters. Note that various averaging methods can be considered as upscaling transformations, whereas various interpolation methods can be considered as downscaling transformations.

When linear transformations are used as usual, upscaling and downscaling can be represented, respectively, by

$$\boldsymbol{\theta}_c = \mathbf{U}\boldsymbol{\theta}_f, \tag{7.3.2}$$

and

$$\boldsymbol{\theta}_f = \mathbf{D}\boldsymbol{\theta}_c, \tag{7.3.3}$$

where $\mathbf{U}$ is an $N_c \times N_f$ matrix for upscaling and $\mathbf{D}$ is an $N_f \times N_c$ matrix for downscaling. Note that the form of (7.3.3) is identical to (7.2.2), where the matrix $\mathbf{D}$ was called the structure matrix of parameterization and denoted by $\mathbf{G}$.

Different methods of upscaling and downscaling will generally lead to different results. The parameter values obtained from downscaling can be used directly for forward solution. For example, during the identification of a parameterized model, the grid parameter values are obtained from the identified parameter vector (downscaling) and used for calculating the values of the objective function; if a software package has a grid refinement function, the grid parameter values will be generated automatically by interpolation (downscaling) and used for model prediction. On the other hand, however, the parameter values obtained by local homogenization (upscaling) cannot be used directly in a coarse-scale grid for obtaining the forward solution. In EWR modeling, upscaling depends not only on the local structure, but also on global flow conditions, including boundary conditions (Renard and De Marsily 1997; Gerritsen and Durlofsky 2005; Farmer 2002). In multiscale inversion, however, parameter values obtained by upscaling or downscaling are only used as initial estimation or regularization for stabilizing the inverse solution. As described in the following algorithm, different models are constructed for different scales. Therefore, the effect of upscaling and downscaling errors is insignificant.

7.3.2.2 Two-Scale Inversion

A simple, two-scale inversion process consists of the following major steps:

1. Estimate a fine-scale model $\boldsymbol{\theta}_f^0$. The objective function used for fine scale inversion is

$$S_f(\boldsymbol{\theta}_f) = \left\|\mathbf{u}_f(\boldsymbol{\theta}_f) - \mathbf{u}_f^{obs}\right\| + \alpha_f R_f(\boldsymbol{\theta}_f), \tag{7.3.4}$$

   where $\mathbf{u}_f^{obs}$ is observation data and $\alpha_f R_f(\boldsymbol{\theta}_f)$ is the regularization term for inversion at the fine scale. We assume that the structure error at the fine scale can be ignored.
2. Upscale $\boldsymbol{\theta}_f^0$ to obtain an initial coarse-scale model, $\boldsymbol{\theta}_c^0 = \mathbf{U}\boldsymbol{\theta}_f^0$.
3. Solve CIP at the coarse scale to obtain an inverse solution $\hat{\boldsymbol{\theta}}_c$ by minimizing the coarse-scale objective function

$$S_c(\boldsymbol{\theta}_c) = \left\|\mathbf{u}_c(\boldsymbol{\theta}_c) - \mathbf{u}_c^{obs}\right\| + \alpha_c R_c(\boldsymbol{\theta}_c), \tag{7.3.5}$$

   where $\mathbf{u}_c^{obs}$ is a modified observation data vector and the determination of which will be discussed below.
4. Downscale $\hat{\boldsymbol{\theta}}_c$ to obtain a fine-scale model, $\hat{\boldsymbol{\theta}}_f = \boldsymbol{\theta}_f^0 + \mathbf{D}(\hat{\boldsymbol{\theta}}_c - \boldsymbol{\theta}_c^0)$.
5. Starting from $\hat{\boldsymbol{\theta}}_f$, solve CIP at the fine scale until convergence.

The most important issue in the above process is to keep the value of the objective function $S_f(\boldsymbol{\theta}_f)$ decreasing monotonically after scale change. Because $\mathbf{u}_c(\boldsymbol{\theta}_c)$ and $\mathbf{u}_f(\boldsymbol{\theta}_f)$ are obtained by solving the forward problem at different scales, their values are generally different. If we enforce the consistence condition, $\mathbf{u}_c(\mathbf{U}\boldsymbol{\theta}_f) - \mathbf{u}_c^{obs} = \mathbf{u}_f(\boldsymbol{\theta}_f) - \mathbf{u}_f^{obs}$, namely, let the following hold at observation locations/times $\mathbf{x}_{obs}$

$$\mathbf{u}_c^{obs} = \mathbf{u}_f^{obs} + [\mathbf{u}_c(\mathbf{U}\boldsymbol{\theta}_f, \mathbf{x}_{obs}) - \mathbf{u}_f(\boldsymbol{\theta}_f, \mathbf{x}_{obs})], \tag{7.3.6}$$

then the two objective functions $S_c(\boldsymbol{\theta}_c)$ and $S_f(\boldsymbol{\theta}_f)$ will become consistent: $\mathbf{u}_c(\hat{\boldsymbol{\theta}}_c) \rightarrow \mathbf{u}_c^{obs}$ in the coarse scale implies $\mathbf{u}_f(\hat{\boldsymbol{\theta}}_f) \rightarrow \mathbf{u}_f^{obs}$ in the fine scale. The term in the square brackets of (7.3.6) is the upscaling error or bias of forward solution. Furthermore, we need to ensure that the gradients of the two objectives are equal so that when $\hat{\boldsymbol{\theta}}_c$ is the minimizer at the coarse scale, $\hat{\boldsymbol{\theta}}_f$ will be the minimizer at the fine scale. In order to satisfy this requirement, let

$$\mathbf{r}_c = \nabla S_c(\mathbf{U}\boldsymbol{\theta}_f) - \nabla S_f(\boldsymbol{\theta}_f)\mathbf{D}, \tag{7.3.7}$$

and redefine the objective function of the coarse scale by

$$\tilde{S}_c(\boldsymbol{\theta}_c) = S_c(\boldsymbol{\theta}_c) - \mathbf{r}_c\boldsymbol{\theta}_c. \tag{7.3.8}$$

Note that the dimension of gradient residual $\mathbf{r}_c$ is $m \times N_c$, where m is the number of observations. By calculating the gradient of this equation, we can find that $\tilde{S}_c(\boldsymbol{\theta}_c)$ satisfies the equal gradient requirement. The gradient residual $\mathbf{r}_c$ in (7.3.7) can be calculated by the adjoint-state method. Note that the scale difference between the two scales should not be too great; otherwise, large errors in $\mathbf{u}_c^{obs}$ and large value of $\mathbf{r}_c$ will cause the algorithm to fail.

In practical applications, multiscale inversion often involves a hierarchy of scales, which starts from a fine scale, followed by several intermediate scales, and then to a coarse scale. Upscaling is first performed at each scale, followed by downscaling, and the inversion is completed for each scale. Although multiscale inversion involves relatively complex algorithm and computation, it has attractive advantages. First, the major effort of inverse solution is completed at the coarse scale. At this scale, the forward solution becomes faster because of the smaller number of nodes, the inverse solution becomes more stable, and no structure parameters need to be identified. Second, once the optimal parameter is found at the coarse scale, the downscaled solution will provide a good initial guess to guide the inverse solution at the fine scale. Finally, a more detailed characterization of the unknown parameter is obtained from the available data.

Multiscale inversion is a promising approach for model calibration when data are sufficient. Unfortunately, in EWR modeling, the available data are usually insufficient for identifying the parameter values of all blocks even at a coarse scale. Up to date, this approach has not become popular. Yoon et al. (2001) applied multigrid inversion in history matching; starting with a coarse description, the production history at the wells is matched by recursively refining the reservoir grid. Because production data are integrated at a coarse scale with fewer parameters, the authors found that their method is significantly faster than direct fine-scale inversion of the production data. Similarly, Aanonsen and Eydinov (2006) applied multiscale inversion to reservoir history matching, in which the objective functions for different scales are modified by recalculating the regularization coefficients for each scale. The multilevel model correction method presented by Li and Zou (2007) avoids the use of different regularization coefficients at different scales; instead, the authors added a correction term to the objective functions of coarse scales, similar to that used in the above algorithm. The method presented by Fu et al. (2010) uses the adjoint-state method to reduce the computational effort of inversion at a fine scale, in which the gradient of objective function is calculated at a coarse scale instead of fine scale.

7.4 Statistical Structure Parameter Estimation

7.4.1 Statistical Parameterization and Hyperparameters

In Chap. 4, the unknown parameter $\theta(\mathbf{x})$ is regarded as a realization of a random field, where the random field is assumed to be known and characterized by some statistical parameters. Let $\{\theta_j = \theta(\mathbf{x}_j) \mid j = 1, 2, \cdots, m\}$ be *m* samples of the parameter taken from *m* locations (or times), $\mathbf{v} = \{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_m}\}$, and linear interpolation in the statistical framework has the following general form:

$$\theta(\mathbf{x}) \approx \hat{\theta}(\mathbf{x}) = \sum_{j=1}^{m} \theta_j \phi_j(\mathbf{x}, \mathbf{v}, \boldsymbol{\psi}), \tag{7.4.1}$$

where $\boldsymbol{\phi} = \{\phi_1, \phi_2, \cdots, \phi_m\}$ is a set of orthonormal basis functions satisfying:

$$\phi_j(\mathbf{x}_i) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

and $\boldsymbol{\psi} = \{\psi_1, \psi_2, \cdots, \psi_r\}$ is a set of *statistical parameters* that characterize the structure of the random field. Recall that similar forms of parameterization have been used extensively in Chap. 6 for both deterministic (Sect. 6.1.1) and statistical parameterization methods (Sect. 6.1.5).

When interpolation (7.4.1) is used for parameterization, $\hat{\theta}(\mathbf{x})$ is a model of $\theta(\mathbf{x})$, *m* is the dimension of parameterization, and $\boldsymbol{\theta} = \{\theta_1, \theta_2, \cdots, \theta_m\}$ is a vector of parameter values to be estimated. A model under statistical parameterization can also be represented by the pair $(\mathbf{S}, \boldsymbol{\theta})$, where $\mathbf{S} = \{m, \boldsymbol{\phi}, \mathbf{v}, \boldsymbol{\psi}\}$ is the model structure. In Chap. 6, we have shown that different types of model structures can be represented by choosing different types of basis functions. To represent a complex structure, we can either increase the dimension of parameterization or increase the number of statistical parameters. When the model structure is predetermined and only the value vector $\boldsymbol{\theta}$ needs to be identified, the number of DOF of the estimation problem is *m*; when *r* statistical parameters need to be estimated, the number of DOF is increased to $m + r$; the number of DOF will be increased further when the shape vector $\mathbf{v}$ needs to be identified. The full structure identification also includes the determination of the parameterization dimension and the selection of basis functions. In this section, we will focus on the estimation of statistical parameters.

Statistical parameters are hyperparameters of inverse solution. Although estimating hyperparameters is not the direct objective of inversion because they are not the model parameters, we have to estimate them as part of the inversion process because the hyperparameters affect model structures and are uncertain. In this section, we will use $\boldsymbol{\psi}$ to represent all hyperparameters appearing in the formulation of inversion. The following lists some examples of hyperparameters that we have seen before:

- A typical example of statistical parameterization is kriging interpolation introduced in Sect. 6.2.4, in which the kriging weighting coefficients are obtained by solving the kriging equations. The kriging coefficients, $\{\lambda_j(\mathbf{x}, \boldsymbol{\psi}) \mid j = 1, 2, \cdots, m\}$, play the role of basis functions in (7.4.1), where $\boldsymbol{\psi}$ is the statistical parameters in a selected variogram model, such as variance σ^2 and range l in the exponential and Gaussian models. In Sect. 6.2.3, these two parameters are estimated by samples of the unknown $\theta(\mathbf{x})$. But, they can be considered as hyperparameters to be estimated by the observations of state variables.
- The objective functions derived from MAP or MLE contain statistical parameters in the covariance matrices of data and prior distributions, such as the variance of measurements, the mean and variance of Gaussian prior. In Chap. 4, these statistical parameters are assumed to be known, but in practice, they are usually unknown and need to be estimated by data. Such a case was actually presented in Example 4.6, in which Gibbs sampling algorithm was demonstrated for estimating the mean and variance of a Gaussian PDF.
- The regularization coefficients of the regularization terms and the weighting coefficients of the coupled inverse problems cannot be easily determined by methods learned in Chap. 3. Thus, they can be considered as hyperparameters to be estimated as part of inversion.

The simplest method of dealing with hyperparameters is marginalization. Assume that a model has two unknown parameters $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$, and we are only interested in estimating $\boldsymbol{\theta}$ from data D. In this case, we can regard $\boldsymbol{\psi}$ as a nuisance parameter vector and integrate it out by calculating the marginal posterior

$$p(\boldsymbol{\theta} \mid D) = \int p(\boldsymbol{\theta}, \boldsymbol{\psi} \mid D)\, d\boldsymbol{\psi}. \tag{7.4.2}$$

Marginalization technique has been used in the derivation of the Bayes' theorem in Sect. 4.1.3. Because the marginal probability distribution

$$p(D) = \int p(D \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{7.4.3}$$

is a constant after $\boldsymbol{\theta}$ is integrated out, the Bayes' theorem then can be written as $p(\boldsymbol{\theta} \mid D) \propto p(D \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})$.

Marginalization (7.4.2) gives only the average effect of hyperparameter $\boldsymbol{\psi}$ to the estimation of $\boldsymbol{\theta}$. Because $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ are dependent, if $\boldsymbol{\psi}$ can be estimated more accurately, the estimated $\boldsymbol{\theta}$ will become more accurate. In fact, we can estimate them simultaneously by solving a statistical EIP as we have done for the deterministic framework. With the min-min optimization process described in Sect. 7.2.1, for any $\hat{\boldsymbol{\Psi}}$ chosen from its admissible region, we can use a classical statistical inversion method, the MAP for example, to find an optimal parameter $\hat{\boldsymbol{\theta}}$ associated with $\hat{\boldsymbol{\Psi}}$. When $\hat{\boldsymbol{\psi}}$ is changed systematically according to a global optimization method, the optimal parameter pairs $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\psi}})$ then can be found. In general, the fitting residual is significantly decreased during this process. As shown in the following example, the optimal $\hat{\boldsymbol{\psi}}$ can be found explicitly for simple cases.

Example 7.6 *Estimation of Unknown Variance*
Consider the identification of transmissivity $T(x,y)$ and storage coefficient $S(x,y)$ of a confined aquifer, where head observations and prior estimation of the two parameters are available. Let the observed heads be an n-dimensional vector $\mathbf{u}_D^{obs}$. After discretization and parameterization, T and S are represented by an m_T-dimensional vector $\mathbf{T}$ and an m_S-dimensional vector $\mathbf{S}$, respectively. We assume that all data and prior distributions are Gaussian, namely

$$p(\mathbf{e}_D) = \mathcal{N}(\mathbf{0},\mathbf{C}_D), p(\mathbf{T}) = \mathcal{N}(\mathbf{T}^0,\mathbf{C}_T), \text{and } p(\mathbf{S}) = \mathcal{N}(\mathbf{S}^0,\mathbf{C}_S).$$

Furthermore, we assume $\mathbf{C}_D = \sigma_D^2\mathbf{I}$, $\mathbf{C}_T = \sigma_T^2\mathbf{I}$, and $\mathbf{C}_S = \sigma_S^2\mathbf{I}$. In this case, the unknown parameters and hyperparameters are $\boldsymbol{\theta} = \{\mathbf{T},\mathbf{S}\}$ and $\boldsymbol{\psi} = \{\sigma_D^2,\sigma_T^2,\sigma_S^2\}$, respectively. From Sect. 4.2.3, the conjugate MAP estimate of them is

$$(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\psi}}) = \arg\min_{(\boldsymbol{\theta},\boldsymbol{\psi})}\left\{\Phi(\boldsymbol{\theta}\boldsymbol{\theta},\boldsymbol{\psi})\right\}, \tag{7.4.4}$$

where

$$\begin{aligned}\Phi(\boldsymbol{\theta},\boldsymbol{\psi}) = \Big\{&n\log\sigma_D^2 + m_T\log\sigma_T^2 + m_S\log\sigma_S^2\\ &+\sigma_D^{-2}[\mathbf{u}_D^{obs} - \mathbf{u}_D(\mathbf{T},\mathbf{S})]^T[\mathbf{u}_D^{obs} - \mathbf{u}_D(\mathbf{T},\mathbf{S})]\\ &+\sigma_T^{-2}(\mathbf{T}-\mathbf{T}^0)^T(\mathbf{T}-\mathbf{T}^0) + \sigma_S^{-2}(\mathbf{S}-\mathbf{S}^0)^T(\mathbf{S}-\mathbf{S}^0)\Big\}.\end{aligned} \tag{7.4.5}$$

This optimization problem can be solved iteratively. At the *k*th iteration, $\boldsymbol{\psi}^{(k)}$ are known and the above objective function reduces to

$$\begin{aligned}\Phi(\boldsymbol{\theta},\boldsymbol{\psi}^{(k)}) \propto \Big\{&[\mathbf{u}_D^{obs} - \mathbf{u}_D(\mathbf{T},\mathbf{S})]^T[\mathbf{u}_D^{obs} - \mathbf{u}_D(\mathbf{T},\mathbf{S})]\\ &+\alpha_T^{(k)}(\mathbf{T}-\mathbf{T}^0)^T(\mathbf{T}-\mathbf{T}^0) + \alpha_S^{(k)}(\mathbf{S}-\mathbf{S}^0)^T(\mathbf{S}-\mathbf{S}^0)\Big\},\end{aligned} \tag{7.4.6}$$

where the regularization coefficients are

$$\alpha_T = \sigma_D^2 / \sigma_T^2, \quad \alpha_S = \sigma_D^2 / \sigma_S^2. \tag{7.4.7}$$

Problem (7.4.6) is a regularized least squares problem with known regularization coefficients. Let its solution be $\boldsymbol{\theta}^{(k)} = (\mathbf{T}^{(k)},\mathbf{S}^{(k)})$. Substituting $\boldsymbol{\theta}^{(k)}$ into (7.4.5) and using the necessary condition of minimization with respect to σ_D, we obtain

$$\frac{\partial\Phi(\boldsymbol{\theta}^{(k)},\boldsymbol{\psi})}{\partial\sigma_D} = 2\left(\frac{n}{\sigma_D} - \frac{R_{LS}}{\sigma_D^3}\right) = 0 \tag{7.4.8}$$

where $R_{LS} = [\mathbf{u}_D^{obs} - \mathbf{u}_D(\mathbf{T}^{(k)},\mathbf{S}^{(k)})]^T[\mathbf{u}_D^{obs} - \mathbf{u}_D(\mathbf{T}^{(k)},\mathbf{S}^{(k)})]$ is the fitting residual. The solution of (7.4.8) gives an updated estimate of hyperparameter σ_D^2,

$$\left(\sigma_D^2\right)^{(k+1)} = \frac{1}{n} R_{LS}. \tag{7.4.9}$$

Similarly, we can obtain

$$\begin{aligned}\left(\sigma_T^2\right)^{(k+1)} &= \frac{1}{m_T}(\mathbf{T}^{(k)} - \mathbf{T}^0)^T(\mathbf{T}^{(k)} - \mathbf{T}^0) \\ \left(\sigma_S^2\right)^{(k+1)} &= \frac{1}{m_S}(\mathbf{S}^{(k)} - \mathbf{S}^0)^T(\mathbf{S}^{(k)} - \mathbf{S}^0).\end{aligned} \tag{7.4.10}$$

These are biased estimates. Unbiased estimates of them are

$$\begin{aligned}\left(\sigma_D^2\right)^{(k+1)} &= \frac{1}{n - m_T - m_S} R_{LS} \\ \left(\sigma_T^2\right)^{(k+1)} &= \frac{1}{m_T - 1}(\mathbf{T}^{(k)} - \mathbf{T}^0)^T(\mathbf{T}^{(k)} - \mathbf{T}^0) \\ \left(\sigma_S^2\right)^{(k+1)} &= \frac{1}{m_S - 1}(\mathbf{S}^{(k)} - \mathbf{S}^0)^T(\mathbf{S}^{(k)} - \mathbf{S}^0).\end{aligned} \tag{7.4.11}$$

Once we have

$$\boldsymbol{\psi}^{(k+1)} = \left\{\left(\sigma_D^2\right)^{k+1}, \left(\sigma_T^2\right)^{k+1}, \left(\sigma_S^2\right)^{k+1}\right\}$$

we can use it to replace $\boldsymbol{\psi}^{(k)}$ in (7.4.6) for the next iteration until convergence is reached.

7.4.2 Hierarchical Bayesian Inversion

If we have prior information on hyperparameters, we can assume both $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ as random variables and estimate them simultaneously in the Bayesian framework. This approach is called *hierarchical Bayesian* or *full Bayesian* because it may involve also the selection of optimal dimension of parameterization and the optimal type of basis functions (Wikle et al. 1998; Wikle 2003; Gilks et al. 1998). Hierarchical Bayesian incorporates prior distributions of all unknowns into the inversion and also gives the reliability estimate.

7.4.2.1 Joint MAP Estimate

Joint estimation of both $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ is a statistical EIP defined in Sect. 7.1.3. Using a sampling method to search the joint posterior distribution is generally infeasible because of the high dimension of inversion. Nevertheless, there are approaches to find the maximum of the joint posterior distribution. According to (7.1.5), the MAP estimate of $\boldsymbol{\theta}$ and $\boldsymbol{\Psi}$ is given by

$$(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\psi}}) = \arg\max_{(\boldsymbol{\theta}, \boldsymbol{\psi})} \left\{ p_*(\boldsymbol{\theta}, \boldsymbol{\psi}) \right\} \tag{7.4.12}$$

where

$$p_*(\boldsymbol{\theta}, \boldsymbol{\psi}) = p(\boldsymbol{\theta}, \boldsymbol{\psi} \mid \mathbf{d}) \propto p(\mathbf{d} \mid \boldsymbol{\theta}, \boldsymbol{\psi}) p(\boldsymbol{\theta} | \boldsymbol{\psi}) p(\boldsymbol{\psi}). \tag{7.4.13}$$

The above equation is derived from Bayesian inference, where $\mathbf{d}$ is observation data. Once the joint posterior distribution is found, the solution of (7.4.12) can be obtained by the MCMC search process (see Sect. 4.3.2). Like the min-min optimization problem (7.1.6) used for EIP solution, problem (7.4.12) can be solved iteratively by the max-max optimization. From step k to step $k+1$, we need to estimate

$$\hat{\boldsymbol{\theta}}^{(k+1)} = \arg\max_{\boldsymbol{\theta}} \left\{ p_*(\boldsymbol{\theta}, \hat{\boldsymbol{\psi}}^{(k)}) \propto p(\mathbf{d} \mid \boldsymbol{\theta}, \hat{\boldsymbol{\psi}}^{(k)}) p(\boldsymbol{\theta} | \hat{\boldsymbol{\psi}}^{(k)}) \right\}, \tag{7.4.14}$$

$$\hat{\boldsymbol{\psi}}^{(k+1)} = \arg\max_{\boldsymbol{\psi}} \left\{ p_*(\hat{\boldsymbol{\theta}}^{(k+1)}, \boldsymbol{\psi}) \propto p(\mathbf{d} \mid \hat{\boldsymbol{\theta}}^{(k+1)}, \boldsymbol{\psi}) p(\hat{\boldsymbol{\theta}}^{(k+1)} | \boldsymbol{\psi}) p(\boldsymbol{\psi}) \right\}. \tag{7.4.15}$$

In this process, two MAP problems are solved in each iteration.

Example 7.7 *Hierarchical Bayesian for Conjugate Priors*
In Chap. 4, we considered the problem of estimating parameter $\boldsymbol{\theta}$ from observation data $\mathbf{u}_D^{obs} = \mathbf{u}_D(\boldsymbol{\theta}) + \mathbf{e}_D$, where the probability distribution of error term $\mathbf{e}_D$ was assumed known. Now, we assume that $p(\mathbf{e}_D)$ is Gaussian with zero mean and covariance matrix $\mathbf{C}_D = \sigma_D^2 \mathbf{I}$, but the variance σ_D^2 needs to be estimated. Using the Bayesian inference, the posterior distribution of $\boldsymbol{\theta}$ is given by

$$p(\boldsymbol{\theta} | \mathbf{d}) \propto p(\mathbf{d} | \boldsymbol{\theta}) p(\boldsymbol{\theta}), \tag{7.4.16}$$

where $\mathbf{d}=\mathbf{u}_D^{obs}$, $p(\boldsymbol{\theta})$ is the prior distribution of $\boldsymbol{\theta}$, and the likelihood is a Gaussian given by

$$p(\mathbf{d} \mid \boldsymbol{\theta}) = (\beta / 2\pi)^{n/2} \exp\left\{ -\frac{\beta}{2} [\mathbf{d} - \mathbf{u}_D(\boldsymbol{\theta})]^T [\mathbf{d} - \mathbf{u}_D(\boldsymbol{\theta})] \right\}, \tag{7.4.17}$$

where $\beta = 1/\sigma_D^2$. The MRF defined in Sect. 6.2.2 is selected as the prior distribution for $\boldsymbol{\theta}$,

$$p(\boldsymbol{\theta} | \gamma) \propto \gamma^{m/2} \exp\left(-\frac{1}{2} \gamma \boldsymbol{\theta}^T \mathbf{W} \boldsymbol{\theta} \right). \tag{7.4.18}$$

In Bayesian statistics, if the posterior distribution is in the same family as the prior distribution, they are called *conjugate distributions*, and the prior is called a *conjugate prior* of the likelihood function. In Chap. 4, we have seen that Gaussian family is a self-conjugate. In the current case, MRF is a *conjugate prior* of the likelihood (7.4.17) because after multiplication they give rise to a posterior distribution in the same exponential family. By changing the value of the scale parameter γ and the weights in matrix $\mathbf{W}$, this distribution can represent various locally correlated structures. Because of its simplicity and flexibility, MRF is often used as a prior distribution in the field of image processing, as well as spatial statistical modeling. When both β and γ are considered as hyperparameters, the joint posterior distribution will be

$$p(\boldsymbol{\theta}, \beta, \gamma \mid \mathbf{d}) \propto p(\mathbf{d} \mid \boldsymbol{\theta}, \beta) p(\boldsymbol{\theta} \mid \gamma) p(\beta) p(\gamma), \tag{7.4.19}$$

where $p(\mathbf{d} \mid \boldsymbol{\theta}, \beta)$ and $p(\boldsymbol{\theta} | \gamma)$ are given by (7.4.17) and (7.4.18), respectively. In the literature, the gamma distribution is often used as priors for β and γ (Gilks et al. 1998; Besag et al. 1995). Thus, $p(\beta)$ and $p(\gamma)$, also known as hyperpriors, are given by

$$p(\beta) \propto \beta^{a_1 - 1} \exp\left\{-b_1 \beta\right\}, \tag{7.4.20}$$

$$p(\gamma) \propto \gamma^{a_2 - 1} \exp\left\{-b_2 \gamma\right\}. \tag{7.4.21}$$

These two hyperpriors are also conjugate distributions because they keep the posterior distribution (7.4.19) in exponential family after multiplication. Moreover, they are nearly non-informative in the regions $(0, \infty)$ when constants a_1, b_1, a_2 and b_2 are small (Congdon 2006; Robert 2007). Substituting the prior, likelihood, and hyperpriors into (7.4.19), we obtain

$$\begin{aligned} p(\boldsymbol{\theta}, \beta, \gamma \mid \mathbf{d}) \propto\ & \beta^{(n/2 + a_1 - 1)} \exp\left\{-\frac{1}{2}\beta [\mathbf{d} - \mathbf{u}_D(\boldsymbol{\theta})]^T [\mathbf{d} - \mathbf{u}_D(\boldsymbol{\theta})] - b_1 \beta\right\} \\ & \cdot \gamma^{(m/2 + a_2 - 1)} \exp\left(-\frac{1}{2}\gamma \boldsymbol{\theta}^T \mathbf{W} \boldsymbol{\theta} - b_2 \gamma\right). \end{aligned} \tag{7.4.22}$$

Now, we can use MCMC algorithms described in Sect. 4.3.2 to explore this joint posterior distribution to find the hierarchical Bayesian inverse solution $(\hat{\boldsymbol{\theta}}, \hat{\beta}, \hat{\gamma})$. Detailed discussions on the algorithm and its application to recover the release history of a contaminant source can be found in Wang and Zabaras (2006), in which the authors used MRF to model the unknown concentration field; the Gibbs sampler was used to sample the concentration field (one component at a time), and the Metropolis–Hastings sampler was used to estimate PDFs of the hyperparameters of the MRF.

7.4.2.2 Marginalized MAP Estimate

Besides using joint inversion, there are two marginalization-based methods for solving problem (7.4.12), which may be more effective in some cases.

- *Marginalization of hyperparameters*, $\boldsymbol{\psi}$

Integrating out $\boldsymbol{\psi}$ from the joint posterior distribution (7.4.13), the MAP estimation of $\boldsymbol{\theta}$ is given by

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}}\left\{\int p(\mathbf{d} \mid \boldsymbol{\theta}, \boldsymbol{\psi}) p(\boldsymbol{\theta} | \boldsymbol{\psi}) p(\boldsymbol{\psi}) d\boldsymbol{\psi}\right\}. \quad (7.4.23)$$

Then, $\boldsymbol{\psi}$ is estimated by

$$\hat{\boldsymbol{\psi}} = \arg\max_{\boldsymbol{\psi}}\left\{p_*(\hat{\boldsymbol{\theta}}, \boldsymbol{\psi}) \propto p(\mathbf{d} \mid \hat{\boldsymbol{\theta}}, \boldsymbol{\psi}) p(\boldsymbol{\psi})\right\}. \quad (7.4.24)$$

- *Marginalization of the unknown parameters,* $\boldsymbol{\theta}$

First, integrating out $\boldsymbol{\theta}$ from the joint posterior distribution (7.4.13) to find

$$\hat{\boldsymbol{\psi}} = \arg\max_{\boldsymbol{\psi}}\left\{p(\boldsymbol{\psi}) \int p(\mathbf{d} | \boldsymbol{\theta}, \boldsymbol{\psi}) p(\boldsymbol{\theta} | \boldsymbol{\psi}) d\boldsymbol{\theta}\right\}. \quad (7.4.25)$$

Then, $\boldsymbol{\theta}$ is estimated by

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}}\left\{p_*(\boldsymbol{\theta}, \hat{\boldsymbol{\psi}}) \propto p(\mathbf{d} | \boldsymbol{\theta}, \hat{\boldsymbol{\psi}}) p(\boldsymbol{\theta} | \hat{\boldsymbol{\psi}})\right\}. \quad (7.4.26)$$

Completing these calculations is difficult in general because numerical integration is needed. Explicit solutions can be derived only for linear models with simple parameter structures.

Example 7.8 *Estimation of Hyperparameters with Uniform Priors*
Let the mean of parameter $\boldsymbol{\theta}$ be $E(\boldsymbol{\theta}) = \bar{\boldsymbol{\theta}}$, and the integral in (7.4.25) actually is the likelihood $p(\mathbf{d} \mid \bar{\boldsymbol{\theta}}, \boldsymbol{\psi})$. If $p(\boldsymbol{\psi})$ is uniform and the probability distribution of data is Gaussian, the marginalized MAP (7.4.25) becomes the MLE of $\boldsymbol{\psi}$,

$$\hat{\boldsymbol{\psi}} = \arg\min_{\boldsymbol{\psi}}\left\{\log\left|\mathbf{C}_D(\boldsymbol{\psi})\right| + [\mathbf{d} - \mathbf{u}_D(\bar{\boldsymbol{\theta}})]^T \mathbf{C}_D^{-1}(\boldsymbol{\psi})[\mathbf{d} - \mathbf{u}_D(\bar{\boldsymbol{\theta}})]\right\}. \quad (7.4.27)$$

Then, (7.4.26) gives the MLE of $\boldsymbol{\theta}$,

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}}\left\{[\mathbf{d} - \mathbf{u}_D(\boldsymbol{\theta})]^T \mathbf{C}_D^{-1}(\hat{\boldsymbol{\psi}})[\mathbf{d} - \mathbf{u}_D(\boldsymbol{\theta})]\right\}, \boldsymbol{\theta} \in P_{ad}. \quad (7.4.28)$$

■

Other types of prior distributions can also be incorporated into the marginalized MAP estimation but with more computational effort, for example, by integrating out β and γ from the joint posterior distribution (7.4.22) in Example 7.7. Hierarchical Bayesian has been used in various fields for hyperparameter estimation. Readers may refer to Robert (2007) for a general discussion on this topic and more examples. In groundwater modeling, marginalization-based algorithms were derived for both linear and nonlinear models by Kitanidis (1995). In Woodbury and Rubin (2000), the transport parameters that determine the travel time moments are estimated, while the unknown variances of observation errors are considered as hyperparameters and integrated out. In Woodbury and Ulrych (2000), the log transmissivity field is estimated, while hyperparameters, including the variance of observation error, prior guess, as well as its uncertainty, are integrated out. Recent applications can be found in (Koutsourelakis 2009; Mondal et al. 2010; Chen et al. 2012). More applications of hierarchical Bayesian methods can be found in subsequent chapters. For example, hierarchical Bayesian will be used to estimate the hyperparameters of Gaussian process regression models in Chap. 8.

7.4.3 Geostatistical Inversion

7.4.3.1 Cokriging Estimate

Cokriging, an extension of kriging, can estimate two or more random fields. Thus, not only parameter measurements, but also state measurements can be used for parameter estimation simultaneously. Because the information contained in all related sources is used for estimation, cokriging can often give better results than that of using a single source alone. Theoretical basis of cokriging was developed originally by Matheron (1971), Journel and Huijbregts (1978), and Myers (1982). For detailed discussion on cokriging and its applications, readers may refer to Isaaks and Srivastava (1989), Cressie (1993), Kitanidis (1997), Goovaerts (1997), Deutsch and Journel (1998), Webster and Oliver (2001), Rubin (2003), and Knotters et al. (2010).

Let $f(\mathbf{x})$ and $h(\mathbf{x})$ be two zero-mean random fields. Assume that we know their variance and covariance functions and the measurements of $f(\mathbf{x})$ and $h(\mathbf{x})$ are given by

$$f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_m); h(\mathbf{x}'_1), h(\mathbf{x}'_2), \cdots, h(\mathbf{x}'_n) \tag{7.4.29}$$

in which the measurement locations of $f(\mathbf{x})$ and $h(\mathbf{x})$ may be identical or different. Like in kriging, the cokriging estimate of $f(\mathbf{x})$ at any point $\mathbf{x}_0$ in the field, denoted by $\hat{f}(\mathbf{x}_0)$, is a linear combination of all measurements of both $f(\mathbf{x})$ and $h(\mathbf{x})$,

$$\hat{f}(\mathbf{x}_0) = \sum_{i=1}^{m} \lambda_i(\mathbf{x}_0) f(\mathbf{x}_i) + \sum_{j=1}^{n} \mu_j(\mathbf{x}_0) h(\mathbf{x}'_j), \tag{7.4.30}$$

which is written shortly as

$$\hat{f}_0 = \sum_{i=1}^{m} \lambda_i^0 f_i + \sum_{j=1}^{n} \mu_j^0 h_j,$$

where λ_i^0 ($i = 1, 2, \cdots, m$) and μ_j^0 (j=1, 2, $\cdots$, n) are called cokriging coefficients. They can be determined by the requirement that $\hat{f}_0$ is an unbiased estimate of f_0 and the variance of estimation is the minimum. Because $f(\mathbf{x})$ and $h(\mathbf{x})$ have zero mean, the unbiased requirement is satisfied automatically. The variance of estimation is given by

$$\begin{aligned} Var(\hat{f}_0 - f_0) &= E\left\{\left[\left(\sum_{i=1}^{m} \lambda_i^0 f_i + \sum_{j=1}^{n} \mu_j^0 h_j - f_0\right)^2\right]\right\} \\ &= \sum_{i=1}^{m}\sum_{k=1}^{m} \lambda_i^0 \lambda_k^0 E(f_i f_k) + 2\sum_{i=1}^{m}\sum_{j=1}^{n} \lambda_i^0 \mu_j^0 E(f_i h_j) + \sum_{j=1}^{n}\sum_{l=1}^{n} \mu_j^0 \mu_l^0 E(h_j h_l) \\ &\quad - 2\sum_{i=1}^{m} \lambda_i^0 E(f_i f_0) - 2\sum_{j=1}^{n} \mu_j^0 E(h_j f_0) + \mathrm{var}(f_0), \end{aligned} \tag{7.4.31}$$

where all covariance components $E(\cdot\,\cdot)$ are assumed to be known (we will show how to find them shortly). The necessary condition for minimizing $Var(\hat{f}_0 - f_0)$ gives the following $m + n$ cokriging equations for solving the $m + n$ cokriging coefficients

$$\begin{aligned} &\sum_{k=1}^{m} \lambda_k^0 E(f_i f_k) + \sum_{j=1}^{n} \mu_j^0 E(f_i h_j) = E(f_i f_0),\ i = 1, 2, \cdots, m \\ &\sum_{i=1}^{m} \lambda_i^0 E(f_i h_j) + \sum_{l=1}^{n} \mu_l^0 E(h_j h_l) = E(h_j f_0),\ j = 1, 2, \cdots, n \end{aligned} \tag{7.4.32}$$

And, the minimum variance of estimation is given by

$$Var(\hat{f}_0 - f_0) = Var(\hat{f}_0) - \sum_{i=1}^{m} \lambda_i^0 E(f_i f_0) - \sum_{j=1}^{n} \mu_j^0 E(h_j f_0). \tag{7.4.33}$$

On the right-hand side of the above equation, the first term is the unconditional variance of the random field $f(\mathbf{x})$, the second term is the decrease of the variance (or reduction in uncertainty) due to $f(\mathbf{x})$ measurements only (kriging), and the third term is the decrease of the variance after incorporating measurements of $h(\mathbf{x})$ as conditioning data (cokriging). In parallel to the above derivations for $f(\mathbf{x})$, we can derive the cokriging estimate of $h(\mathbf{x})$ by incorporating $f(\mathbf{x})$ measurements.

As in kriging, the coefficient matrix of the cokriging equations (7.4.32) is independent of the location $\mathbf{x}_0$. Therefore, to calculate the cokriging coefficients for different locations, we only need to change the right-hand terms of the equations.

In the most general case, cokriging only requires that the involved random fields be somehow correlated, not necessarily through a model.

7.4.3.2 The Procedure of Geostatistical Inversion

Cokriging estimate can be used for parameter estimation of a model in the statistical framework by considering the unknown parameter $\theta(\mathbf{x})$ as a random field and the state variable $u(\mathbf{x})$ as another and the two fields are related through a model $u=\mathcal{M}(\theta)$. Assume that we have the following measurements of the two fields:

$$\theta(\mathbf{x}_1), \theta(\mathbf{x}_2), \cdots, \theta(\mathbf{x}_m);\ u(\mathbf{x}'_1), u(\mathbf{x}'_2), \cdots, u(\mathbf{x}'_n). \tag{7.4.34}$$

In the above equation, $X_\theta = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ are measurement locations and/or times of the unknown parameter $\theta(\mathbf{x})$, and $X_u = \{\mathbf{x}'_1, \mathbf{x}'_2, \cdots, \mathbf{x}'_n\}$ are observation locations and/or times of the state variable $u(\mathbf{x})$. The locations/times of parameter measurements and state observations may be identical or different.

According to Sect. 6.2.4, a random field can be decomposed into two parts: a deterministic trend (or mean) term for describing the large-scale variability of the field and a random fluctuation term for describing the small-scale variability of the field. Therefore, $\theta(\mathbf{x})$ and $u(\mathbf{x})$ can be decomposed into

$$\theta(\mathbf{x}) = \overline{\theta}(\mathbf{x}) + f(\mathbf{x}), \quad u(\mathbf{x}) = \overline{u}(\mathbf{x}) + h(\mathbf{x}) \tag{7.4.35}$$

where $\overline{\theta}(\mathbf{x})$ and $\overline{u}(\mathbf{x})$ are deterministic trends of $\theta(\mathbf{x})$ and $u(\mathbf{x})$, respectively, and $f(\mathbf{x})$ and $h(\mathbf{x})$ are their random fluctuations (with zero mean). Geostatistical inversion uses the following steps to obtain an estimate of $\theta(\mathbf{x})$:

- Estimate deterministic trend $\overline{\theta}(\mathbf{x})$ and let $\overline{u} = \mathcal{M}(\overline{\theta})$.
- Filter the trends out from the measurement data in (7.4.34) to obtain “measurements” of perturbations $f(\mathbf{x})$ and $h(\mathbf{x})$.
- Use cokriging to obtain an estimate of $f(\mathbf{x})$.

The remaining problems are as follows: how to estimate $\overline{\theta}(\mathbf{x})$ and how to find covariance components needed in (7.4.32) for cokriging estimate. As a function, $\overline{\theta}(\mathbf{x})$ must be parameterized before it is estimated, for example, by zonation, interpolation, or functional approximation learned in Sect. 6.1. The estimation of $\overline{\theta}(\mathbf{x})$ then becomes estimation of a low-dimensional vector $\boldsymbol{\beta}$. For parameterizing the random field $f(\mathbf{x})$, a covariance model of $\theta(\mathbf{x})$, denoted by $Cov_{\theta\theta}(\boldsymbol{\psi})$, must be chosen, in which $\boldsymbol{\psi}$ represents some statistical parameters that need to be estimated (see also Sect. 6.2). Let us give a simple example.

Example 7.9 *Estimate of a Second-Order Stationary Random Field*

A Gaussian random field can be characterized by its first two moments, the mean and covariance. Even so, the number of candidate random fields for inversion is still too large to select. A second-order stationary random field defined in Sect. 6.2.1 is a simple case, for which the mean function μ is constant and the covariance function depends only on the separation vector $\mathbf{r} = \mathbf{x}_2 - \mathbf{x}_1$. For example, we can select the isotropic exponential covariance model

$$C_{\theta\theta}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_\theta^2 \exp(-r / l), \; r = \left\| \mathbf{x}_i - \mathbf{x}_j \right\| \tag{7.4.36}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are two arbitrary points, σ_θ^2 is the variance, and l is the correlation length. In this case, the random field is determined by only three scalar parameters, $\{\mu, \sigma_\theta^2, l\}$. With kriging, these parameters are estimated only by parameter measurements. Now, we want to improve their estimates with the information provided by state observations.

7.4.3.3 Correlations Between Parameter Measurements and State Observations

Our first task is to find the covariance matrices between parameter measurements and state observations so that the information embedded in them can be combined for inversion. Let $\boldsymbol{\theta}_D$ be true parameter values measured at (X_θ) and $\mathbf{u}_D$ be the model outputs measured at (X_u) as defined in (7.4.34),

$$\boldsymbol{\theta}_D = [\theta(\mathbf{x}_1), \theta(\mathbf{x}_2), \cdots, \theta(\mathbf{x}_m)]^T, \quad \mathbf{u}_D = [u(\mathbf{x}_1'), u(\mathbf{x}_2'), \cdots, u(\mathbf{x}_m')]^T. \tag{7.4.37}$$

The covariance matrices that we need for geostatistical inversion are defined by

$$\begin{aligned} \mathbf{C}_{D,\boldsymbol{\theta\theta}} &= E[(\boldsymbol{\theta}_D - \overline{\boldsymbol{\theta}}_D)(\boldsymbol{\theta}_D - \overline{\boldsymbol{\theta}}_D)^T] = E[\mathbf{f}_D \mathbf{f}_D^T] \\ \mathbf{C}_{D,\boldsymbol{\theta}\mathbf{u}} &= E[(\mathbf{u}_D - \overline{\mathbf{u}}_D)(\boldsymbol{\theta}_D - \overline{\boldsymbol{\theta}}_D)^T] = E[\mathbf{h}_D \mathbf{f}_D^T] \\ \mathbf{C}_{D,\mathbf{uu}} &= E[(\mathbf{u}_D - \overline{\mathbf{u}}_D)(\mathbf{u}_D - \overline{\mathbf{u}}_D)^T] = E[\mathbf{h}_D \mathbf{h}_D^T]. \end{aligned} \tag{7.4.38}$$

The first one, $\mathbf{C}_{D,\boldsymbol{\theta\theta}}$, can be obtained directly from the chosen covariance model of $\theta(\mathbf{x})$. The other two, $\mathbf{C}_{D,\boldsymbol{\theta}\mathbf{u}}$ and $\mathbf{C}_{D,\mathbf{uu}}$, can be found after we find $\overline{u}(\mathbf{x})$ from $\overline{\theta}(\mathbf{x})$ and find $h(\mathbf{x})$ from $f(\mathbf{x})$ as shown in the two methods described below.

Perturbation Equation and Adjoint-State Method Under certain assumptions, the model equation $\mathcal{L}(\overline{\theta} + f, \overline{u} + h, \mathbf{x}) = 0$ can be separated into two, one for mean and the other for fluctuation (or perturbation), viz.

$$\overline{\mathcal{L}}(\overline{\theta}, \overline{u}, \mathbf{x}) = 0 \quad \text{and} \quad \widetilde{\mathcal{L}}(f, h, \mathbf{x}) = 0. \tag{7.4.39}$$

Example 7.10 *Perturbation Equation of Steady-State Groundwater Flow*
Two-dimensional steady-state groundwater flow in a confined aquifer is governed by

$$\frac{\partial}{\partial x}\left(T\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(T\frac{\partial u}{\partial y}\right)+Q=0,(x,y)\equiv\mathbf{x}\in(\Omega), \tag{7.4.40}$$

where $u(\mathbf{x})$ is hydraulic head, S is storage coefficient, $T(\mathbf{x})$ is aquifer transmissivity to be estimated, m is the aquifer thickness, Q is a sink/source term, and Ω is the flow region. The equation is subjected to appropriate initial and boundary conditions. Assume that all parameters and variables in the governing equation and initial and boundary conditions are known, except transmissivity $T(\mathbf{x})$. As usual, we will use $\theta=\log T$ as the estimated random field and assume that it is normally distributed. Substituting (7.4.35) into (7.4.40) gives

$$\begin{aligned}&\frac{\partial^2\overline{u}}{\partial x^2}+\frac{\partial^2 h}{\partial x^2}+\left(\frac{\partial\overline{\theta}}{\partial x}+\frac{\partial f}{\partial x}\right)\left(\frac{\partial\overline{u}}{\partial x}+\frac{\partial h}{\partial x}\right)+\frac{\partial^2\overline{u}}{\partial y^2}+\frac{\partial^2 h}{\partial y^2}\\&+\left(\frac{\partial\overline{\theta}}{\partial y}+\frac{\partial f}{\partial y}\right)\left(\frac{\partial\overline{u}}{\partial y}+\frac{\partial h}{\partial y}\right)+e^{-\overline{\theta}-f}Q=0.\end{aligned} \tag{7.4.41}$$

To proceed further, let us make the following simplifications: (i) The random field $\theta(\mathbf{x})$ is second-order stationary, and (ii) the second-order terms $(\partial f/\partial x)(\partial h/\partial x)$ and $(\partial f/\partial y)(\partial h/\partial y)$ can be ignored. With these assumptions, (7.4.41) is simplified to

$$\frac{\partial^2\overline{u}}{\partial x^2}+\frac{\partial^2 h}{\partial x^2}+\frac{\partial^2\overline{u}}{\partial y^2}+\frac{\partial^2 h}{\partial y^2}+\frac{\partial f}{\partial x}\frac{\partial\overline{u}}{\partial x}+\frac{\partial f}{\partial y}\frac{\partial\overline{u}}{\partial y}+Qe^{-\overline{\theta}}(1-f)=0 \tag{7.4.42}$$

where e^{-f} is approximated by $(1-f)$. Taking expectation of this equation, we obtain the mean equation

$$\overline{\mathcal{L}}(\overline{\theta},\overline{u},\mathbf{x})\equiv\frac{\partial^2\overline{u}}{\partial x^2}+\frac{\partial^2\overline{u}}{\partial y^2}+e^{-\overline{\theta}}Q=0. \tag{7.4.43}$$

Subtracting (7.4.43) from (7.4.42), we obtain the perturbation equation

$$\tilde{\mathcal{L}}(f,h,\mathbf{x})\equiv\frac{\partial^2 h}{\partial x^2}+\frac{\partial^2 h}{\partial y^2}+\frac{\partial f}{\partial x}\frac{\partial\overline{u}}{\partial x}+\frac{\partial f}{\partial y}\frac{\partial\overline{u}}{\partial y}-fe^{-\overline{\theta}}Q=0. \tag{7.4.44}$$

The mean equation (7.4.43) is subjected to the original boundary conditions, and the perturbation equation (7.4.44) is subjected to zero-head and/or no-flow boundary conditions. The perturbation equation (7.4.44) was first used by Hoeksema and Kitanidis (1984) for geostatistical inversion.

Applying the adjoint-state method in Sect. 5.3.1, we can form the adjoint $\psi(\mathbf{x})$ of $h(\mathbf{x})$. Then, the sensitivity coefficient of any $h_j=h(\mathbf{x}'_j)$ with respect to any f_i

(where $\mathbf{x}'_j$ is an observation location in X_u and f_i is the nodal value of $f(\mathbf{x})$ at node i) can be calculated by

$$\frac{\partial h_j}{\partial f_i} = \int_{(\Omega_i)} e^{\overline{\theta}} \left[\frac{\partial \psi}{\partial x}\frac{\partial \overline{u}}{\partial x} + \frac{\partial \psi}{\partial y}\frac{\partial \overline{u}}{\partial y} \right] d\Omega \approx \alpha_i \Omega_i \tag{7.4.45}$$
$$(i = 1, 2, \cdots, N;\ j = 1, 2, \cdots, n).$$

In the above equation, (Ω_i) is the exclusive subdomain of the node, Ω_i is its area, and α_i is the nodal value of the integrand at the node. All sensitivity coefficients in (7.4.45) comprise an $n \times N$ Jacobian matrix $\mathbf{J}_D(\overline{\boldsymbol{\theta}}) = [\partial \mathbf{h}_D / \partial \mathbf{f}]$, and the number of model runs needed for calculating these sensitivity coefficients is ($n+1$). Detailed description and numerical results for this example and also for transient flow problems can be found in Sun and Yeh (1992). ■

After having Jacobian $\mathbf{J}_D$, the first-order approximation $\mathbf{h}_D \approx \mathbf{J}_D\mathbf{f}$ can be used to calculate the required covariance matrices in (7.4.38) as follows:

$$\begin{aligned}
\mathbf{C}_{D,\boldsymbol{\theta\theta}} &= E[\mathbf{f}_D\mathbf{f}_D^T] \\
\mathbf{C}_{D,\boldsymbol{\theta}\mathbf{u}} &= E[\mathbf{h}_D\mathbf{f}_D^T] = \mathbf{J}_D E[\mathbf{f}\mathbf{f}_D^T] \\
\mathbf{C}_{D,\mathbf{uu}} &= E[\mathbf{h}_D\mathbf{h}_D^T] = \mathbf{J}_D E[\mathbf{f}\mathbf{f}^T]\mathbf{J}_D^T.
\end{aligned} \tag{7.4.46}$$

Perturbation Expansion and Moment Equation Method Although the two assumptions used in Example 7.10 have significantly simplified the task of calculating the covariance matrices, they make the use of the method limited and the results inaccurate. The *perturbation expansion–moment equation* method, however, does not depend on these assumptions, namely, it can handle non-stationary random fields and retain high-order variations in the perturbation equation and thus make the calculation of covariance more general and accurate.

Let us consider again the two-dimensional steady-state model in (7.4.40) but written with the summation convention,

$$\frac{\partial^2 u}{\partial x_i^2} + \frac{\partial \theta}{\partial x_i}\frac{\partial u}{\partial x_i} = -Qe^{-\theta}, i = 1, 2 \tag{7.4.47}$$

where $x_1 = x$ and $x_2 = y$. The perturbation expansion of $u(\mathbf{x})$ is defined by

$$u = u^{(0)} + u^{(1)} + u^{(2)} + \cdots \tag{7.4.48}$$

where $u^{(k)}$ is the *k*-order term proportional to $(\sigma_\theta)^k$ in the statistical sense and σ_θ is the standard deviation of θ. Substituting $\theta = \overline{\theta} + f$ defined in (7.4.35) and the above expansion into equation (7.4.47), we have

$$\frac{\partial^2}{\partial x_i^2}\left[u^{(0)}+u^{(1)}+u^{(2)}+\cdots\right]+\frac{\partial}{\partial x_i}\left[\bar{\theta}+f\right]\frac{\partial}{\partial x_i}\left[u^{(0)}+u^{(1)}+u^{(2)}+\cdots\right]$$
$$=-Qe^{-\bar{\theta}}\left[1-f+\frac{1}{2}f^2-\cdots\right]$$

This equation can be decomposed into a series of equations by collecting the terms having the same order:

$$\begin{aligned}
&\text{Zeroth-order: } \frac{\partial^2 u^{(0)}}{\partial x_i^2}+\frac{\partial\bar{\theta}}{\partial x_i}\frac{\partial u^{(0)}}{\partial x_i}=-Qe^{-\bar{\theta}},\\
&\text{First-order: } \frac{\partial^2 u^{(1)}}{\partial x_i^2}+\frac{\partial\bar{\theta}}{\partial x_i}\frac{\partial u^{(1)}}{\partial x_i}=-\frac{\partial f}{\partial x_i}\frac{\partial u^{(0)}}{\partial x_i}+Qe^{-\bar{\theta}}f,\\
&\text{Second-order: } \frac{\partial^2 u^{(2)}}{\partial x_i^2}+\frac{\partial\bar{\theta}}{\partial x_i}\frac{\partial u^{(2)}}{\partial x_i}=-\frac{\partial f}{\partial x_i}\frac{\partial u^{(1)}}{\partial x_i}-Qe^{-\bar{\theta}}f^2,\\
&\qquad\cdots\cdots\cdots
\end{aligned}\tag{7.4.49}$$

where summation over index i is implied in the above equations. The zeroth-order equation is the mean equation as (7.4.43) but without invoking the secondary stationarity assumption. It is subject to the original boundary conditions. The perturbation equation is decomposed into a series of equations without ignoring high-order terms, and all of these equations are subject to zero boundary conditions. By solving the equations in (7.4.49) one by one, all $u^{(0)}, u^{(1)}, u^{(2)}, \cdots$ in (7.4.48) can be obtained successively. As a result, u can approximated to any order. For example, to the second order, we have $u \approx u^{(0)}+u^{(1)}+u^{(2)}$. By taking expectation, we have $E[u^{(0)}]=u^{(0)}$ from the zeroth-order equation, $E[u^{(1)}]=0$ and $h=u-\bar{u}\approx u^{(1)}$ from the first-order equation, and then, we can find the second-order variations of u characterized by covariance $C_{uu}(\mathbf{x},\mathbf{x}')=E[h(\mathbf{x})h(\mathbf{x}')]$ and cross-covariance $C_{\theta u}(\mathbf{x},\mathbf{x}')=E[f(\mathbf{x})h(\mathbf{x}')]$.

By multiplying the first-order equation in (7.4.49) with $u^{(1)}$ at another location $\mathbf{x}_0$ and taking expectation, we obtain the PDE for solving $C_{uu}(\mathbf{x},\mathbf{x}')$:

$$\begin{aligned}
&\frac{\partial^2 C_{uu}(\mathbf{x},\mathbf{x}')}{\partial x_i^2}+\frac{\partial\bar{\theta}}{\partial x_i}\frac{\partial C_{uu}(\mathbf{x},\mathbf{x}')}{\partial x_i}\\
&=-\frac{\partial u^{(0)}(\mathbf{x})}{\partial x_i}\frac{\partial C_{\theta u}(\mathbf{x},\mathbf{x}')}{\partial x_i}+Qe^{-\bar{\theta}}C_{\theta u}(\mathbf{x},\mathbf{x}')
\end{aligned}\tag{7.4.50}$$

By rewriting the first-order equation in (7.4.49) in terms of $\mathbf{x}'$, multiplying it with $f(\mathbf{x})$, and taking expectation, we obtain the PDE for solving $C_{\theta u}(\mathbf{x},\mathbf{x}')$:

$$\begin{aligned}&\frac{\partial^2 C_{\theta u}(\mathbf{x},\mathbf{x}')}{\partial x_i'^2}+\frac{\partial \bar{\theta}}{\partial x_i'}\frac{\partial C_{\theta u}(\mathbf{x},\mathbf{x}')}{\partial x_i'}\\&=-\frac{\partial u^{(0)}(\mathbf{x}')}{\partial x_i'}\frac{\partial C_{\theta\theta}(\mathbf{x},\mathbf{x}')}{\partial x_i'}+Qe^{-\bar{\theta}}C_{\theta\theta}(\mathbf{x},\mathbf{x}')\end{aligned} \tag{7.4.51}$$

The above PDEs for $C_{\theta u}(\mathbf{x},\mathbf{x}')$ and $C_{uu}(\mathbf{x},\mathbf{x}')$ have the same structures as the first-order equation, subject to the same zero boundary conditions and thus can be solved by the same code. First, we use the known $C_{\theta\theta}(\mathbf{x},\mathbf{x}')$ to obtain $C_{\theta u}(\mathbf{x},\mathbf{x}')$ by solving (7.4.51) and then use the latter to obtain $C_{uu}(\mathbf{x},\mathbf{x}')$ by solving (7.4.50). Detailed discussions on theory, algorithms, and applications of this method can be found in Zhang (2002) and Rubin (2003).

By systematically changing the pair $(\mathbf{x},\mathbf{x}')$ to the specified parameter measurement locations in X_θ and state observation locations in X_u defined in (7.4.34), all variance matrices in (7.4.38) can be calculated and used for geostatistical inversion.

7.4.3.4 Finding the Inverse Solution

Let us concatenate all data in (7.4.34) into a vector form $\mathbf{z}_D^{obs}$ and denote the corresponding model-simulated values by $\mathbf{z}_D$, namely

$$\begin{aligned}&\mathbf{z}_D^{obs}=\left\{z_{D,1}^{obs},z_{D,2}^{obs},\cdots,z_{D,m}^{obs},z_{D,m+1}^{obs},\cdots,z_{D,m+n}^{obs}\right\},\\&\text{where}\\&\left\{z_{D,i}^{obs}=\theta(\mathbf{x}_i)+\varepsilon_i \mid i=1,2,\cdots,m\right\},\\&\left\{z_{D,m+j}^{obs}=u(\mathbf{x}_j')+\eta_j \mid j=1,2,\cdots,n\right\}\\&\text{and}\\&\mathbf{z}_D(\boldsymbol{\beta},\boldsymbol{\psi})=\left\{\hat{\theta}(\mathbf{x}_1),\hat{\theta}(\mathbf{x}_2),\cdots,\hat{\theta}(\mathbf{x}_m),\hat{u}(\mathbf{x}_1'),\hat{u}(\mathbf{x}_2'),\cdots,\hat{u}(\mathbf{x}_n')\right\}\end{aligned} \tag{7.4.52}$$

where $\hat{\theta}(\mathbf{x},\boldsymbol{\beta})$ is the current parameter estimate and $\hat{u}(\mathbf{x},\boldsymbol{\beta},\boldsymbol{\psi})$ is the current state estimate. Geostatistical inversion consists of two stages: First, the combined data ($\mathbf{z}_D^{obs}$) in (7.4.52) are used to estimate $\{\boldsymbol{\beta},\boldsymbol{\psi}\}$, and second, the same data are used to determine a realization of the field as the inverse solution.

In order to use all data $\mathbf{z}_D^{obs}$ for parameter estimation, after we obtain the covariance matrices in (7.4.46), we can combine them into an $(m+n)\times(m+n)$ block covariance matrix

$$\mathbf{C}_D(\boldsymbol{\psi})=\begin{bmatrix}\mathbf{C}_{D,\theta\theta} & \mathbf{C}_{D,\theta u}\\ \mathbf{C}_{D,\theta u} & \mathbf{C}_{D,uu}\end{bmatrix}. \tag{7.4.53}$$

A method given in Sect. 7.4.2 for statistical parameter inversion can be used to estimate $\{\boldsymbol{\beta}, \boldsymbol{\psi}\}$. When a uniform prior is used, we have the following MLE estimates according to (7.4.27) and (7.4.28):

$$\hat{\boldsymbol{\psi}} = \arg\min_{\boldsymbol{\psi}} \left\{ \log \left| \mathbf{C}_D(\boldsymbol{\psi}) \right| + [\mathbf{z}_D^{obs} - \mathbf{z}_D(\hat{\boldsymbol{\beta}}, \boldsymbol{\psi})]^T \mathbf{C}_D^{-1}(\boldsymbol{\psi}) [\mathbf{z}_D^{obs} - \mathbf{z}_D(\hat{\boldsymbol{\beta}}, \boldsymbol{\psi})] \right\}, \quad (7.4.54)$$

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ [\mathbf{z}_D^{obs} - \mathbf{z}_D(\boldsymbol{\beta}, \hat{\boldsymbol{\psi}})]^T \mathbf{C}_D^{-1}(\hat{\boldsymbol{\psi}}) [\mathbf{z}_D^{obs} - \mathbf{z}_D(\boldsymbol{\beta}, \hat{\boldsymbol{\psi}})] \right\}. \quad (7.4.55)$$

When a Gaussian prior is given, the conjugate Gaussian estimator in Sect. 4.2.3 can be used, instead of the above two equations.

After $\{\boldsymbol{\beta}, \boldsymbol{\psi}\}$ are estimated, we have the estimates of $\overline{\theta}(\mathbf{x})$ and $\overline{u}(\mathbf{x})$, and the final task is to find a realization of $\theta(\mathbf{x})$. Now, we have all required data for cokriging: two zero-mean random fields $f(\mathbf{x}) = \theta(\mathbf{x}) - \overline{\theta}(\mathbf{x})$ and $h(\mathbf{x}) = u(\mathbf{x}) - \overline{u}(\mathbf{x})$, their covariance matrices (7.4.53), and their measurements (7.4.29) obtained by filtering out the means from (7.4.34). Using the cokriging estimate (7.4.30) and the variance of estimation (7.4.33) and writing them in terms of $\theta(\mathbf{x})$, then at any unmeasured location $\mathbf{x}$, we have

$$\hat{\theta}(\mathbf{x}) = \overline{\theta}(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i(\mathbf{x})[z_{D,i}^{obs} - \overline{\theta}(\mathbf{x}_i)] + \sum_{j=1}^{n} \mu_j(\mathbf{x})[z_{D,m+j}^{obs} - \overline{u}(\mathbf{x}'_j)], \quad (7.4.56)$$

$$Var[\hat{\theta}(\mathbf{x}) - \theta(\mathbf{x})] = \sigma_\theta^2 - \sum_{i=1}^{m} \lambda_i(\mathbf{x}) C_{D,\theta\theta}(\mathbf{x}, \mathbf{x}_i) - \sum_{j=1}^{n} \mu_j(\mathbf{x}) C_{D,\theta u}(\mathbf{x}, \mathbf{x}'_j). \quad (7.4.57)$$

For a transient problem, the state variable and observation data depend on time. The covariance matrix $\mathbf{C}_D$ should contain the covariance of data between different times. When there are k time periods, the number of state observations will be $k \times n$. In this case, (7.4.56) and (7.4.57) become

$$\begin{aligned} \hat{\theta}(\mathbf{x}) = \overline{\theta}(\mathbf{x}) &+ \sum_{i=1}^{m} \lambda_i(\mathbf{x})[z_{D,i}^{obs} - \overline{\theta}(\mathbf{x}_i)] \\ &+ \sum_{s=1}^{k} \sum_{j=1}^{n} \mu_{j,s}(\mathbf{x})[z_{D,m+js}^{obs} - \overline{u}(\mathbf{x}'_j, t_s)], \end{aligned} \quad (7.4.58)$$

$$\begin{aligned} Var[\hat{\theta}(\mathbf{x}) - \theta(\mathbf{x})] = \sigma_\theta^2 &- \sum_{i=1}^{m} \lambda_i(\mathbf{x}) C_{D,\theta\theta}(\mathbf{x}, \mathbf{x}_i) \\ &- \sum_{s=1}^{k} \sum_{j=1}^{n} \mu_{j,s}(\mathbf{x}) C_{D,\theta u}(\mathbf{x}, \mathbf{x}'_j, t_s). \end{aligned} \quad (7.4.59)$$

Equation (7.4.59) shows that transient observation data contain more information content to reduce the uncertainty of estimation. For details, see Sun and Yeh (1992). Note that the cross-covariance $C_{D,\theta u}$ in (7.4.59) involves both spatial and temporal lags. The space–time covariance models described under Sect. 6.2.3 can be used to fit the data. Cokriging has found wide applications not only in hydrogeology and petroleum engineering (Kitanidis 1995; Rubin 2003; Oliver and Chen 2011), but also in remote sensing and natural resources modeling (Goovaerts 2000; Zhou et al. 2013; Michalak et al. 2004; Krajewski 1987; Smith and Kummerow 2013).

7.4.3.5 Geostatistical Inversion for Multistate Models

In multistate model inversion, we have k state variables, $\{u_1(\boldsymbol{\theta}), u_2(\boldsymbol{\theta}), \cdots, u_k(\boldsymbol{\theta})\}$, and k sets of observation data, $\{D_1, D_2, \cdots, D_k\}$, where $D_l = u_l^{obs}$ ($l = 1, 2, \cdots, k$). Using these data, the unknown parameter can be estimated either in the deterministic framework as a multiobjective optimization problem (Sect. 3.4.2) or in the statistical framework as a Bayesian inference problem (Sect. 4.2.5). The extension of the geostatistical inversion process from single state variable to multistates is straightforward. The key is to find all cross-covariance matrices between measurements of the unknown parameter and observations of state variables. When the coupled forward model containing these state variables is given, we can use the adjoint-state method for multistate models (Sect. 5.3.3) to calculate these cross-covariance matrices. A typical example is the estimation of hydraulic conductivity using both head and concentration observations. Once the combined covariance matrix $\mathbf{C}_D(\boldsymbol{\psi})$ is formed, all unknown statistical parameters can be estimated by MLE. We will not repeat the details of this process here. Finally, the cokriging estimate with multistate data is given by

$$\hat{\theta}(\mathbf{x}) = \sum_{i=1}^{m} \lambda_i(\mathbf{x})\theta^f(\mathbf{x}_i) + \sum_{l=1}^{k}\sum_{j=1}^{n_l} \mu_{l,j}(\mathbf{x}) u_l^f(\mathbf{x}'_{l,j}), \tag{7.4.60}$$

where n_l is the number of observations in D_l, $\mathbf{x}_{l,j}$ are the locations of state observations, and $\mu_{l,j}$ are the cokriging coefficients associated with state u_l for $l = 1, 2, \cdots, k$ and $j = 1, 2, \cdots, n_l$.

The applications of multistate geostatistical inversion are numerous. For example, observations on water level (or pressure), concentration, temperature, arrival time, well logs, water age, head slope, electrical resistivity, and seismic geophysical data have been used to estimate hydraulic conductivity (Kowalsky et al. 2004; Yeh et al. 2002; Sun et al. 1995; Rubin and Dagan 1992; Fienen et al. 2009; Harvey and Gorelick 1995; Pollock and Cirpka 2012).

7.4.3.6 Other Methods for Combined Parameter and State Estimation

Using both parameter and state measurements for parameter estimation is a central issue of inverse problem study. In fact, it has been studied in the previous chapters in different forms. In the deterministic framework, parameter measurements are used to generate constraints or penalty terms for stabilizing the inverse solution (Using them as hard constraints is not recommended because of the existence of model error). In the statistical framework, parameter measurements are used to generate prior PDF and this information is then transferred into the posterior PDF with Bayesian inference. Identifying model structure parameters and decreasing model error allow us to use parameter measurements more directly. The problem considered in this section is a joint estimation problem, in which parameter measurements and state observations are combined to provide information for inversion. The geostatistical inversion method described above is not the only method for solving such a joint estimation problem. The pilot-point method to be introduced in the next subsection is another interesting approach of using parameter measurements directly for inversion. In Chap. 9, we will introduce the less computationally intensive, sequential data assimilation methods that can perform joint parameter and state estimates when new measurements become available.

7.4.4 The Pilot-Point Method

7.4.4.1 Pilot Points as Shape Parameters

Geostatistical inversion with cokriging can provide a distributed estimation to the unknown distributed parameter. But, there are problems associated with the approach. The first problem is structure error: The estimated parameter is regarded as a realization of a random field parameterized by a selected statistical structure with only a few statistical parameters. Although the estimated parameter is distributed, the number of its DOF is actually low compared to that of the true parameter. As a result, the fitting residual of the estimated parameter generally could not be reduced to a satisfactory level. The second problem is the error in statistical parameters: MLE may give erroneous results because of the insensitivity of observations to these parameters. The third problem is plausibility: There are infinite realizations of the estimated random field that have the same statistical structure and can produce nearly the same data. The realization resulted from kriging or cokriging interpolation is merely a smoothed, or even an oversmoothed one. Unfortunately, errors caused by these problems are not accounted for in the variance of estimation.

At the end of Sect. 6.3.3, we mentioned three cases of using kriging as parameterization for inversion, where kriging is represented by

$$\hat{\theta}(\mathbf{x}) = \sum_{i=1}^{m} \theta(\mathbf{x}_i)\lambda_i(\mathbf{x}, \boldsymbol{\nu}, \boldsymbol{\psi}), \mathbf{x} \in (\Omega). \tag{7.4.61}$$

When state observations are available, $\boldsymbol{\theta} = \left\{\theta(\mathbf{x}_i) \mid i = 1, 2, \cdots, \mathrm{m}\right\}$ are considered as "hypothetical measurements" to be estimated by inversion. The basis points of interpolation, $\boldsymbol{\nu} = \left\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\right\}$, now become the shape parameters of parameterization. After $\boldsymbol{\theta}$ is estimated, a distributed estimate of the unknown parameter $\theta(\mathbf{x})$ can be obtained by kriging or cokriging. Introducing shape parameters $\boldsymbol{\nu}$ increases the DOF of parameterization, making the shape of parameter structure to be adjustable and, thus, decreasing the structure error and fitting residual. This method, known as the *pilot-point method*, was originally introduced by de Marsily et al. (1984), and the shape parameters $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ are called *pilot points*.

As mentioned in Sect. 6.2.1, there are two approaches to deal with real measurements, $\theta_{m+1}, \theta_{m+2}, \cdots, \theta_{m+k}$, of the unknown parameter. The first approach, which is often used in the literature, considers the real measurements as hard data (i.e., values of the estimated parameter at measurement locations are fixed and equal to the measured values during inversion). This approach can make the inverse solution more stable, but at the expense of producing biased estimates because of the model structure error. In the second approach, the parameter values at measurement locations are estimated together with the parameter values at the pilot points, while the real measurements are used as prior information to stabilize the inverse solution.

To formulate the pilot-point method, we can use a regularized or generalized least squares objective function for inversion as we have done before in Chap. 4, but now with additional hyperparameters to be estimated

$$\begin{aligned} S(\boldsymbol{\theta}, \boldsymbol{\nu}, \boldsymbol{\psi}, \lambda) = {} & [\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta})]^T \mathbf{C}_D^{-1} [\mathbf{u}_D^{obs} - \mathbf{u}_D(\boldsymbol{\theta})] \\ & + \lambda [\boldsymbol{\theta} - \boldsymbol{\theta}_0]^T \mathbf{C}_P^{-1} [\boldsymbol{\theta} - \boldsymbol{\theta}_0] \end{aligned} \tag{7.4.62}$$

where $\boldsymbol{\theta} = \{\theta_1, \theta_2, \cdots, \theta_m, \theta_{m+1}, \cdots, \theta_{m+k}\}$ is the unknown parameter vector that consists of the parameter values at m pilot points and k measurement points; the locations of pilot points, $\boldsymbol{\nu} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$, are shape parameters; $\boldsymbol{\psi}$ is a vector of hyperparameters; and λ is a regularization coefficient.

7.4.4.2 Algorithms

After the pioneering work of de Marsily et al. (1984) and Ahmed and de Marsily (1987), numerous variants of pilot-point algorithms have been introduced (LaVenue et al. 1995; Alcolea et al. 2008; Doherty 2003; LaVenue and Pickens 1992; RamaRao et al. 1995; Le Ravalec-Dupin 2010; Alcolea et al. 2006; Singh et al. 2008).

Like the Voronoi zonation method in the deterministic framework, pilot-point method is essentially a method of model structure parameterization and model structure identification in the geostatistical framework. The number of pilot points determines the complexity of model structure and their locations determine the structure pattern. Therefore, the basic concepts and methods of adaptive parameterization can also be used to solve the pilot-point problem formulated in (7.4.62). In fact, by combining different methods for structure parameter identification with methods for statistical parameter estimation described in this chapter, different pilot-point algorithms can be developed. A modified pilot-point algorithm, which is based on the idea of adaptive parameterization and the minimization of objective function (7.4.62), consists of the following major steps:

1. Use the k real measurements to identify a covariance (or variogram) model with estimated statistical parameters $\boldsymbol{\psi}_0$, and then, use kriging to find a distributed estimation $\theta_0(\mathbf{x})$ for all nodes (Sect. 6.3.3).

2. Use the adjoint-state method to calculate the gradient $\partial S / \partial \theta$ for all nodes evaluated at $\theta_0(\mathbf{x})$. Then, m nodes are chosen as the pilot points (shape parameter $\mathbf{v}$) according to the criteria: (i) The selected nodes have larger gradient values than other locations and (ii) their pairwise distances are no smaller than the correlation length (a statistical parameter in $\boldsymbol{\psi}_0$). The components of $\boldsymbol{\theta}_0$ are values of $\theta_0(\mathbf{x})$ at m pilot points and k measurement locations.
3. Minimize the objective function (7.4.62) with the fixed $\mathbf{v}$ obtained in Step 2 and $\boldsymbol{\psi} = \boldsymbol{\psi}_0$ obtained in Step 1 to find the optimal solution $\hat{\boldsymbol{\theta}}$ together with the optimal regularization coefficient λ (Sect. 3.3).
4. As in Step1, but use $\hat{\boldsymbol{\theta}}$ as $m + k$ "measurements" to estimate the covariance model and update the statistical parameters to $\hat{\boldsymbol{\psi}}$, and vestimate $\hat{\theta}(\mathbf{x})$. Now, we can check the stopping criteria presented in Sect. 7.2.2 to determine whether we should (i) stop the iteration, or (ii) continue the iteration (i.e., return to Step 2 by setting $\hat{\theta}(\mathbf{x}) \to \theta_0(\mathbf{x})$, $\hat{\boldsymbol{\theta}} \to \boldsymbol{\theta}_0$, and $\hat{\boldsymbol{\psi}} \to \boldsymbol{\psi}_0$), or (iii) increase the complexity of model structure (i.e., add new pilot points, and then, continue the iteration).

7.4.4.3 Variations and Applications

There are several variations of the above basic pilot-point algorithm that may require additional computational effort

- Use different methods to determine the number and locations of pilot points
- Use parameter measurements as hard data
- Use different regularization functions in (7.4.62)
- Use cokriging instead of using kriging in Step 1 and Step 4
- Use adaptive parameterization to optimize the locations of pilot points (i.e., identifying $(\mathbf{v}, \boldsymbol{\theta})$ simultaneously by solving EIP (Sect. 7.2))
- Use hierarchical Bayesian to estimate $(\boldsymbol{\theta}, \boldsymbol{\psi}, \lambda)$ simultaneously.

These variants have not been fully studied and compared. A recent review on various inverse methods, including the pilot-point method, is given by Hendricks Franssen et al. (2009). A major problem to be considered in the references is how to determine the number of pilot points and how to optimize their locations. The method presented by LaVenue and Pickens (1992) and used by RamaRao et al. (1995) is based on adjoint sensitivity analysis. Wen et al. (1999) found that two to three master points per correlation range in each direction are sufficient for most application. Doherty (2003) used a local homogeneous regularization function to stabilize the inverse solution when the number of pilot points is large. Alcolea et al. (2006) gave detailed steps to the regularized pilot-point algorithm and shows the importance of finding an appropriate regularization coefficient. Tonkin and Doherty (2005) and Moore and Doherty (2005) suggested to use a large number of densely distributed pilot points (a high-dimensional parameterization) to better characterize the structure of heterogeneous aquifers. In order to avoid the overparameterization problem, after the model is linearized, the PCA (see Sect. 6.3.1) is used to reduce the DOF of the pilot-point parameterization (reparameterization). As we have denoted in Sect. 6.3.1, the unknown parameters associated with reparameterization have no physical meanings and the linearization error may be amplified after dimension reduction. The variance of nonlinear prediction error associated with the high-dimensional parameterization was estimated in Tokin et al. (2005). Riva et al. (2010) analyzed the influence of the number of pilot points in geostatistical inversion. Jung et al. (2011) simply chose the number of pilot points to be less than the number of observation data; the problem of "what is the optimal number of pilot points" was not addressed.

The aforementioned pilot-point-based, adaptive parameterization algorithm already includes determination of the locations of pilot points, but the results may not be the optimal solution. In Chap. 11, the problem of determining the optimal locations of pilot points is considered as an optimal experimental design problem. In Chap. 12, the problem of determining an appropriate number of pilot points is associated with the selection of an appropriate model complexity.

7.4.4.4 Plausibility

We have noted the existence of plausible realizations of a random field estimated by geostatistical inversion. After introducing pilot points and regularization terms for inversion, the plausibility problem is mitigated by additional conditioning, but not eliminated. In the pilot-point method, after an estimated $\hat{\theta}(\mathbf{x})$ is obtained by kriging or cokriging, its plausible realizations $\left\{\tilde{\theta}(\mathbf{x})\right\}$ can be obtained by conditional simulation (Gómez-Hernánez et al. 1997; Delhomme 1979)

$$\tilde{\theta}(\mathbf{x}) = \hat{\theta}(\mathbf{x}) + [\theta_a(\mathbf{x}) - \hat{\theta}_a(\mathbf{x})], \tag{7.4.63}$$

where $\theta_a(\mathbf{x})$ is a realitxzation drawn from the estimated random field and $\hat{\theta}_a(\mathbf{x})$ is a kriging or cokriging estimation of $\theta_a(\mathbf{x})$ conditioned at the same pilot points. Eq. (7.4.63) indicates that $\tilde{\theta}(\mathbf{x})$ and $\hat{\theta}(\mathbf{x})$ are realizations of the same random field and have the same values at all pilot points. Thus, the uncertainty caused by plausibility must be considered in uncertainty analysis (see Chap. 10).

7.5 Review Questions

1. Under what conditions the true structure and the true parameter values of a system can be identified approximately by solving EIP?
2. How the statistical EIP is formulated?
3. Why the adjoint-state method of model differentiation is very useful in the solution of EIP? And, how is it used for sensitivity analysis and structure selection?
4. Give more explanations on the stopping criteria given in Sect. 7.2.2. Especially, under what situation the criterion (iii) is satisfied but the criterion (ii) is not?

5. Compare the three adaptive parameterization methods given in Sect. 7.2.2, 7.2.3, and 7.3.1 by considering the simplicity, flexibility, computational effort, and convergence rate.
6. Why the consistence condition and the equal gradient condition are needed and how they can be satisfied in the two-scale inversion process?
7. What hyperparameters are needed for statistical inversion? Which of them are used for determining the statistical structure of a random field? Why these non-physical parameters can be estimated by state measurements?
8. What does "marginalization" mean? How the two marginalized MAP estimates in equations (7.4.23) to (7.4.26) are obtained?
9. What data are needed for cokriging estimate and how these data are obtained during the procedure of geostatistical inversion?
10. Read a paper from the recommended reference in Sect. 7.4.4 to find out the data requirement, algorithms, and problems associated with the regular pilot-point method when it is used for distributed parameter estimation.

Chapter 8
Development of Data-Driven Models

In previous chapters, we have mainly dealt with physics-based models represented by a general mapping $\mathbf{u} = \mathcal{M}(\boldsymbol{\theta})$, where $\mathbf{u}$ denotes state variable(s), $\mathcal{M}(\boldsymbol{\theta})$ is obtained from mathematical equation(s) describing the underlying physical processes, and $\boldsymbol{\theta}$ is a set of unknown physical parameters that need to be estimated from measurements. As mentioned in Chap. 1, various *data-driven models* that connect model inputs and outputs directly are also developed and used extensively in EWR fields. They are most useful when little a priori knowledge is available about the actual physical processes, or when it is desired to replace a physics-based model with a surrogate model for improving computational efficiency in optimization problems. The latter usage is referred to as reduced-order modeling or metamodeling.

In mathematics, data-driven models can be represented by a general mapping $\mathbf{z} = \mathcal{F}(\mathbf{x})$, where input variables $\mathbf{x}$ are sometimes called *predictors* and output variables $\mathbf{z}$ are called *targets*. These terms will be used interchangeably throughout this chapter. For a data-driven model, the mapping $\mathcal{F}(\mathbf{x})$ is not derived based on physical laws; rather, it is regarded as a black box with an unknown structure that needs to be trained by using co-observed input–output pairs, $(\mathbf{x}, \mathbf{z})$. The construction of a data-driven model is a process of identifying the mapping $\mathcal{F}(\cdot)$ with relevant input–output training data, which is known as *supervised machine learning* due to its origins in artificial intelligence (Haykin 1994). The term "supervised" refers to the fact that output data are also used during training. Note that $\mathbf{x}$ and $\mathbf{z}$ can be any variables that are useful for describing the observed excitation-response relationship of a system. When $\mathbf{x}$ is a *P*-dimensional vector and $\mathbf{z}$ is a *K*-dimensional vector, the mapping becomes a vector function $\mathbf{z} = \mathbf{f}(\mathbf{x})$, where $\mathbf{f} = \left\{f_1, f_2, \cdots, f_K\right\}$. When $K = 1$ (i.e., a scalar target variable), the problem of constructing a data-driven model becomes the identification of a single function $z = f(\mathbf{x})$.

We have learned how to identify a distributed function in Chaps. 6 and 7, in which the unknown function is a distributed physical parameter. Major techniques include parameterization, structure (or shape) parameter identification, hyperparameter estimation, and model structure reduction. Basically, all of these techniques can be used here for developing a data-driven model. The only difference is that the

N-Z. Sun, A. Sun, *Model Calibration and Parameter Estimation,*
DOI 10.1007/978-1-4939-2323-6_8

identified function is now a function that does not have much physical meaning. Major steps of data-driven modeling include the following:

- Finding a general functional form, $f(\mathbf{x};\boldsymbol{\alpha})$, as the model candidate, where $\boldsymbol{\alpha}$ is a set of undetermined model (structure) parameters. This step is similar to the parameterization step for distributed functions (see Chap. 6). A commonly used functional form is the linear combination of basis functions. In this case, the parameter vector $\boldsymbol{\alpha}$ may include weighting coefficients, shape parameters, and hyperparameters associated with the basis functions.
- Identifying the optimal model parameters $\boldsymbol{\alpha}$ through iteratively fitting the input–output data. In the language of artificial intelligence, this process is called training. Besides using various optimization algorithms learned in Chaps. 2 and 3, some efficient training algorithms will be introduced in this chapter for obtaining $\boldsymbol{\alpha}$.
- Selecting an appropriate model complexity to avoid the problem of overfitting or underfitting. For a data-driven model, its complexity is determined by the number of undetermined model parameters (i.e., the dimension of $\boldsymbol{\alpha}$). This is actually a model reduction problem or a model structure identification problem considered in Chap. 6 and 7. But, in this case the appropriate complexity of a data-driven model depends completely on the quantity and quality of training data. In this chapter, using training data to control the model complexity is considered the key to successful construction of a data-driven model. Besides regularization-based techniques, special methods used in machine learning will be introduced in this chapter.
- Testing the reliability of model prediction or the *generalization* capability of a trained model. This can be done by quantifying the model prediction error, also known as the *generalization error* in artificial intelligence, by testing the model on target values not used during training. This is actually a model validation process. The generalization error can be used to guide the selection of different structures in an iterative manner and with the aid of certain information criterion such as AIC and BIC described in Chap. 7. It is desirable to test the generalization performance of a developed model on different validation datasets. In reality, however, most training datasets often have limited number of records. Statistical sampling techniques such as bootstrapping are introduced in this chapter for artificially expanding datasets by randomly sampling the original training data. Further, ensemble methods such as bagging and boosting are also introduced to enhance the model performance while providing means for uncertainty quantification.

Section 8.1 describes some classical linear regression techniques, which remain the most widely used data-driven modeling methods and serve as the foundation for more sophisticated nonlinear machine learning methods. Section 8.2 discusses artificial neural networks, which are considered one of the most well-established nonlinear data-driven models. Section 8.3 introduces support vector machine and relevance vector machine algorithms, both belong to the so-called kernel methods. Finally, Sect. 8.4 describes Gaussian process regression methods, which are flexible Bayesian learning algorithms that possess attributes of both linear regression and kernel methods.

8.1 Linear Regression

Let us first consider linear models with P predictors $(x_i, i = 1, \cdots, P)$ and only a scalar target variable. This kind of model has the following general expression:

$$z = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_P x_P = \mathbf{w}^T \mathbf{x}, \tag{8.1.1}$$

where $\mathbf{x} = (1, x_1, x_2, \cdots, x_P)^T$ is the model input vector; $\mathbf{w} = (w_0, w_1, w_2, \cdots, w_P)^T$ is the weight vector, in which w_0 is the intercept term (also called the bias term); and z is the model output or target variable. The intercept term is used to account for a constant shift in data and can often be eliminated by preprocessing data (e.g., eliminating the mean of measurements). Because (8.1.1) uniquely determines the model structure of linear models, the model identification problem now becomes estimation of $\mathbf{w}$ with a dataset consisting of N input–output data pairs:

$$(\mathbf{x}_1, z_1),\ (\mathbf{x}_2, z_2),\ \cdots,\ (\mathbf{x}_N, z_N). \tag{8.1.2}$$

Substituting (8.1.2) into (8.1.1), we obtain a set of linear equations:

$$\mathbf{X}\mathbf{w} = \mathbf{z} + \boldsymbol{\varepsilon}, \tag{8.1.3}$$

where $\mathbf{X}$ is an $N \times (P+1)$ input data matrix with each row representing a sample of the input vector,

$$\mathbf{X} = \begin{pmatrix} 1 & x_1^1 & \cdots & x_P^1 \\ 1 & x_1^2 & \cdots & x_P^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^N & \cdots & x_P^N \end{pmatrix},$$

in which the superscript denotes the sample index; $\mathbf{z} = (z_1, z_2, \cdots, z_N)^T$ is an N-dimensional output data vector; and $\boldsymbol{\varepsilon}$ is an additive error vector that may contain the effect of both measurement error and model error.

Solving $\mathbf{w}$ from (8.1.3) is a linear inversion problem we are already very familiar with (see Chap. 2), but now $\mathbf{w}$ plays the role of the unknown parameters $\boldsymbol{\theta}$ in Eq. (2.2.1). When $\boldsymbol{\varepsilon}$ is Gaussian with zero mean and its components are i.i.d., the least squares solution of (8.1.3) can be obtained by minimizing the following L_2-norm objective function

$$S(\mathbf{w}) = \left\| \mathbf{X}\mathbf{w} - \mathbf{z} \right\|_2^2 . \tag{8.1.4}$$

When $\mathbf{X}^T\mathbf{X}$ is invertible, the least squares solution is (see Sect. 2.2.1),

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{z}. \tag{8.1.5}$$

More generally, we can find the pseudoinverse solution $\mathbf{w}^\dagger$ by using SVD as described in Sect. 2.2.2, viz.

$$\mathbf{w}^\dagger = \sum_{i=1}^{r} \frac{\mathbf{u}_{r,i}^T \mathbf{z}}{s_i} \mathbf{v}_{r,i}, \tag{8.1.6}$$

where $(s_1, s_2, \cdots, s_r)$ are r nonzero singular values obtained from the SVD of $\mathbf{X}$, $\mathbf{X} = \mathbf{U}_r \Sigma_r \mathbf{V}_r^T$, and $r \le \min(N, P+1)$ (See Eq. (2.2.15) for explanation of other symbols in the above equation). Eq. (8.1.6) provides a means for reducing redundant information, as we will learn soon toward the end of this subsection. In the statistical framework, the GLS solution of (8.1.3) and the variance of estimation can be found in Sect. 4.2.2.

8.1.1 Autoregressive Models

A large number of applications related to the linear regression problem (8.1.1) appear in time series analyses and forecasting. Time series has the unique property that all input data are ordered temporally. In this case, the input data and the target may be either observations of the same variable (endogenous) or different variables (exogenous). Special linear models and algorithms have been developed for time series regression. We will introduce several common ones below.

AR is the simplest time series regression model. It is used to predict the value of a variable on the basis of its own past states. Let us assume that the time series X_t is a zero-mean stationary stochastic process. A *p*-order AR model, AR(*p*), is defined as

$$X_t = \sum_{i=1}^{p} a_i X_{t-i} + V_t, \tag{8.1.7}$$

where subscript *t* is time index, $V_t \sim \mathcal{N}(0, \sigma_v^2)$ is white noise uncorrelated with X_s for all $s < t$, σ_v^2 is the variance of the noise. Eq. (8.1.7) has the same form as (8.1.3). For AR models, the unknown parameters a_i serve as a measure of persistence or correlation between X_t and its antecedent states. If the mean of X_t is nonzero, we can first remove the mean from X_t and then use the resulting time series to define AR(*p*) accordingly, thus eliminating the intercept term in (8.1.1). AR models assume temporal stationarity, and their parameters a_i must satisfy certain conditions to ensure existence and uniqueness of the solution, as shown by the following example.

Example 8.1 *Estimate Parameters of an AR(1) model*
Let us consider the AR(1) model given below,

$$X_t = aX_{t-1} + V_t. \tag{8.1.8}$$

The autocovariance function of X_t, $\rho(h)$, is obtained by multiplying both sides of (8.1.8) by X_{t-h} and taking expectations

$$\begin{aligned}\rho(h) &= C(X_t, X_{t-h}) = C(aX_{t-1}, X_{t-h}) + C(V_t, X_{t-h}) \\ &= C(aX_{t-1}, X_{t-h}) \\ &= a\rho(h-1),\end{aligned} \tag{8.1.9}$$

where h is the lag between two observation times, $C(\cdot)$ represents covariance, and $C(V_t, X_{t-h})$ is equal to zero by definition of the noise term V_t. Similar to spatial random processes introduced in Chap. 6, the autocovariance function of a temporal stationary process has the following basic properties,

$$\rho(0) \geq 0,$$

$$|\rho(h)| \leq \rho(0) \text{ for all } h, \text{ and}$$

$$\rho(h) = \rho(-h).$$

Applying the procedure in (8.1.9) recursively, we obtain the following expression for the autocovariance $\rho(h)$

$$\rho(h) = \rho(0)a^{|h|},\ h = 0, \pm 1, \ldots. \tag{8.1.10}$$

The sample autocovariance $\hat{\rho}(h)$ can be calculated as

$$\hat{\rho}(h) = \frac{1}{N}\sum_{t=1}^{N-|h|} X_{t+|h|}X_t,$$

where N is the number of samples. Multiplying both sides of (8.1.8) with X_t and taking expectations, we obtain

$$\begin{aligned}\sigma_X^2 &= aC(X_t, X_{t-1}) + C(X_t, V_t) \\ &= a^2\rho(0) + C(X_t, V_t)\end{aligned} \tag{8.1.11}$$

where σ_X^2 is the variance of X_t, and the first term in the second equality is obtained using the results from (8.1.10). Similarly, multiplying both sides of (8.1.8) by V_t and taking expectations, we get

$$C(X_t, V_t) = \sigma_v^2. \tag{8.1.12}$$

Thus, the variance and autocovariance of X_t are obtained by substituting (8.1.11) into (8.1.12) and (8.1.10), respectively,

$$\sigma_X^2 = \rho(0) = \frac{\sigma_v^2}{1-a^2}, \ \rho(h) = \frac{a^{|h|}\sigma_v^2}{1-a^2}. \tag{8.1.13}$$

Invoking the nonnegativity condition on variance, we see that the coefficient a must lie within $(-1,1)$. The AR(1) model (8.1.8) has a unique stationary solution

$$X_t = \sum_{i=0}^{\infty} a^i V_{t-i}, \tag{8.1.14}$$

which has zero mean and an autocovariance function as defined in (8.1.13). It is straightforward to show that (8.1.14) is a solution of (8.1.8). The proof of uniqueness is provided in Brockwell and Davis (2002, p. 52).

To estimate the unknown parameters in the AR(1) model, a and σ_v^2, we multiply both sides of (8.1.8) by X_{t-1} and then take expectations

$$\rho(1) - a\rho(0) = 0. \tag{8.1.15}$$

Using the sample autocovariance $\hat{\rho}(\cdot)$ to replace the autocovariance in (8.1.15) and (8.1.13), we obtain the solution for a and σ_v^2

$$a = \hat{\rho}(1) / \hat{\rho}(0), \ \sigma_v^2 = \hat{\rho}(0)(1-a^2). \qquad \blacksquare$$

The procedure delineated in Example 8.1 is generally known as the *Yule-Walker* algorithm. When applied to the AR(p) model (8.1.7), the Yule-Walker algorithm has the following matrix form

$$\begin{pmatrix} \rho(0) & \rho(1) & \cdots & \rho(p-1) \\ \rho(1) & \rho(0) & \cdots & \rho(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ \rho(p-1) & \rho(p-2) & \cdots & \rho(0) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{pmatrix} = \begin{pmatrix} \rho(1) \\ \rho(2) \\ \vdots \\ \rho(p) \end{pmatrix}, \tag{8.1.16}$$

which can be written in a compact form as

$$\mathbf{\Lambda a} = \boldsymbol{\rho}, \tag{8.1.17}$$

where $\mathbf{\Lambda} = (\rho_{i-j})$, $1 \le i, j \le p$, is the covariance matrix, $\mathbf{a} = (a_1, \cdots, a_p)^T$ is the parameter vector, and $\boldsymbol{\rho} = (\rho_1, \cdots, \rho_p)^T$ are values of autocovariance function at different lags. The symmetric matrix $\mathbf{\Lambda}$ is nonsingular if the autocovariance function satisfies: $\rho(0) > 0$ and $\rho(h) \to 0$ as $h \to \infty$.

After solving for a_i by replacing $\rho(\cdot)$ using sample estimates $\hat{\rho}(\cdot)$ in (8.1.17), we can estimate σ_v^2 from the equation below

$$\hat{\sigma}_v^2 = \hat{\rho}(0) - \mathbf{a}^T \hat{\boldsymbol{\rho}} = \hat{\rho}(0) - \sum_{i=1}^{p} a_i \hat{\rho}(i). \tag{8.1.18}$$

Other methods for estimating the parameters of an AR(p) model include linear estimators mentioned in Chap. 2 and Burg's algorithm (Burg 1968). The latter successively minimizes forward and backward one-step prediction errors and can sometimes yield better solutions than the Yule-Walker algorithm for pure autoregressive models. Interested readers may refer to Brockwell and Davis (2002, Chap. 5) for derivation of the Burg algorithm.

After the AR(p) model parameters are fitted using data, it can be used for prediction. For example, the one-step-ahead prediction given by the AR(p) model is

$$\hat{X}_{t+1} = \sum_{j=1}^{p} a_j X_{t+1-j} = \mathbf{a}^T \mathbf{X}_p, \tag{8.1.19}$$

where $\mathbf{X}_p = (X_{t+1-p}, \cdots, X_t)^T$.

8.1.1.1 Autoregressive-Moving-Average Model (ARMA)

ARMA models represent another special case of linear regression model, (8.1.3). It was originally described by Whittle (1953), but made popular by G.E.P. Box and G.M. Jenkins after the publication of their 1970 book (Box and Jenkins 1970). The ARMA(p, q) model for a mean-removed stationary time series X_t is

$$X_t = \sum_{i=1}^{p} a_i X_{t-i} + \sum_{j=1}^{q} b_j V_{t-j} + V_t, \tag{8.1.20}$$

where X_t is correlated not only with its past states, but also with past noise terms. Eq. (8.1.20) combines an AR(p) model and q moving-average terms, resulting $p + q + 1$ unknowns, namely, $\{a_i\}_{i=1}^{p}$, $\{b_j\}_{j=1}^{q}$, and σ_v^2. The parameters of an ARMA(p, q) model can be estimated using the Yule-Walker algorithm.

8.1.1.2 Autoregressive-Moving-Average with Exogenous Inputs (ARMAX)

The ARMAX model extends the AR and ARMA models further by including predictors other than the variable itself. Thus, an ARMAX(p, q; $\mathbf{n}^b$, $\mathbf{n}^d$) model is a combination of a p-order AR(p) model, q moving-average terms, and weighted contribution from a set of exogenous predictors

$$X_t = \sum_{i=1}^{p} a_i X_{t-i} + \sum_{k=1}^{K} \sum_{l=1}^{n_k^b} c_{kl} U_{k,t-n_k^d-l} + \sum_{j=1}^{q} b_j V_{t-j} + V_t, \qquad (8.1.21)$$

where K is the number of exogenous variables denoted by $\{U_{k,t}\}_{k=1}^{K}$, the integer vector $\mathbf{n}^b \in \mathbb{R}^K$ contains the orders of different exogenous variables, and $\mathbf{n}^d \in \mathbb{R}^K$ contains the delays associated with each of the exogenous variables (i.e., time elapsed before the effect of an exogenous variable is first reflected in the output). Comparing to the ARMA(*p, q*) model in (8.1.20), we see that the ARMAX model requires estimation of an additional set of parameters, $\{c_{kl}\}$.

Example 8.2 *Daily Streamflow Forecasting Using AR*
In this example, an AR model is developed for streamflow forecasting. The time series consists of daily streamflow records collected from a gauge located on Neches River in Texas, USA (USGS gauge no. 08033500) for the period 1948–2003. The drainage area upstream of the station is approximately 3600 mi^2 (9300 km^2). For the study period, the basin can be considered free of anthropogenic impacts (Duan 2003).

The distribution of streamflow rates is skewed. Thus, as part of the preprocessing the streamflow time series is first normalized by using the Box-Cox transformation defined below

$$Q_{bc} = \begin{cases} (Q^{\lambda} - 1)/\lambda, & \lambda \neq 0 \\ \log(Q), & \lambda = 0 \end{cases}$$

where Q_{bc} represents transformed streamflow and λ is a transformation parameter determined to be 0.072 using the Matlab function `boxcox`. The total time series is then split into two parts, one for training (1948–1992), and the rest for testing (1993–2003). A sample autocorrelation analysis on the time series indicated that the daily streamflow has relatively strong temporal correlation. To identify the optimal model order, the Matlab functions `arxstruc` and `selstruc` are used to evaluate the performance of AR models over a range of model orders. The best model order for one-day-ahead prediction is found to be 10,

$$\begin{aligned} \hat{Q}_{bc}(t+1) = {} & 1.457Q_{bc}(t) - 0.574Q_{bc}(t-1) + 0.1625Q_{bc}(t-2) - 0.0515Q_{bc}(t-3) \\ & + 0.00480Q_{bc}(t-4) - 0.00878Q_{bc}(t-5) - 0.0104Q_{bc}(t-6) \\ & + 0.00251Q_{bc}(t-7) + 0.00568Q_{bc}(t-8) + 0.0111Q_{bc}(t-9). \end{aligned}$$

In essence, the model-order selection problem for AR models is tantamount to model structure identification problem for distributed models. In addition to AIC and BIC mentioned in Chap. 7, a commonly used criterion in time series analyses is the Final Prediction Error (FPE) criterion originally proposed by Akaike (1969). The FPE is based on minimization of the mean squared error (MSE) associated with one-step prediction

$$\text{FPE} = \hat{\sigma}_v^2 \frac{N+p}{N-p}, \qquad (8.1.22)$$

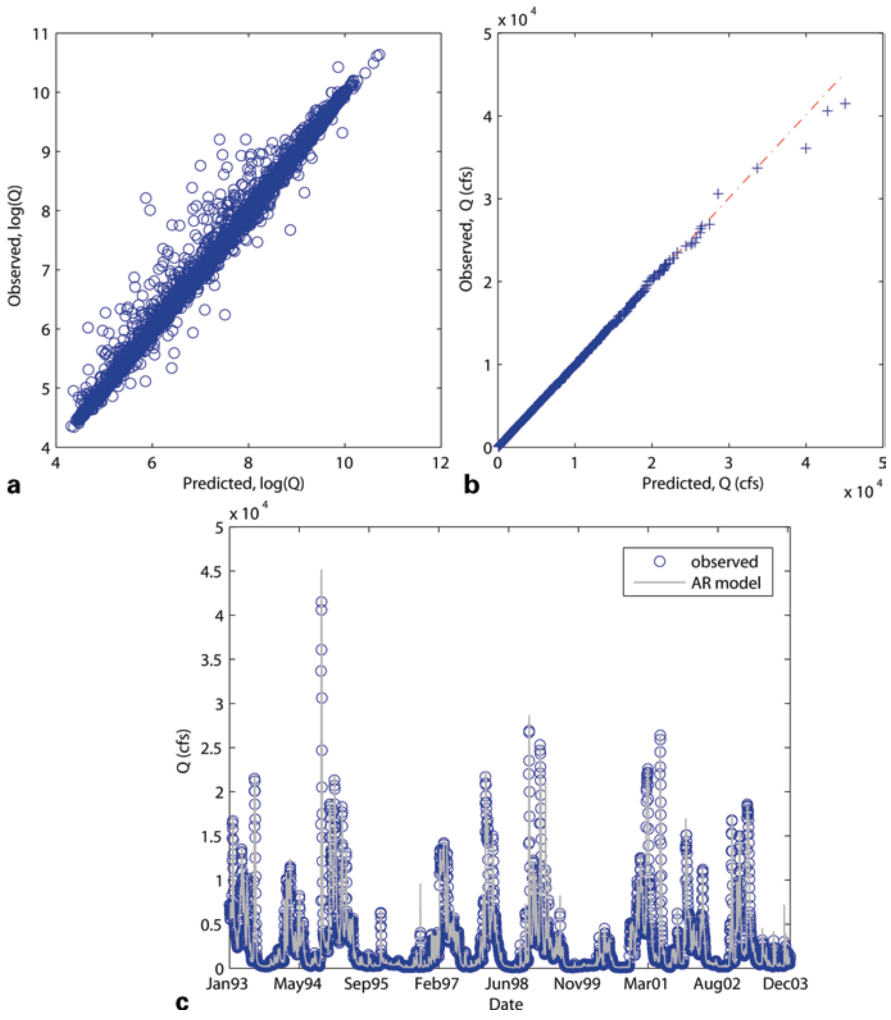


Fig. 8.1 **a** Scatterplot of predicted and observed streamflow (log-transformed) for Neches River watershed in Texas, USA. **b** Quantile–quantile plot. **c** Comparison between observed (open circles) and predicted streamflow obtained for the testing period

where $\hat{\sigma}_v^2$ is the estimated variance of the white noise of AR(p) model and N is sample size. For the current example, FPE is an appropriate model-order selection criterion because a large number of training samples are available.

Figure 8.1a, b show the scatterplot and quantile–quantile plot using predicted and observed streamflow values over the testing period. Figure 8.1c compares the time series of the same datasets. The Nash-Sutcliff coefficient (NSC) obtained by the AR model for the testing dataset is 0.978. Thus, the simple AR model achieved a relatively good performance for one-day-ahead streamflow prediction, although some peak values are underestimated as indicated by all three plots.

According to (8.1.19), the one-step AR model developed here can be extended to multistep-ahead prediction

$$\hat{Q}_{bc}(t+T) = \sum_{j=1}^{p} a_j Q_{bc}(t+T-j),$$

where T is the prediction interval (in multiples of days). Starting from $t+1$, when an intermediate forecast value is obtained, it is put back to the front of the predictor vector for estimating the next value until the prediction interval T is reached. In general, the prediction reliability deteriorates quickly as the prediction interval T increases.

Example 8.3 *ARMAX Models*
In practice, it is common to include multiple co-observed variables to improve prediction accuracy, as well as to enhance information content. We have seen examples of such multivariate applications in FA (Chap. 6) and cokriging (Chap. 7). For streamflow forecasting, the common predictors include both in situ and remotely sensed hydrometeorological variables. For example, Coulibaly et al. (2000) developed the following ARMAX model using both endogenous (streamflow) and exogenous (precipitation, snowmelt, and maximum and minimum temperatures) predictors,

$$\begin{aligned} Q(t) &= a_1 Q(t-1) + b_1 T_{\max}(t) + b_2 T_{\min}(t) + b_3 \bar{T}(t) \\ &\quad + \sum_{j=0}^{4} b_{4,j+1} P(t-j) + \sum_{j=0}^{4} b_{5,j+1} S(t-j) + e(t), \end{aligned} \tag{8.1.23}$$

where $T_{\max}$, $T_{\min}$, $\bar{T}$ are maximum, minimum, and average temperature, respectively; P is precipitation; and S is snowmelt. In (8.1.23), the order of the AR model is $p=1$, the number of moving-average terms is 0, and the orders of exogenous variables are [1, 1, 1, 5, 5] for $T_{\max}$, $T_{\min}$, $\bar{T}$, P, and S, respectively.

Sun et al. (2014b) used the following ARMAX model for one-month-ahead streamflow forecast:

$$\begin{aligned} Q(t) &= \sum_{i=1}^{2} a_i Q(t-i) + b_1 P(t-1) + b_2 \bar{P}(t) + b_3 T_{\min}(t-1) + b_4 \bar{T}_{\min}(t) \\ &\quad + \sum_{j=1}^{2} b_{5,j} T_{\max}(t-j) + b_6 \bar{T}_{\max}(t) + e_t, \end{aligned}$$

in which the long-term monthly averages of precipitation and temperature (symbols with overbar) for the predicting month are incorporated as predictors.

In hydrology, time series methods are best suited for short-term forecasting based on daily or weekly timescales, but become less accurate for long-term forecasting involving seasonal or annual timescales, neither can they handle nonlinearities inherent in rainfall-runoff processes very well (Sun et al. 2014b). The deficiencies associated with linear regression models have prompted the development of various machine learning methods that are shown to have better capability for handling nonlinear models. This topic will be introduced starting from Sect. 8.2.

8.1.2 Model Complexity Control

The complexity of model (8.1.1) is determined by the number of predictors, P. Using a large number of predictors may cause the over-fitting problem (see Sect. 3.2.2), making the identified weights $\mathbf{w}$ unstable and the results of model prediction unreliable. Common selection criteria, such as AIC, BIC, and FPE, can be used to test different combinations of predictors in a systematic manner to select the best performer. A statistical procedure for doing so is cross-validation, which will be described shortly in the next subsection.

Other methods attempt to reduce information redundancy by looking into correlation between predictors and target variables, and among predictors themselves. We have learned several such dimension reduction methods in Sect. 6.3. For example, PCA and FA can be used to decrease the number of predictors, but they do not incorporate model outputs in the dimension-reduction process. The *principal component regression* (PCR) (Mandel 1982) is built on the SVD of input data matrix and its solution is already given in (8.1.6). If the number of nonzero singular values, r, is less than P, it implies information redundancy and colinearity among some of the P predictors. Mathematically, the PCR method is the same as the TSVD method introduced for parameter estimation (see Sect. 2.2.3).

A common issue with FA, PCR, or TSVD is that the transformed predictors (i.e., latent variables) no longer possess physical meanings. Here we describe a class of linear regression methods called shrinkage (or weight decay) methods that can help identify a subset of the predictors exhibiting the strongest impact on model prediction while preserving the physical meanings of predictors. By reducing the number of DOF, the hope is that the inverse solution can become more stable, the model variability caused by the correlation between predictors is reduced, and consequently the model prediction can become more accurate than that of the full model.

Shrinkage methods attempt to decrease the weights of insignificant predictors to effectively reduce model complexity. Because all predictors may not have the same dimensions and scales, it is important to preprocess the input data in (8.1.2) before applying shrinkage methods. This can be done by making the measurements of all predictors dimensionless and normalized (see discussion under Sect. 2.1.4). However, the intercept term w_0 must be calculated before this process. Taking the mean of (8.1.1) and assuming $\bar{x}_1 = \bar{x}_2 = \cdots = \bar{x}_P = 0$, we will have $w_0 = \bar{z}$. Therefore, if we remove the respective sample means from input and output data, the intercept w_0 in (8.1.1) can be eliminated. In the rest of this discussion, we assume that all sample means of input and output data in (8.1.2) have been removed, and all input datasets are dimensionless and normalized. The dimensions of input data matrix $\mathbf{X}$ are now $N \times P$.

Shrinkage methods use the following regularized objective function to control the sizes of its solution

$$S_{\alpha,q}(\mathbf{w}) = \left\{ \|\mathbf{X}\mathbf{w} - \mathbf{z}\|_2^2 + \alpha \sum_{i=1}^{P} |w_i|^q \right\}, \tag{8.1.24}$$

where the power $q > 0$, and $\alpha > 0$ is a penalty coefficient to be determined. An often-seen shrinkage method is called the *ridge regression method* (Hoerl and Kennard 1970) in which $q = 2$. In this case, the objective function in (8.1.24) becomes

$$S_{\alpha,2}(\mathbf{w}) = \left\{ \|\mathbf{X}\mathbf{w} - \mathbf{z}\|_2^2 + \alpha \|\mathbf{w}\|_2^2 \right\}. \tag{8.1.25}$$

As in the case of the full regression problem (8.1.5), when $\mathbf{X}^T\mathbf{X}$ is invertible the following ridge regression solution can be obtained explicitly

$$\mathbf{w}_\alpha = (\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{X}^T\mathbf{z}. \tag{8.1.26}$$

From (8.1.26), we see that α is a complexity control parameter: (i) when $\alpha \to 0$, $\mathbf{w}_\alpha$ approaches the solution of the full regression problem (8.1.5); (ii) when α increases, the amount of "shrinkage" increases and the components of $\mathbf{w}_\alpha$ are forced to decrease; and (iii) when $\alpha \to \infty$, $\mathbf{w}_\alpha \to \mathbf{0}$ (i.e., the DOF of the model becomes zero). Recall that the objective function in (8.1.25) has been used in Sect. 3.3.2 for linear model regularization problems, with α acting as the regularization coefficient.

Another shrinkage method is called the *Lasso method* (Tibshirani 1996), which is obtained by setting $q = 1$ in (8.1.24), viz.

$$S_{\alpha,1}(\mathbf{w}) = \left\{ \|\mathbf{X}\mathbf{w} - \mathbf{z}\|_2^2 + \alpha \|\mathbf{w}\|_1 \right\}, \tag{8.1.27}$$

where the regularization term is defined by L_1-norm. As a result, the first-order derivatives of $S_{\alpha,1}(\mathbf{w})$ are discontinuous at the origin $\mathbf{w} = \mathbf{0}$ and the minimizer of $S_{\alpha,1}(\mathbf{w})$ does not have an analytical expression. Usually, Lasso is solved by quadratic programming or a numerical optimization algorithm. In this case, increasing α will force some of the weights to be identically zero, leading to a sparse linear regression model.

Although both ridge regression and Lasso can control the model complexity continuously by varying α, there is a main difference between the two. In ridge regression, when α increases, all components of $\mathbf{w}$ are reduced proportionally but still remain nonzero, while in Lasso the increase of α will cause more components of $\mathbf{w}$ to become identically zero.

A known shortcoming of the Lasso method is that when there is a group of predictors with high pairwise correlation, it tends to select only one predictor from the group and does not care which one is selected. Zou and Hastie (2005) introduced an *elastic net method* which, in addition to automatic variable selection and continuous shrinkage, can also select groups of correlated variables. The penalty term in the elastic net method is a hybrid of those used in ridge regression and Lasso methods, viz.

$$\alpha\left[(1-\beta)\left\|\mathbf{w}\right\|_1+\beta\left\|\mathbf{w}\right\|_2^2\right], \tag{8.1.28}$$

in which β is an additional parameter introduced to control model complexity. The elastic net method has been shown to outperform both Lasso and ridge regression. Detailed discussion on these shrinkage methods, including method comparisons and application examples, can be found in Hastie et al. (2009).

As mentioned in the beginning of this section, a close connection exists between the linear regression methods discussed here and the linear model inversion methods considered in the Bayesian framework in Chap. 4. The solution in (8.1.26) has been obtained in Sect. 4.2.3 under the assumptions that the prior distribution of $\mathbf{w}$ is $\mathcal{N}(\mathbf{0},\sigma_P^2\mathbf{I})$ and the distribution of observation error is $\mathcal{N}(\mathbf{0},\sigma_D^2\mathbf{I})$. In this case, we can also have the variance of estimation given by (4.2.20), and the minimum variance corresponds to $\alpha=\sigma_P^2/\sigma_D^2$. Similarly, Lasso corresponds to the use of a double-exponential prior (also known as Laplacian or Gaussian kernel) for α (Tibshirani 1996); this link prompts some authors to treat α as a hyperparameter and adopt Markov Chain Monte Carlo (MCMC) to estimate Lasso parameters (Park, Casella 2008; Kyung et al. 2010). In Sect. 8.3 and 8.4, we will introduce kernel-based learning algorithms that explicitly incorporate the PDF of weights in a hierarchical Bayesian framework; many of the kernel methods also yield sparse solutions. In EWR, shrinkage methods have been used to select predictors for statistical downscaling global climate model outputs (Hammami et al. 2012; Tareghian and Rasmussen 2013).

8.1.3 Multiple Target Regression

So far, we have mainly dealt with a scalar target variable. Extension to multiple target variables is possible if they all share the same set of predictors. The general form of linear models with K model outputs (targets) consists of a set of linear equations, one linear model (8.1.1) for each different target, viz.

$$z_j=w_{j,0}+w_{j,1}x_1+w_{j,2}x_2+\cdots+w_{j,P}x_P=\mathbf{w}_j^T\mathbf{x},\ (j=1,2,\cdots,K). \tag{8.1.29}$$

The model can be expressed in a matrix form as

$$\mathbf{z} = \mathbf{f}(\mathbf{x}, \mathbf{W}) = \mathbf{W}^T \mathbf{x}, \tag{8.1.30}$$

where $\mathbf{x} = (1, x_1, x_2, \cdots, x_P)^T$ is the model input vector, $\mathbf{z} = (z_1, z_2, \cdots, z_K)^T$ is the model output vector and each of its component represents a different target variable; $\mathbf{W}$ is a $(P+1) \times K$ weight matrix,

$$\mathbf{W} = \begin{pmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_K \end{pmatrix} = \begin{pmatrix} w_{1,0} & w_{2,0} & \cdots & w_{K,0} \\ w_{1,1} & w_{2,1} & \cdots & w_{K,1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1,P} & w_{2,P} & \cdots & w_{K,P} \end{pmatrix}. \tag{8.1.31}$$

The identification of model (8.1.30) becomes estimation of $\mathbf{W}$ with a set of N input–output data pairs

$$(\mathbf{x}_1, \mathbf{z}_1),\ (\mathbf{x}_2, \mathbf{z}_2),\ \cdots,\ (\mathbf{x}_N, \mathbf{z}_N). \tag{8.1.32}$$

Substituting (8.1.32) into (8.1.30) yields

$$\mathbf{XW} = \mathbf{Z} + \mathbf{E}, \tag{8.1.33}$$

where $\mathbf{X}$ is an $N \times (P+1)$ matrix of input data, $\mathbf{Z}$ is an $N \times K$ matrix of output data, and $\mathbf{E}$ is an $N \times K$ matrix of observation error. Using the same process for deriving (8.1.5) and assuming that the components of $\mathbf{E}$ are i.i.d. with zero mean and $\mathbf{X}^T\mathbf{X}$ is invertible, the weight matrix $\mathbf{W}$ can be estimated explicitly by

$$\hat{\mathbf{W}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Z}. \tag{8.1.34}$$

Equation (8.1.34) shows that columns of $\hat{\mathbf{W}}$ can be solved column by column because $\hat{\mathbf{w}}_j = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{z}_j$, which is exactly the least squares solution (8.1.5) for the j-th target. In other words, a multiple target linear model can be identified separately for each target. Therefore, the shrinkage methods can still be used to the multiple target case by finding either different regularization coefficients for different target or the same one for all targets. Readers may refer to Hastie et al. (2009) for detailed discussions and examples on this topic. Another example will appear in the context of metamodeling (see Sect. 8.2.4), in which the same set of predictors are used to predict model outputs at multiple observation locations.

8.1.4 Linear Regression with Basis Functions

Assume that the model input $\mathbf{x}$ is a P-dimensional vector and the model output z is a scalar, but the model $z = f(\mathbf{x})$ is a nonlinear function. Our purpose is to identify the model with the same input–output training data pairs given in (8.1.2).

In Chap. 6, we have learned parameterization of a distributed parameter $\theta(\mathbf{x})$ for inversion. The same approach can be used here to identify the model $f(\mathbf{x})$. The basic assumption is that $f(\mathbf{x})$ can be represented approximately by

$$z = f(\mathbf{x};\mathbf{w}) \approx \sum_{i=0}^{M} w_i \phi_i(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}). \tag{8.1.35}$$

In the above equation, $\boldsymbol{\phi}(\mathbf{x}) = (\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \cdots, \phi_M(\mathbf{x}))^T$ are M+1 basis functions, and $\mathbf{w} = (w_0, w_1, \cdots, w_M)^T$ are weights associated with these basis functions. If $\phi_0(\mathbf{x}) \equiv 1$, the first term of the summation in (8.1.35) becomes the bias term w_0. We have seen in Chap. 6 that (8.1.35) can represent a wide range of functions, continuous or discontinuous, linear or nonlinear, by selecting different classes of basis functions.

After the model form (8.1.35) is specified and its basis functions are selected (this means the model structure is determined), the training task becomes estimating weights $\mathbf{w}$ using the input–output pairs. Note that model (8.1.35) is nonlinear with respect to $\mathbf{x}$ when the basis functions are nonlinear, but it is always linear with respect to $\mathbf{w}$ and, thus, can be estimated by linear regression.

Substituting all N training data pairs in (8.1.2) into (8.1.35), the following set of N linear equations are obtained:

$$\boldsymbol{\Phi}(\mathbf{x})\mathbf{w} = \mathbf{z} + \boldsymbol{\varepsilon}, \tag{8.1.36}$$

where the coefficient matrix is defined by

$$\boldsymbol{\Phi}(\mathbf{x}) = \begin{pmatrix} 1 & \phi_1(\mathbf{x}_1) & \cdots & \phi_M(\mathbf{x}_1) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(\mathbf{x}_N) & \cdots & \phi_M(\mathbf{x}_N) \end{pmatrix}. \tag{8.1.37}$$

Assuming that all observation errors $(\varepsilon_1, \cdots, \varepsilon_N)$ are i.i.d., all weights can then be solved from this set of equations by the least squares method, and the training of model (8.1.35) is thus completed. From (8.1.37), it can be seen that the linear model regression (8.1.3) is nothing but a special case of (8.1.35) when $\phi_i(\mathbf{x}) = x_i$ for $i = 1, 2, \cdots, M$ and $M = P$.

A large number of nonlinear basis functions exist in the literature. Some of them, such as the linear basis function, polynomial basis functions, RBF, and level set basis functions, have already been used in previous chapters. More will be defined later in this chapter.

8.1.4.1 Cross-Validation and Model Reliability

A trained data-driven model may not be a useful one because of model and data errors. The real relationship between **x** and **z** may not be accurately represented by the functional form in (8.1.35), the number of terms and the form of basis functions may not be appropriately selected, and the estimated weights (i.e., model parameters) may contain significant errors due to inadequacy and inaccuracy of data used for model training. These issues are exactly the same as what we have seen in Chap. 6 when parameterizing a distributed parameter.

Therefore, the correctness of a trained model must be verified and its reliability must be tested before it is used. If there are new input–output data available after the model is constructed, we can certainly use them to validate and test the usefulness of the model. When there are no new data available, however, we can still do some verification and testing using the existing data. The basic idea of *cross-validation* (CV) is to partition the existing data into two portions, one for model training and the other for model validation. Such notion of cross-validation has already been implicitly applied in Example 8.2 when determining the order of AP model.

The most often used CV method is the *K*-fold CV, which partitions the whole training dataset into *K* groups of equal sizes $\{G_1, G_2, \cdots, G_K\}$. Common choices are $K = 5$ and $K = 10$. For generality, let $\mathbf{z} = \mathbf{f}(\mathbf{x})$ be the model to be identified and $\hat{\mathbf{z}}_{-k} = \hat{\mathbf{f}}_{-k}(\mathbf{x})$ be the model trained without using the data in group G_k. The mean square error of *CV* when the data in G_k are used for validation is given by

$$(CVE)_k = \frac{1}{N_k} \sum_{(\mathbf{x},\mathbf{z}) \in G_k} \left\| \mathbf{z} - \hat{\mathbf{f}}_{-k}(\mathbf{x}) \right\|_2^2, \tag{8.1.38}$$

where N_k is the number of data pairs in G_k, and the average overall CV error is

$$CVE = \frac{1}{K} \sum_{k=1}^{K} (CVE)_k. \tag{8.1.39}$$

This error can be used not only for validating a trained model, but also for model selection. When there is a set of candidate models, we can calculate the *CVE* for each of them and select the one that has the smallest *CVE*. For example, using different values of regularization coefficient α to minimize the objective function

$$S_\alpha(\mathbf{x}, \mathbf{w}) = \left\| \mathbf{f}(\mathbf{x}, \mathbf{w}) - \mathbf{z} \right\|_2^2 + \alpha \left\| \mathbf{w} \right\|_2^2, \tag{8.1.40}$$

we will have a set of candidate models $\{\hat{\mathbf{z}}_\alpha = \hat{\mathbf{f}}_\alpha(\mathbf{x})\}$. Let the *CVE*, (8.1.39), associated with model $\hat{\mathbf{f}}_\alpha(\mathbf{x})$ be $CVE(\alpha)$, then the optimal regularization coefficient would be $\alpha^* = \arg\min_\alpha CVE(\alpha)$. Similarly, we can use CV in the selection of model structure, such as determining the optimal number and shape parameters of basis functions.

The CV method, however, is computationally expensive when the size of data and the number of unknowns are large. Moreover, a model with small validation error does not mean it must be reliable for model application. Unlike a physics-based model, a data-driven model may be good for interpolation, but is usually poor for extrapolation when the model application is out of the range of training data.

8.2 Artificial Neural Network

We have seen in Sect. 8.1.4 that a nonlinear model can still be trained by linear regression if it is approximated by a linear combination of known basis functions. But this is not always possible. For example, when the true system is $z = a_1 x / (a_2 + x)$, the identification of a_1 and a_2 is a nonlinear regression problem. In this section, we introduce artificial neural network (ANN) that can handle both linear and nonlinear mappings.

ANN uses layers of interconnected information processing units, or artificial neurons, to mimic the learning process of human brains. Because of its flexible structure and tunable structure parameters, ANN has become a popular method for constructing data-driven models. Introduced originally in 1940s for representing information processing in biological systems (McCulloch and Pitts 1943), ANN models have been used in various fields, including EWR. For example, ANN has been applied to rainfall-runoff forecasting (Chang et al. 2007; Christian and Wilby 1998; Coulibaly et al. 2001b; Hsu et al. 1995; ASCE 2000; Moradkhani et al. 2004), urban water demand forecasting (Liu et al. 2003), water resources management (Bowden et al. 2005; Maier and Dandy 2000), groundwater level forecasting (Coppola et al. 2005; Coulibaly et al. 2001a; Sun 2013), and statistical downscaling of climate models (Cannon and Whitfield 2002; Schoof and Pryor 2001; Tisseuil et al. 2010). A recent review indicates that prediction of water resource variables using ANNs has become a well-established research area over the last two decades (Maier et al. 2010). ANN has also found its applications in the inversion and dimension reduction of physics-based models (i.e., metamodeling). The main strengths of ANN models include the following: (i) they can be trained to learn both linear and nonlinear mappings; (ii) once trained, an ANN model has the capability to quickly generate useful results even for inputs not encountered during training, provided that a sufficient number of bounding input patterns are used for training; and (iii) a trained ANN is relatively robust to process noise (Haykin 1994).

8.2.1 Multilayer Perceptron Networks

A number of ANN architectures have been developed over the years. The best known and arguably the most widely used ANN is *multilayer perceptron network* (MLP), which is a type of *feedforward neural networks* (FNN). As its name suggests, an FNN is obtained by connecting multiple layers one-by-one in a forward-only direction through a series of transformations. The outputs are not used to provide feedbacks to the input and, thus, the FNNs are referred to as static neural networks, as opposed to dynamic neural networks.

In general, an MLP can be represented by a connected network of the form

$$\begin{aligned} \mathbf{x} \equiv \mathbf{y}^{(0)} \underset{[\mathbf{W}^{(1)}]}{\rightarrow} \mathbf{y}^{(1)} \underset{[\mathbf{W}^{(2)}]}{\rightarrow} \mathbf{y}^{(2)} \bullet \bullet \bullet \underset{[\mathbf{W}^{(l)}]}{\rightarrow} \mathbf{y}^{(l)} \rightarrow \\ \bullet \bullet \bullet \rightarrow \mathbf{y}^{(L-1)} \underset{[\mathbf{W}^{(L)}]}{\rightarrow} \mathbf{y}^{(L)} \underset{[\mathbf{W}^{(L+1)}]}{\rightarrow} \mathbf{y}^{(L+1)} \equiv \mathbf{z} \end{aligned} \tag{8.2.1}$$

where $\mathbf{x} \equiv \mathbf{y}^{(0)}$ is an input layer, $\left\{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \cdots, \mathbf{y}^{(L)}\right\}$ are L hidden layers, and $\mathbf{y}^{(L+1)} \equiv \mathbf{z}$ is an output layer. Each layer consists of a number of neurons (or units) that receive, process, and transmit information. Neurons of the l-th layer $\mathbf{y}^{(l)}$ are denoted by $\left\{y_1^{(l)}, y_2^{(l)}, \cdots, y_i^{(l)}, \cdots, y_{M_l}^{(l)}\right\}$, in which M_l is the number of neurons used by the layer; and $\mathbf{W}^{(l)}$ $(l = 1, 2, \cdots, L+1)$ are weight matrices. An element $w_{ij}^{(l)}$ of $\mathbf{W}^{(l)}$ is the weight that links a neuron $y_i^{(l-1)}$ of layer $(l-1)$ to a neuron $y_j^{(l)}$ of layer l. Fig. 8.2 illustrates an MLP with a single hidden layer.

An MLP defined in (8.2.1) is characterized by the following equations:

$$\begin{gathered} a_j^{(l)} = w_{0j}^{(l)} + \sum_{i=1}^{M_{l-1}} w_{ij}^{(l)} y_i^{(l-1)} = \sum_{i=0}^{M_{l-1}} w_{ij}^{(l)} y_i^{(l-1)}, \\ y_j^{(l)} = \phi(a_j^{(l)}), \\ (j = 1, 2, \cdots, M_l;\ l = 1, 2, \cdots, L+1) \end{gathered} \tag{8.2.2}$$

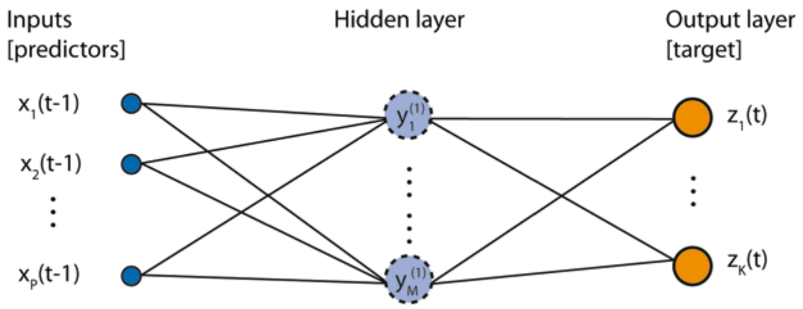


Fig. 8.2 An MLP with multiple inputs, one hidden layer, and multiple outputs, in which the number of neurons in the input, hidden, and output layers are $M_0 = P, M_1 = M$, and $M_2 = K$

Equation (8.2.2) shows how a neuron $y_j^{(l)}$ of layer l is determined by all neurons of layer $(l-1)$, in which $a_j^{(l)}$ is called the activation, and function $\phi(\cdot)$ is known as the activation function or transfer function. Some commonly used activation functions include:

- Linear function: $\phi(a) = a$
- Hyperbolic tangent sigmoid function: $\phi(a) = \dfrac{1-e^{-2a}}{1+e^{-2a}}$
- Gaussian function (mean, μ; variance, σ^2): $\phi(a) = e^{-(a-\mu)^2/2\sigma^2}$
- Logistic sigmoid function: $\phi(a) = \dfrac{1}{1+e^{-a}}$

Comparing the first equation of (8.2.2) to the linear regression model (8.1.35), we see that the neurons of layer $(l-1)$ essentially serve as basis functions for neurons of layer l, although the dependency is nonlinear when the selected activation function is nonlinear. This means that the data-driven model, $\mathbf{z} = \mathbf{f}(\mathbf{x}, \mathcal{W})$, defined by (8.2.1) and (8.2.2), is a nonlinear model with respect to both $\mathbf{x}$ and $\mathcal{W}$, where $\mathcal{W}$ denotes the set of all weight matrices, $\mathcal{W} = \left\{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \cdots, \mathbf{W}^{(L+1)}\right\}$. It has been shown that the MLP, even with only one hidden layer (Fig. 8.2), can uniformly approximate any continuous mapping $\mathbf{z} = \mathbf{f}(\mathbf{x})$ to arbitrary accuracy, provided that the network has a sufficiently large number of hidden units. This result is known as the *universal approximation theorem* (Cybenko 1989). The theorem, however, does not say how many hidden neurons should be used. A popular strategy is to treat the number of hidden neurons as additional unknowns to be adjusted during network training. In the case of a large number of inputs, it is worthwhile to include more than one hidden layer to improve network training efficiency (Bishop 2006).

8.2.2 Training of an ANN

8.2.2.1 The Training Process

Once the structure of an ANN model $\mathbf{z} = \mathbf{f}(\mathbf{x}, \mathcal{W})$ is determined, the model construction problem becomes identifying weight matrices $\mathcal{W}$ using the input–output data pairs given in (8.1.32). There exist two approaches to train an ANN model: (i) using all data pairs simultaneously and (ii) using data pairs sequentially by a recursive procedure. When a data pair $(\mathbf{x}_n, \mathbf{z}_n)$ is used to train, the objective of training is to minimize the following fitting residual of the model outputs

$$S_n(\mathbf{x}_n, \mathbf{z}_n, \mathcal{W}) = \frac{1}{2}\left\|\mathbf{y}_n^{(L+1)}(\mathcal{W}) - \mathbf{z}_n\right\|_2^2 = \frac{1}{2}\sum_{j=1}^{M_{L+1}} e_{n,j}^2, \tag{8.2.3}$$

where $\mathbf{y}_n^{(L+1)}(\mathcal{W}) = \mathbf{y}^{(L+1)}(\mathbf{x}_n, \mathcal{W}) \equiv \mathbf{z}(\mathbf{x}_n, \mathcal{W})$ is the model output layer, and

$$e_{n,j} = \sum_{i=1}^{M_L} w_{ij}^{(L+1)} y_{n,i}^{(L)} - z_{n,j}, \ j = 1, \cdots, M_{L+1}. \tag{8.2.4}$$

When all N data pairs are used for training, the objective is to minimize the total fitting residual

$$S(\mathbf{X}, \mathbf{Z}, \mathcal{W}) = \sum_{n=1}^{N} S_n(\mathbf{x}_n, \mathbf{z}_n, \mathcal{W}). \tag{8.2.5}$$

Because of the nonlinearity of the model, an iteration process is required for identifying the unknown set of weight matrices $\mathcal{W}$. This process is exactly the same as that was described in Chap. 2 for nonlinear model inversion, namely, the objective function (8.2.5) is minimized gradually by generating a series of weight matrix sets:

$$\mathcal{W}_0, \mathcal{W}_1, \cdots, \mathcal{W}_t, \mathcal{W}_{t+1}, \cdots \tag{8.2.6}$$

where $\mathcal{W}_0$ is an initial guess, and $\mathcal{W}_{t+1}$ is an update of $\mathcal{W}_t$ such that condition $S(\mathbf{X}, \mathbf{Z}, \mathcal{W}_{t+1}) < S(\mathbf{X}, \mathbf{Z}, \mathcal{W}_t)$ is satisfied for $t = 0, 1, \cdots$. This process is terminated until a convergence criterion is reached.

8.2.2.2 Backpropagation Algorithm

This algorithm provides a way to find $\mathcal{W}_{t+1}$ from $\mathcal{W}_t$ by iteration. It consists of two distinct passes:

- *Forward pass* solves the forward problem. For a given input layer $\mathbf{x}_n$ and the current weight matrix set $\mathcal{W}_t$, calculate all hidden layers and the output layer. This can be done according to (8.2.1): first $\mathbf{y}_n^{(0)} \equiv \mathbf{x}_n$ and $\mathbf{W}_t^{(1)}$ are used to obtain the hidden layer $\mathbf{y}_n^{(1)}$, then $\mathbf{y}_n^{(1)}$ and $\mathbf{W}_t^{(2)}$ are used to obtain $\mathbf{y}_n^{(2)}$ and so forth, until finally $\mathbf{y}_n^{(L)}$ and $\mathbf{W}_t^{(L+1)}$ are used to obtain the output layer $\mathbf{y}_n^{(L+1)} \equiv \mathbf{z}(\mathbf{x}_n, \mathcal{W}_t)$. The value of the objective function $S_n(\mathbf{x}_n, \mathbf{z}_n, \mathcal{W}_t)$ can then be calculated according to (8.2.3).
- *Backward pass* solves the inverse problem to estimate the final weight matrix $\mathbf{W}_{t+1}^{(L+1)}$ first with the observed data $\mathbf{z}_n$, then a recursive process is used to estimate $\mathbf{W}_{t+1}^{(L)}, \mathbf{W}_{t+1}^{(L-1)}, \cdots$, until $\mathbf{W}_{t+1}^{(2)}$, and finally $\mathbf{W}_{t+1}^{(1)}$ is estimated with the input data $\mathbf{x}_n$. The set of all updated weights $\mathcal{W}_{t+1}$ is thus obtained. Details of this process are shown below. For clarity, the subscript $(t+1)$ associated with all identified weights will be omitted.

Train the Output Layer $\mathbf{y}_n^{(L+1)}$ for Identifying $\mathbf{W}^{(L+1)}$
From (8.2.3), for all $i = 1, 2, \cdots, M_L$ and $j = 1, 2, \cdots, M_{L+1}$, we have

$$\frac{\partial S_n}{\partial w_{ij}^{(L+1)}} = e_{n,j} \frac{\partial y_{n,j}^{(L+1)}}{\partial a_j^{(L+1)}} \frac{\partial a_j^{(L+1)}}{\partial w_{ij}^{(L+1)}} = e_{n,j} \phi'(a_j^{(L+1)}) y_{n,i}^{(L)} = \delta_{n,j}^{(L+1)} y_{n,i}^{(L)}, \tag{8.2.7}$$

where

$$\delta_{n,j}^{(L+1)} = e_{n,j}\phi'(a_j^{(L+1)}). \tag{8.2.8}$$

Because $\delta_{n,j}^{(L+1)}$ and $y_{n,i}^{(L)}$ are known, the gradients of objective function S_n with respect to all $w_{ij}^{(L+1)}$ are obtained by (8.2.7). Various gradient-based optimization algorithms in Sect. 2.3.3 can be used here to identify these weights $(\mathbf{W}^{(L+1)})$.

Train the Hidden Layer $\mathbf{y}_n^{(L)}$ for Identifying $\mathbf{W}^{(L)}$

The objective function S_n in (8.2.3) is only indirectly dependent on $\mathbf{y}_n^{(L)}$ through $\mathbf{y}_n^{(L+1)}$, which means S_n is also dependent on $w_{ij}^{(L)}$. For all $i = 1, 2, \cdots, M_{L-1}$ and $j = 1, 2, \cdots, M_L$ we have the following expression according to the chain rule

$$\begin{aligned}\frac{\partial S_n}{\partial w_{ij}^{(L)}} &= \sum_{r=1}^{M_{L+1}} \frac{\partial S_n}{\partial y_{n,r}^{(L+1)}} \frac{\partial y_{n,r}^{(L+1)}}{\partial a_r^{(L+1)}} \frac{\partial a_r^{(L+1)}}{\partial y_{n,j}^{(L)}} \frac{\partial y_{n,j}^{(L)}}{\partial a_j^{(L)}} \frac{\partial a_j^{(L)}}{\partial w_{ij}^{(L)}} \\ &= \sum_{r=1}^{M_{L+1}} e_{n,r}\phi'(a_r^{(L+1)}) w_{jr}^{(L+1)} \phi'(a_j^{(L)}) y_{n,i}^{(L-1)} \\ &= \left(\sum_{r=1}^{M_{L+1}} \delta_{n,r}^{L+1} w_{jr}^{(L+1)} \right) \phi'(a_j^{(L)}) y_{n.i}^{(L-1)} = \delta_{n,j}^{(L)} y_{n,i}^{(L-1)},\end{aligned} \tag{8.2.9}$$

where

$$\delta_{n,j}^{(L)} = \left(\sum_{r=1}^{M_{L+1}} \delta_{n,r}^{(L+1)} w_{jr}^{(L+1)} \right) \phi'(a_j^{(L)}). \tag{8.2.10}$$

Because $\delta_{n,j}^{(L)}$ and $y_{n,i}^{(L-1)}$ are known, the gradients of objective function S_n with respect to all $w_{ij}^{(L)}$ are obtained by (8.2.9), a gradient-based optimization algorithms then can be used to identify these weights $(\mathbf{W}^{(L)})$.

Train Other Layers

Using such a backpropagation process, we will find

$$\frac{\partial S_n}{\partial w_{ij}^{(l)}} = \delta_{n,j}^{(l)} y_{n,i}^{(l-1)}, \text{ where } \delta_{n,j}^{(l)} = \left(\sum_{k=1}^{M_{l+1}} \delta_{n,k}^{(l+1)} w_{jk}^{(l+1)} \right) \phi'(a_j^{(l)}) \tag{8.2.11}$$

for $l = L-1, L-2, \cdots, 1$

Note that when $l = 1$, (8.2.11) becomes

$$\frac{\partial S_n}{\partial w_{ij}^{(1)}} = \delta_{n,j}^{(1)} y_{n,i}^{(0)} = \delta_{n,j}^{(1)} x_{n,i},$$

which is dependent on the input data. With these gradients, all weight matrices $\mathbf{W}^{(L-1)}, \mathbf{W}^{(L-2)}, \cdots \mathbf{W}^{(1)}$ can be identified, and identification of the whole set of weight matrices $\mathcal{W}$ is thus completed.

The simplest but also the most commonly used ANN structure consists of only three layers (Fig. 8.2): an input layer $\mathbf{x}$, a hidden layer $\mathbf{y}$, and an output layer $\mathbf{z}$. Assuming the numbers of neurons in them are P, M, and K, respectively, we have

$$\begin{aligned} a_j &= \sum_{i=1}^{P} w_{ij}^{(1)} x_i,\ y_j = \phi(a_j),\ j = 1, 2, \cdots, M \\ b_k &= \sum_{j=1}^{M} w_{jk}^{(2)} y_j,\ \mathrm{z}_k = \phi(b_k),\ k = 1, 2, \cdots, K \end{aligned} \tag{8.2.12}$$

Following the above general results, we have

$$\begin{aligned} \frac{\partial S_n}{\partial w_{jk}^{(2)}} &= \delta_{n,k}^{(2)} y_{n,j},\ \delta_{n,k}^{(2)} = (z_{n,k} - z_{D,n,k}) \phi'(b_k) \\ \frac{\partial S_n}{\partial w_{ij}^{(1)}} &= \delta_{n,j}^{(1)} x_{n,i},\ \delta_{n,j}^{(1)} = \left(\sum_{k=1}^{K} \delta_{n,k}^{(2)} w_{jk}^{(2)} \right) \phi'(a_j) \end{aligned} \tag{8.2.13}$$

When the steepest descent optimization algorithm is used to train an MLP model, the solved weights are locally optimal, but may not be globally optimal. In fact, the likelihood of trapping at local minima in this case is even higher given the strong nonlinearity of the MLP error function. One commonly used strategy for circumventing the local minima issue is random initialization, in which the MLP is trained many times using different random initial weights. The resulting solutions form an ensemble of MLPs, which can then be used to give an ensemble averaged forecast and associated estimation variance. We will turn to techniques for generating ANN ensembles in Sect. 8.2.4. To improve accuracy and training speed, one may use more sophisticated solvers such as Levenberg-Marquardt, conjugate-gradient, and quasi-Newton method (requires Hessian matrix). So far, Levenberg-Marquardt (see Sect. 2.4.2) is the most widely used algorithm for backpropagation.

The other issue commonly encountered during network training is numerical stability, which can be partly mitigated by adopting more robust algorithms. For example, we can augment the original error function (8.2.3) with a regularization term $S_\alpha(\mathbf{w}) = S(\mathbf{w}) + \alpha \|\mathbf{w}\|_2^2$, where α is a regularization parameter. The regularization problem can be solved either in the deterministic framework (Chap. 3) or in Bayesian framework (see Chap. 4).

Example 8.4 *Daily Streamflow Forecasting Using MLP*
We now develop an MLP model using the same Neches River streamflow dataset considered in Example 8.2. In addition to antecedent streamflow, we would also

like to consider other related predictors, including total daily precipitation and maximum and minimum daily temperatures. The MLP design generally involves two subtasks: input selection and hidden layer structure (e.g., number of layers and neurons). Although in theory both subtasks may be tackled simultaneously using cross-validation and model selection criteria, the number of different combinations can quickly become overwhelming. It is, therefore, not surprising to see heuristic design approaches. For instance, one can first build an ARMAX model by using an automated model selection procedure to select the number of lags for each type of inputs, and then construct an ANN using the same inputs. We choose the following 17 predictors for the base model, largely through correlation analyses and the knowledge gained from building the AR model in Example 8.2,

Precipitation: $P(t-i),\ i = 1,2,3$
Temperature: $T_{\min}(t-1),\ T_{\min}(t-2),\ T_{\max}(t-1),\ T_{\max}(t-2)$,
Streamflow: $Q(t-i), i = 1,\cdots,10$

The target variable is $Q(t)$. Before training, all input and target data are scaled to the range $[-1,1]$ using linear scaling. For example, for the precipitation time series, this can be done by subtracting the minimum from all precipitation and then scaling by the range of the precipitation during the study period. Also, streamflow is transformed using the Box-Cox transformation as we have done in Example 9.2.

The total data records are divided into three parts: training (65 %), validation (15 %), and testing (20 %). Matlab's Neural Network Toolbox is used and the Levenberg-Marquart algorithm is chosen for backpropagation.

The number of hidden neurons is a free parameter that needs to be determined during training. In general, more hidden neurons will ease the training process, but at the price of overfitting. Trial-and-error and cross-validation are often used for determining the number of hidden neurons (Zealand et al. 1999; Maier et al. 2010). Although some researchers pointed out that the number of hidden neurons should be smaller than the total number of input variables (Maier and Dandy 1996), the actual number of hidden neurons appears to be case dependent, especially when the system has multiple outputs. Figure 8.3 plots the corrected AIC, or AIC^C, as a function of the number of hidden neurons. The AIC^C corrects the AIC for finite sample sizes (Hurvich and Tsai 1989)

$$AIC^C = AIC + \frac{2(P+1)(P+2)}{N-P-2}.$$

where P is the number of inputs and N is the number of samples. The plot suggests that 5 is the best number of hidden neurons.

Backpropagation training for the base model took 59 iterations. The resulting NSC (calculated on testing data) is 0.982 and the root-mean-square error

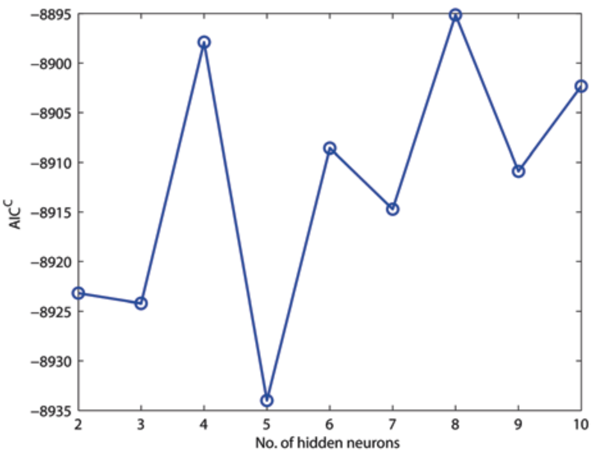


Fig. 8.3 AIC^C as a function of the number of hidden neurons

(RMSE) is about 580 cfs. At this time, a sensitivity analysis may be conducted to formally quantify the effect of exogenous predictors. A number of methods are available for gauging the importance of predictors (Olden et al. 2004). Here, we demonstrate the use of the backward elimination method, in which the predictor variable (or predictor group) is removed one at a time to compare the performance of the reduced model to the base model. Table 8.1 lists the ratios of MSE's between various subsets and the base case, where each subset was formed by removing the corresponding predictor type. The larger the MSE ratio, the greater the importance of a variable to ANN prediction will be. Table 8.1 indicates that all three additional predictor types contribute positively to the predictive power of the base model. The backward elimination method requires developing a new model for each variable selection; alternatively, one may either conduct a global sensitivity study (see Chap. 10) or use the connection weight method suggested by Olden et al. (2004), who showed that the raw input-hidden and hidden-output connection weights in a trained ANN provides the best methodology for accurately quantifying variable importance. An application of the connection-weight method is given in Sun (2013), who studied feasibility of downscaling a satellite data set to predict groundwater level changes.

The ANN performed well in this one-day-ahead streamflow forecast application because the training dataset spans a long period so that a large number of variation patterns are included. Also, the Neches River flow regime is unregulated, which ensures a good correspondence between the predictors and target. If the streamflow is significantly affected by anthropogenic activities (e.g., irrigation or damming), additional information, such as return flow and flow diversion rates, is needed to correct for such impacts.

Table 8.1 Contribution of predictor groups in Example 8.4

Predictor Type	MSE Ratio
Precipitation	1.10
$T_{\max}$	1.11
$T_{\min}$	1.12

8.2.3 *Radial Basis Function Network*

RBFN is a special FNN that has only one hidden layer. In RBFN, radial basis functions (see Sect. 6.1.5) are used as activation functions to connect the input layer to the hidden layer, while the outputs are still modeled as linear combinations of hidden neurons, as in MLP models. Let us assume that the numbers of neurons of the three layers are P, M and K, respectively. An RBFN can be formulated as

$$\begin{aligned} y_j &= \phi_j(\mathbf{x}) = \phi\left(\left\|\mathbf{x} - \mathbf{c}_j\right\|\right), j = 1, 2, \cdots, M \\ z_k &= w_{k1}y_1 + \cdots + w_{kM}y_M = \mathbf{w}_k^T\mathbf{y}, \; k = 1, 2, \cdots, K, \end{aligned} \tag{8.2.14}$$

where $\phi(r)$ is an RBF function, $r = \left\|\mathbf{x} - \mathbf{c}_j\right\|$ is the distance (typically Euclidean) between a training sample $\mathbf{x}$ and *data centers* $\mathbf{c}_k$, and $\mathbf{w}_k$ is the weight vector. Determination of data centers (as structure parameters) will be described in details below. As explained in Sect. 8.1.2, bias terms in the above equations are omitted after preprocessing the data. RBFN (8.2.14) defines a mapping

$$\mathbf{z} = \mathbf{W}^T\boldsymbol{\phi}(\mathbf{x}, \mathbf{C}), \tag{8.2.15}$$

where $\mathbf{W} = [\mathbf{w}_1 \; \mathbf{w}_2 \; \cdots \; \mathbf{w}_K]$ is an $M \times K$ weight matrix as shown in (8.1.31), $\boldsymbol{\phi}=(\phi_1, \phi_2, \cdots, \phi_M)^T$, and $\mathbf{C} = \left\{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_M\right\}$ are data centers. When the output is a scalar variable z, the mapping reduces to a function

$$z = \sum_{j=1}^{M} w_j\phi\left(\left\|\mathbf{x} - \mathbf{c}_j\right\|\right) = \sum_{j=1}^{M} w_j\phi_j(\mathbf{x}). \tag{8.2.16}$$

This is exactly the interpolation or the function approximation problem we considered in Sect. 6.1.5. Now it is viewed and trained as a special ANN. The backpropagation algorithm for training an MLP, of course, can be used to train an RBFN, but there are more effective training methods for this special case. According to (8.2.16), the problem of training an RBFN requires (i) selecting an RBF, (ii) estimating weights, and (iii) optimizing structure parameters.

8.2.3.1 Selecting Radial Functions

A number of RBFs exist in the literature, and some commonly used forms have already been given in Sect. 6.1.5. The Gaussian kernel $\phi(r) = \exp(-r^2 / 2\sigma^2)$ is probably the most widely used RBF. It contains a shape parameter σ that controls the spread of the RBF. The multiquadric RBF $\phi(r) = \sqrt{1 + r^2 / \sigma^2}$ also contains a shape parameter σ. In the rest of this discussion, we denote all shape parameters of RBFs by $\boldsymbol{\sigma} = \left\{\sigma_1, \sigma_2, \cdots, \sigma_M\right\}$. Shape parameters make RBFN more flexible but increase the training effort. Park and Sandberg (1991) showed that RBFNs with a single global spread parameter can approximate any continuous mapping on a compact input domain to arbitrary accuracy, provided that the network has a sufficiently large number of hidden neurons. From the training perspective, the use of a single global spread parameter for all hidden neurons greatly reduces the training effort.

8.2.3.2 Estimating Weights

In RBFN, the input layer and the hidden layer are not connected by weights. Once RBFs are selected, and data centers $\mathbf{C}$ and shape parameters $\boldsymbol{\sigma}$ are determined, all neurons of the hidden layer can be obtained directly by the first equation of (8.2.14) (i.e., $\mathbf{y} = \boldsymbol{\phi}(\mathbf{x})$). For a set of input data $\{\mathbf{x}_n\}$, the corresponding RBF values are $\{\boldsymbol{\phi}(\mathbf{x}_n)\}$. Let the N samples of input data be represented by an $N \times P$ matrix

$$\mathbf{X} = \begin{pmatrix} x_1^1 & \cdots & x_P^1 \\ x_1^2 & \cdots & x_P^2 \\ \vdots & \ddots & \vdots \\ x_1^N & \cdots & x_P^N \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{pmatrix},$$

and the corresponding RBF values be an $N \times M$ matrix

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\phi}(\mathbf{x}_1)^T \\ \boldsymbol{\phi}(\mathbf{x}_2)^T \\ \vdots \\ \boldsymbol{\phi}(\mathbf{x}_N)^T \end{pmatrix}. \tag{8.2.17}$$

The output layer $\mathbf{z}$ is connected to the hidden layer $\mathbf{y} = \boldsymbol{\phi}(\mathbf{x})$ through (8.2.15), which actually is a linear model with multiple outputs as considered in (8.1.30), but here $\mathbf{x}$ is replaced by $\boldsymbol{\phi}(\mathbf{x})$. Therefore, the weight matrix in (8.2.15) can be estimated by replacing $\mathbf{X}$ by $\boldsymbol{\Phi}$ in (8.1.33) and (8.1.34) to get

$$\boldsymbol{\Phi}\mathbf{W} = \mathbf{Z} + \mathbf{E}, \tag{8.2.18}$$

and

$$\hat{\mathbf{W}} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{Z}. \tag{8.2.19}$$

The results show that weights of a RBFN model can be obtained simply by linear regression. In comparison, a nonlinear optimization problem must be solved to estimate weights of an MLP model.

8.2.3.3 Optimizing Structure Parameters

Structure parameters of an RBFN include the number of data centers (M), the locations of data centers $\mathbf{C}$, and shape parameters $\boldsymbol{\sigma}$. Unlike the case of MLP, the bulk effort of most RBFN algorithms focuses on selection of structure parameters, which directly affects the performance of RBFN. Actually, this is the extended inverse problem (EIP) considered in Chap. 7 for physics-based models. According to Eq. (7.6), all structure parameters and weights can be obtained by solving the following min-min problem:

$$\begin{aligned}&\min_{M,\mathbf{C},\boldsymbol{\sigma}} \min_{\mathbf{W}} S(M,\mathbf{C},\boldsymbol{\sigma},\mathbf{W})\\ &S = \frac{1}{N}\sum_{n=1}^{N}\left\|\mathbf{W}^T\boldsymbol{\phi}(\mathbf{x}_n,\mathbf{C},\boldsymbol{\sigma}) - \mathbf{z}_n\right\|_2^2\end{aligned} \tag{8.2.20}$$

Because the inner min solution $\hat{\mathbf{W}}$ has been obtained explicitly in (8.2.19), this min-min problem is reduced to a minimization problem

$$\begin{aligned}&\min_{M,\mathbf{C},\boldsymbol{\sigma}} S(M,\mathbf{C},\boldsymbol{\sigma})\\ &S = \frac{1}{N}\sum_{n=1}^{N} S_n,\quad S_n = \left\|\hat{\mathbf{W}}^T\boldsymbol{\phi}(\mathbf{x}_n,\mathbf{C},\boldsymbol{\sigma}) - \mathbf{z}_n\right\|_2^2 = \sum_{j=1}^{K} e_{n,j}^2\\ &e_{n,j} = \sum_{i=1}^{M} \hat{w}_{ij}\phi_{n,i} - z_{n,j}\end{aligned} \tag{8.2.21}$$

Note that $\hat{\mathbf{W}}$ depends also on the structure parameters. Let the number of data centers, M, be increased gradually from a small initial value; for each fixed M, $\mathbf{C}$ and $\boldsymbol{\sigma}$ can be optimized by a local or a global optimization method introduced in Chaps. 2 and 3. When a gradient-based algorithm is used, according to (8.2.21) the gradients of S_n with respect to all data centers are given by

$$\frac{\partial S_n}{\partial \mathbf{c}_i} = \left(\sum_{j=1}^{K}\hat{w}_{ij}e_{n,j}\right)\phi'_{ni}\frac{\mathbf{x}_n - \mathbf{c}_i}{\left\|\mathbf{x}_n - \mathbf{c}_i\right\|},\ (i = 1,2,\cdots,M). \tag{8.2.22}$$

Especially, when Gaussian kernel is used as the RBF, we have

$$\begin{aligned}
\frac{\partial S_n}{\partial \mathbf{c}_i} &= \left(\sum_{j=1}^{K} \hat{w}_{ij} e_{n,j} \right) \phi_{ni} \frac{\mathbf{x}_n - \mathbf{c}_i}{\sigma_i^2}, \ (i = 1, 2, \cdots, M) \\
\frac{\partial S_n}{\partial \sigma_i} &= \left(\sum_{j=1}^{K} \hat{w}_{ij} e_{n,j} \right) \phi_{ni} \frac{\left\| \mathbf{x}_n - \mathbf{c}_i \right\|_2^2}{\sigma_i^3}, \ (i = 1, 2, \cdots, M)
\end{aligned} \tag{8.2.23}$$

Besides solving (8.2.20), there are other methods for training RBFNs, such as the clustering method (Xu et al. 1993) and the more effective orthogonal least squares method originally introduced by Chen et al. (1991). The clustering method groups the input dataset into appropriate clusters and the centers of these clusters are then used as RBF centers. The orthogonal least squares method adds data centers sequentially until either a predefined MSE is reached on the training set or the maximum number of hidden neurons is exceeded, and all data centers are chosen from the input dataset.

In EWR-related applications, Moradkhani et al. (2004) applied RBFN to develop a one-step-ahead streamflow forecast model; a data clustering algorithm was first used to determine the data centers. Lin and Wu (2011) used RBFN to obtain hourly inflow forecasts to a reservoir during typhoons. They applied a two-step procedure to optimize selection of data centers, which includes the use of both data clustering and orthogonal least squares.

8.2.4 Use ANN for Inverse Solution

Because ANN is a universal function approximator, it can be used to replace a physics-based model, especially when the computational demand associated with such a model is high. The resulting surrogate models are often referred to as response surfaces, emulation models, or metamodels in the literature. Morshed and Kaluarachchi (1998) combined ANN and genetic algorithm (GA) for solving inverse problems related to nonaqueous phase liquids (NAPL) in groundwater aquifers. In their work, ANN was used to approximate the inverse relationship between parameters of a multiphase model and pollutant concentration observations. Mirghani et al. (2012) used the combination of ANN and GA to solve a contaminant source identification problem.

Given a forward mapping $\mathcal{M}$ from parameter space to state space

$$\mathbf{u} = \mathcal{M}(\boldsymbol{\theta}), \tag{8.2.24}$$

we would like to find a metamodel $\mathcal{F}$ that converges to the original $\mathcal{M}$ in the mean square sense. Here, it is assumed that the original model $\mathcal{M}$ is a physics-based model and is computationally demanding to run. The structure of $\mathcal{F}$ depends on the underlying metamodeling technique used. ANN represents one of the common choices. In Chap. 10, we will introduce other metamodeling methods in the context of uncertainty propagation.

The basic ANN metamodeling procedure consists of three major steps:

1. Define the range of parameters and generate a large number of input patterns
2. Run the forward physics-based model on each training pattern and collect model outputs at user-designated observation points
3. Develop, train, and test an ANN model using the dataset obtained from the previous two steps

After obtaining the ANN metamodel, we can couple it with a global optimization solver, such as GA (see Chap. 3), to perform parameter estimation or experimental design. Depending on the underlying physics-based model, the metamodeling approach can offer improvement in computational efficiency by several orders of magnitudes. In the following, we use a simple example to illustrate how the combined ANN-GA metamodeling approach works.

Example 8.5 *Parameter Estimation Using MLP-GA*
We would like to develop an MLP to approximate the following sinusoidal function

$$f(x) = c_1 \sin(c_2 x) + c_3 \sin(c_4 x),$$

for which the parameter vector is $\mathbf{c} = (c_1, c_2, c_3, c_4)$, and $x \in (0, 2\pi)$. All parameters are assumed to be uniformly distributed. The bounds of the parameters are given in Table 8.2 below.

The parameter space is first sampled uniformly to generate 200 realizations of training patterns. The target data consist of "observations" of $f(x)$ from 63 points. The "true" parameter vector that generated those model observations is $\mathbf{c} = (0.25, -0.51, 0.23, 1.1)$, and is not included in training. In the second step, a single-hidden-layer MLP is developed. The size of the input data to the MLP is 4×200 (i.e., one row for each parameter) and the size of the output data is 63×200 (i.e., one row for each observation point). The data are divided into three parts for training, validation, and testing, respectively. The number of hidden neurons is set

Table 8.2 Parameter bounds used in Example 8.5

Parameter	Lower Bound	Upper Bound
c_1	0.1	0.5
c_2	−1	0
c_3	0	0.5
c_4	0.5	1.5

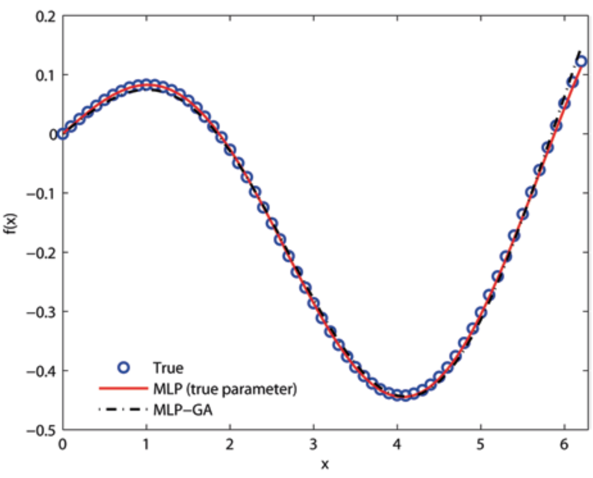


Fig. 8.4 Metamodeling using MLP, in which open circles correspond to training output data, the solid red line is simulated by MLP using "true" parameters, and the dash line corresponds to MLP outputs obtained using model parameters estimated from GA

to 15 through trial and error. The MLP model is trained following the same steps as detailed in Example 8.4.

After the MLP model is trained, it can be used to simulate model output for any input vector within the parameter bounds. For testing, we passed the true parameter vector to the trained metamodel to assess its performance. Figure 8.3 compares the "true" model outputs (open circles) with those from the MLP (solid line). A good match between the two can be observed.

In reality, we can only access observations. Thus, the MLP model is combined with a GA solver to estimate model parameters. For each trial parameter vector, the objective function is the MSE between the MLP solution and actual observations (i.e., circles in Fig. 8.4). The goal of GA is to find a set of global optima that minimize the objective function. The final parameter set returned by GA is $\hat{\mathbf{c}} = (0.28, -0.56,\ 0.26,\ 1.05)$. Figure 8.4 shows that the MLP-GA solution matches the true solution well except near the right end of the curve. We used a relatively dense observation grid in this example. Readers may try to vary observation density to see the impact on final results. ■

Although the actual model used in Example 8.5 is not computationally expensive, the example highlights the potential of MLP-GA for solving inverse problems. In contaminant source identification problems requiring identification of both contaminant source locations and release histories, a popular strategy is to solve a min-min problem (see also Sect. 7.1.2). Because the minimization requires running the forward model many times, a metamodel may be trained using different source locations as inputs. Another possible use of metamodeling is to replace physics-based models in decision support systems, which often requires the capability to generate model predictions under different scenarios (e.g., climate change). The following example shows a prototype.

Example 8.6 *Nonpoint-Source Pollution*
In this example, an RBFN model is used to approximate a physics-based model originally built using the Soil and Water Assessment Tool (SWAT), which is a continuous hydrologic simulation tool developed to quantify the impact of land management practices on surface water quality in large watersheds (Gassman et al. 2007). The study area, Arroyo Colorado Watershed (~1800 km^2), is located in Lower Rio Grande Valley of south Texas, USA, near the Gulf of Mexico coast. It is an intensively cultivated, canal-irrigated watershed. The SWAT model was developed for the study area to model the effect of watershed best management practices (Kannan et al. 2010). The dominant land use categories in the watershed are agriculture (43 %) and rangeland (34 %). From the training performance perspective, it is generally better to develop a separate RBFN for different pollutants of interest. As an application example, here the RBFN model is specifically developed to predict one-month-ahead total nitrogen (TN) loading, which includes nitrate, nitrite, ammonia, and organic nitrogen in the water. Unlike the data-driven streamflow forecast models shown in Example 8.4, in this case the targets of RBFN model are obtained using forward SWAT model runs. In other words, the data-driven modeling is taken place in the model space of a calibrated SWAT model.

As part of the RBFN development, a backward elimination selection procedure was applied to choose predictors. The selected final set of input variables for the one-month-ahead TN loading prediction include

$$\begin{aligned}
&P_{ha}(t-1), P_{md}(t-2),\\
&T_{\min,ha}(t-1), T_{\max,ha}(t-1), T_{\min,mc}(t-1), T_{\max,mc}(t-1),\\
&Q_{ha}(t-1), Q_{ha}(t-2),\\
&TN(t-1), TN(t-2)
\end{aligned}$$

where the subscripts *ha*, *md*, and *mc* are abbreviations of different observation stations, and all other symbols are as defined in Example 8.3. The final RBFN for TN loading prediction consists of 30 hidden neurons. Note that the number of hidden neurons used by RBFN is typically larger than that for a single-hidden-layer MLP.

Figure 8.5 plots the TN loading simulated by the RBFN for both the training and testing periods, and the boundary of the two periods is indicated by the vertical line on the plot. The training period includes a sufficient number of variation patterns, including a high loading peak near the end of 2002. The trained RBFN model gives satisfactory performance. The NSE calculated on the testing set is 0.78. During scenario analyses, forcing data under different climate change scenarios may be created by downscaling from climate models or using weather generators (Schoof and Pryor 2001). The trained RFBN model can be used as a surrogate of the physics-based SWAT model to perform continuous loading prediction. In addition to forcing data, metamodels may also be developed for fast parametric sensitivity studies by using SWAT parameters as inputs and SWAT output as target variable. More details

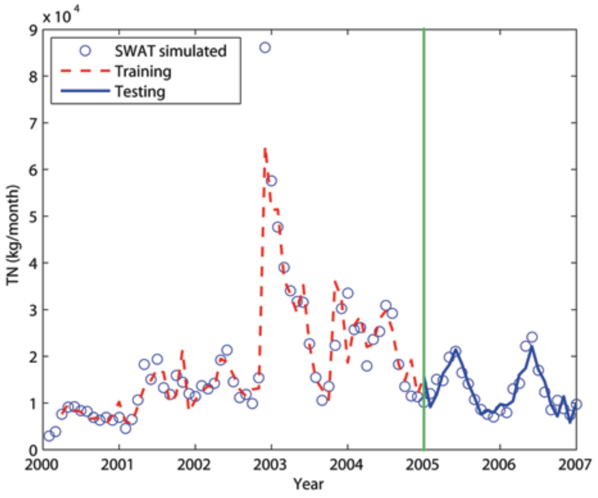


Fig. 8.5 RBFN metamodel for TN output prediction, using SWAT results (circles). Training and testing periods separated by vertical line

of this case study, including the use of metamodels in a decision support framework, can be found in Sun et al. (2014a). ■

When using metamodels for parameter estimation or experimental design, at least three types of errors are involved. The first is model approximation error that is introduced by replacing a physics-based model with a metamodel, the second is model fitting error arising during metamodel training, and the third is model structure error associated with the original physics-based model itself. Several possible strategies exist to mitigate the first two sources of error. One may increase the number of training patterns used for metamodeling, provided that the effort required for generating those training patterns is still significantly less than that of using the physics-based model for the full design. Another possibility is to switch to the original physics-based models for the part of design requiring high fidelity. For example, in model-based system failure probability assessment, Li et al. (2011) applied a hybrid method that utilizes a metamodel in most part of the parameter space, but uses the full model in the region surrounding the failure mode (i.e., parameter values that can lead to system failure). Such method relies on one's ability to identify parameter regions that the developed metamodel is likely to fail. Yet another possibility is to treat the desired reliability of the developed metamodel as a constraint and solve a constrained programming problem. More details on probabilistic metamodeling will be given in Chap. 10.

8.2.5 ANN Ensembles

Typical ANNs do not yield confidence intervals of estimates directly. The most popular method for estimating confidence intervals is by combining outputs of multiple neural networks to form an ANN ensemble,

$$\hat{\mathbf{z}} = \sum_{l=1}^{L} \beta_l \mathbf{f}_l(\mathbf{x}), \tag{8.2.25}$$

where L is the number of different ANNs in an ensemble and β_l is the weight assigned to the l-th ensemble member. From (8.2.25), the statistics of estimation can be estimated. Methods for ensemble generation fall into one of the four categories (Sharkey 1999): (i) random initialization, (ii) variation of network topologies, (iii) variation of the training algorithm, and (iv) random permutation of training datasets. The first category of methods mainly pertains to ANNs trained via backpropagation, for which different initial guesses of weights generally lead to different final weights because of the inherent nonlinearity of the error function. Utilizing this "unstableness" side effect, a number of ensembles can be generated and trained, each using different initial weights. Category (ii) and (iii) are less common because they involve networks of different structures, making it less straightforward to estimate weights for individual ensemble members. The last category of methods, originated from standard statistical techniques, has received the most attention so far. We will briefly go through two of such statistical techniques, bagging and boosting.

8.2.5.1 Bagging

Bagging is short for *bootstrap aggregation*, which was originally introduced by Breiman (1996) for improving stability of learning algorithms when training data are limited. The core concept behind bagging is relatively simple. In the bootstrapping step, the training dataset is randomly sampled with replacement to form L "realizations" of the original dataset. In the aggregation step, all realizations are used separately to train the ANN to form an ensemble of models, from which the ensemble performance can be evaluated using its statistics (e.g., mean and variable). On average, each bootstrapped training set contains 63.2 % of the original training set and the remaining samples are replicates, due to the sampling-with-replacement procedure (Breiman 1996).

Bagging essentially provides a nonparametric method for approximating posterior mean which, as we have learned in Chap. 4, often leads to a reduction in the MSE. In this case, each ensemble member is an individual model and is assigned uniform prior probability. Each model is trained using the same set of training data. The empirical ensemble statistics then reflects the new information in training data in a Bayesian sense. The same empirical Bayesian concept is behind ensemble-based data assimilation and Bayesian model averaging. In particular, we will see in Chap. 9 that data assimilation may be used to sequentially update the weight of each ensemble member as new information becomes available.

So what makes bagging work? Like the random initialization method, bagging also exploits nonlinearity of the loss function and unstableness of the underlying base training algorithm (e.g., backpropagation) to generate diverse ensemble members. In other words, bagging expects that a small change to a training set can cause sensible changes in the trained model, which is true when the sample size is limited and when many local minima exist. Hansen and Salamon (1990) showed that the necessary and sufficient condition for the performance of an ensemble of models to be more accurate than the best of its individual members is that all ensemble models are accurate and diverse. However, the condition does not say how many ensemble members are needed. In practice, the number of ensemble members is usually found by trial-and-error, a process that can be computationally demanding for large-size training data.

Bagging involves a passive sampling method in the sense that the algorithm does not use feedbacks from the likelihood to inform the sampling process (i.e., update weights of ensemble members). The method introduced below represents a different paradigm.

8.2.5.2 Boosting

Boosting, originally described in a seminal paper by Schapire (1990) and later in Freund and Schapire (1996), is a general algorithm for adaptively generating a series of predictive models. Each time, a new predictive model is conditioned on the performance of previous ones, and the sampling weights of the original training data are adjusted. Training data that receive poor prediction by the current predictor will have higher probability of getting sampled in a new training set than those correctly predicted. In boosting, the ensemble members are generated serially, instead of all at once as in bagging. A boosting algorithm is provided below.

For a given training dataset consisting of N pairs of input and target data $\{(\mathbf{x}_i, \mathbf{z}_i)\}_{i=1}^N$, do

1. Initially, assign equal weight to each training sample, $w_i = 1/N$
2. For $m = 1 : M$, where M is the number of iterations, do

- Generate a new training dataset by sampling with replacement from the original dataset using sampling weights $\{w_i\}$
- Construct and train an ANN model, $\mathbf{f}_m$, using the training set
- Calculate the distance or loss function for each sample as

$$L(d_i) = L\left(\left\|\mathbf{z}_i - \mathbf{f}_m(\mathbf{x}_i)\right\|\right), \; i = 1, \cdots, N \tag{8.2.26}$$

where the commonly used loss function $L(d_i)$ are

$$\begin{aligned} \text{Linear: } & L(d_i) = d_i / d_{\max} \\ \text{Exponential: } & L(d_i) = 1 - \exp(-d_i / d_{\max}) \\ \text{Euclidean: } & L(d_i) = (d_i / d_{\max})^2 \end{aligned}$$

in which the normalization constant $d_{\max} = \sup\{d_i\}$ is the upper bound of all $d_i (i = 1, \cdots, N)$.

- Calculate the weighted average loss obtained by $\mathbf{f}_m$ as

$$\bar{\varepsilon}_m = \sum_{i=1}^{N} w_i L(d_i)$$

- Update the weight of each training data according to

$$w_i = \frac{1}{C} w_i \alpha_m^{(1-L(d_i))} \tag{8.2.27}$$

where $\alpha_m = \bar{\varepsilon}_m / (1 - \bar{\varepsilon}_m)$, *m* is looping index, and *C* is a normalization factor to ensure the sum of $\{w_i\}$ is 1.

The above boosting algorithm for regression is known as AdaBoost.R2 in the literature (Drucker 1997). As mentioned previously, the main difference between boosting and bagging is that boosting goes one step further by progressively adjusting training data sampling strategy through (8.2.27) such that more focus is placed on poorly performing data in future iterations. By design, AdaBoost does not explore unstableness of the underlying training algorithm when generating ensembles. On the other hand, by progressively handing more difficult fitting problems to the base training algorithm, AdaBoost exposes itself to overfitting, especially when the data noise level is high. The choice of loss function can also affect the robustness of AdaBoost.

A number of studies have been carried out to compare performance of random initialization, bagging, and boosting. In the context flood frequency analysis, Shu and Burn (2004) reported that the average generalization ability of ANN ensembles was always better than that of single models, regardless of the ensemble method used; among the ensemble averaging methods tested by them, the performance of bagging was better than that of random initialization, and AdaBoost outperformed bagging. In-depth discussion of these ensemble methods can be found in Zhou (2012) and Hastie et al. (2009).

8.3 Support Vector Machine and Relevance Vector Machine

8.3.1 Support Vector Machine Regression

ANN models may involve considerable subjectivity when it comes to model structure selection and training. As we mentioned at the beginning of this chapter, a fundamental question underlying ANNs and any other statistical learning algorithm is related to *generalization capability*: how to design and train a data-driven model such that it is guaranteed to deliver similar performance on data not seen during training. A measure of model prediction discrepancy can be defined as

$$E_p = \int \left| z - f(\mathbf{x};\boldsymbol{\theta}) \right| p(\mathbf{x}, z) d\mathbf{x} dz, \tag{8.3.1}$$

where $\mathbf{x}$ contains all input variables, z is the true system response (assumed to be a scalar), $f(\mathbf{x};\boldsymbol{\theta})$ is the model-predicted response, $\boldsymbol{\theta}$ are parameters associated with the trained model, $|\cdot|$ is a distance measure, and $p(\mathbf{x}, z)$ is the joint PDF of input and output variables. Eq. (8.3.1) is often referred to as the *true prediction risk* in statistical learning theory (Burges 1998; Vapnik 1995). Because neither the true system nor the joint PDF $p(\mathbf{x}, z)$ is known exactly, (8.3.1) has only theoretical meaning. In practice, the system response is observed only at discrete points. Thus, discrepancy or training error is approximated by some empirical loss function, or *empirical prediction risk* E_e. A commonly used measure of the empirical risk is MSE. Training algorithms that seek to minimize empirical risks are said to follow the *empirical risk minimization* principle. For instance, the backpropagation algorithm in MLP minimizes the MSE. We see that the empirical risk minimization principle is essentially equivalent to Criterion (C-1) underlying the CIP (Chap. 2).

If the training dataset is large and the values are accurate, MSE provides a good estimate of true prediction error. On the other hand, if the sample size is limited and noisy, relying solely on MSE minimization may weaken a model's generalization capability. Another issue with the empirical risk minimization approach is that the process does not give direct control on model structure complexity; instead, ad hoc procedures (e.g., cross-validation) must be repeated on all possible model structures and the optimal structure is selected based on one or more model selection criteria, such as the AIC or BIC.

8.3.1.1 Bound of Prediction Risk

Is it possible to establish a relationship between the size of training data and generalization capability of a trained model? Or better, can we incorporate such knowledge in the training process to prevent overfitting? These questions motivated the development of the support vector machine (SVM), which grew out of the statistical learning theory pioneered by Vapnik and Chervonenkis (1974) and later formalized by Vapnik and his co-workers at the former AT&T Bell Laboratory (Vapnik 1995). In statistical learning theory, the term *learning machine* refers to data-driven algorithms that can be used to learn and generalize from training data.

The starting point of SVM is to establish a bound of the true prediction risk, E_p, by using two types of error

$$E_p(\boldsymbol{\theta}) \le E_e(\boldsymbol{\theta}) + E_s(\boldsymbol{\theta}), \tag{8.3.2}$$

where the fitting error $E_e(\boldsymbol{\theta})$ is fixed for given model parameters $\boldsymbol{\theta}$, and $E_s(\boldsymbol{\theta})$ represents the capacity of a statistical learning algorithm to learn any dataset without error. In SVM terminology, $E_s(\boldsymbol{\theta})$ provides a *capacity control* of model complexity.

Functional forms of $E_s(\boldsymbol{\theta})$ will be provided later in this section when we describe SVM training. The significance of (8.3.2) is that it establishes an upper bound of generalization error. Minimization of this upper bound calls for minimization of two objective functions: the empirical risk (fitting error) and the capacity control. Generally speaking, increasing model complexity tends to reduce the fitting error, but at the risk of overfitting, and vice versa. This is analogous to the concept illustrated in Fig. 7.2 in the context of distributed parameter models. In statistical learning theory, minimization of the right-hand side of (8.3.2) is referred to as the *structural risk minimization* principle.

The upper bound of prediction risk is sometimes given in a multiplicative form instead of additive form (Cherkassky and Mulier 2007),

$$E_p(\boldsymbol{\theta}) \le \gamma E_e(\boldsymbol{\theta}), \tag{8.3.3}$$

in which $\gamma \ge 1$ is a penalization factor. A definition of γ is given by Cherkassky and Mulier (2007)

$$\gamma = \left(1 - \sqrt{\kappa - \kappa \ln \kappa + \ln N / 2N}\right)^{-1}, \tag{8.3.4}$$

where $\kappa = v / N$, v is called the Vapnik-Chervonenkis (VC) dimension; and N is the sample size. The VC dimension is a measure of model complexity of a model class. Larger VC is usually indicative of more complex models (Vapnik 1995). Thus, giving a set of models, the one that yields the smallest γ should be chosen. Eq. (8.3.4) shows that $\gamma \to 1$ as $N \to \infty$, meaning that the empirical risk converges to true prediction risk when sample size becomes large.

Based on the structural risk minimization principle, SVM seeks to improve a trained model's generalization performance by striking a balance between the empirical risk and capacity control. SVM has been used widely for both classification and regression problems. We will mainly focus on the latter in this book.

8.3.1.2 Support Vector Regression (SVR)

Our starting point is the general regression problem involving a single target variable

$$f(\mathbf{x};\mathbf{w}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + w_0, \tag{8.3.5}$$

where $\boldsymbol{\phi}(\cdot)$ is a set of basis functions and the unknown parameters are becomes the weight vector $\mathbf{w}$ and the intercept term w_0 in this case. In SVM, $\boldsymbol{\phi}(\cdot)$ is a mapping that transforms data from the input space to a feature space in which linear regression can be performed, but the feature space may have high or even infinite dimension. In other words, the dimension of $\mathbf{w}$ is only implicitly defined. Fortunately, as we will show shortly, the actual form of $\boldsymbol{\phi}(\cdot)$ does not need to be known explicitly because all we need is a kernel function that describes the inner product of $\boldsymbol{\phi}(\cdot)$ in the feature space,

$$k(\mathbf{x},\mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{x}'), \tag{8.3.6}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are two data points in the input space. For our purpose, it suffices to know that feature space is an inner product space. The idea of avoiding explicit manipulation of $\boldsymbol{\phi}(\cdot)$ by working with their inner products in a feature space is known as the "kernel trick," or kernel substitution. Kernel trick is used broadly in density estimation, machine learning, and dimension reduction. For example, we have already seen applications of kernels in the KPCA algorithm introduced in Sect. 6.4. Kernel trick will also appear in several other machine learning algorithms to be introduced later in this chapter.

The necessary and sufficient condition for a function to be a valid kernel is that the Gram matrix, whose elements are obtained by applying the same kernel on all combinations of input data, is positive semidefinite (Bishop 2006). The most commonly used kernel functions are Gaussian kernel and polynomial kernel, which were already given separately in Sect. 6.1 and Sect. 6.4.

Now let us return to the regression problem (8.3.5). As we have learned in Chap. 3, with the training data pairs in (8.2), the unknown weights $\mathbf{w}$ can be estimated by solving the following minimization problem

$$\min_{\mathbf{w}} \left\{ \frac{1}{2} c \sum_{i=1}^{N} \left| z_i - f(\mathbf{x}_i;\mathbf{w}) \right|^q + \frac{1}{2} \left\| \mathbf{w} \right\|_2^2 \right\}, \tag{8.3.7}$$

where $q = 1$ or 2 is the power, and N is the number of training data pairs. Note that the objective function in (8.3.7) is defined according to structural risk minimization principle, in which the first term corresponds to fitting error E_e and the second term can be considered a form of capacity control E_s. The parameter c controls the trade-off between fitting error and model complexity. At this point, readers may recognize the linkage between the structural risk minimization principle and Tikhonov regularization (see Sect. 3.3), both seeking to avoid overfitting via an additional penalty term, and parameter c can be considered the inverse of regularization parameter in Tikhonov regularization. Both methods also involve the solution of a bi-criterion minimization problem that we have discussed extensively in Chap. 3. By the same token, we see that (8.3.7) is closely related to the regularized objective function, (8.1.24), used in shrinkage methods, only that in (8.3.7) the penalty parameter appears in front of the fitting error term.

Direct calculation of the terms in (8.3.7) would suggest that all training data pairs are used. However, not all of these data are equally important and it may offer some advantage to treat them differently. Toward this end, Cortes and Vapnik (1995) used an ε-insensitive cost function to replace the sum-of-squares error term in (8.3.7) by introducing two sets of slack variables, $\xi_i \geq 0$ and $\xi_i^* \geq 0$

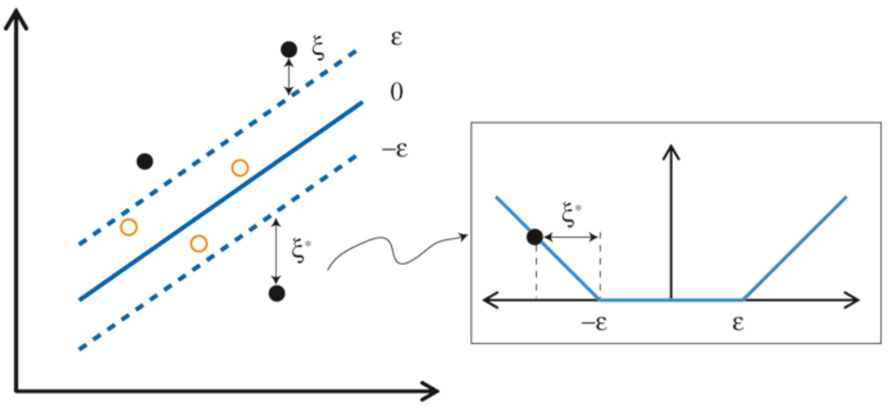


Fig. 8.6 Illustration of the ε-insensitive cost function, in which circles are data points and only points lie outside and on the boundary of the ε-tube contribute to the cost. The inset shows how a deviation term is measured

$$\xi_i = \begin{cases} 0 & \text{if } |d_i| \le \varepsilon \\ d_i - \varepsilon, & \text{otherwise} \end{cases}, \quad \xi_{\text{i}}^* = \begin{cases} 0 & \text{if } |d_i| \le \varepsilon \\ -d_i - \varepsilon, & \text{otherwise} \end{cases}, \tag{8.3.8}$$

where $d_i = z_i - f(\mathbf{x}_i;\mathbf{w})$, $i = 1,\cdots,N$; and parameter ε specifies the amount of deviations that can be tolerated. The concept behind the ε-insensitive cost function is further illustrated in Fig. 8.6, in which the residual space is split into two regions: one inside the dash lines and the other outside. The part that is inside the dash lines is referred to as the ε-*tube*. Positive deviations are denoted as ξ and negative deviations as ξ^*. Only those points that lie outside of the ε-tube are used by SVR to train the model. This way, the SVM gives sparse solutions while mitigating the issue of overfitting. The ε-insensitive cost function (8.3.8) can also be interpreted as a means for achieving robustness because of the penalty exerted on outliers.

After substituting (8.3.8) into (8.3.7) and setting $q = 1$, we obtain a constrained optimization problem for ε-insensitive SVR or ε-SVR (Vapnik 1998, 1995):

$$\min\left\{ c\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right) + \frac{1}{2}\|\mathbf{w}\|_2^2 \right\} \tag{8.3.9}$$

subject to

$$\begin{cases} z_i - \mathbf{w}^T\boldsymbol{\phi} - w_0 \le \varepsilon + \xi_i \\ -(z_i - \mathbf{w}^T\boldsymbol{\phi} - w_0) \le \varepsilon + \xi_i^*, i = 1,\cdots,N \\ \xi_i, \xi_i^* \ge 0 \end{cases} \tag{8.3.10}$$

In ε-SVR, the constant $c > 0$ determines the threshold to which deviations larger than ε are tolerated. An illustration of deviation measurement is shown in the inset of Fig. 8.6.

The constrained quadratic programming problem (8.3.9)–(8.3.10) can be solved more easily in its dual formulation by incorporating the constraints into a Lagrangian

$$
\begin{aligned}
L = c\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right) + \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N}\left(\eta_i\xi_i + \eta_i^*\xi_i^*\right) \\
-\sum_{i=1}^{N}\alpha_i\left(\varepsilon + \xi_i - z_i + \mathbf{w}^T\boldsymbol{\phi} + w_0\right) \\
-\sum_{i=1}^{N}\alpha_i^*\left(\varepsilon + \xi_i^* + z_i - \mathbf{w}^T\boldsymbol{\phi} - w_0\right)
\end{aligned}
\qquad (8.3.11)
$$

where $\eta_i, \eta_i^*, \alpha_i$, and α_i^* are nonnegative Lagrange multipliers. Substituting (8.3.5) into (8.3.11) and taking partial derivatives with respect to the primal variables ($\mathbf{w}$, w_0, ξ_i, and ξ_i^*), we arrive at the following dual optimization problem

$$
\begin{aligned}
\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*}\Bigg\{-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\alpha_i - \alpha_i^*\right)\left(\alpha_j - \alpha_j^*\right)k\left(\mathbf{x}_i,\mathbf{x}_j\right) \\
-\varepsilon\sum_{i=1}^{N}\left(\alpha_i + \alpha_i^*\right) + \sum_{i=1}^{N} z_i\left(\alpha_i - \alpha_i^*\right)\Bigg\},
\end{aligned}
\qquad (8.3.12)
$$

subject to the linear constraints

$$
\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0,\ 0 \le \alpha_i, \alpha_i^* \le c. \qquad (8.3.13)
$$

From (8.3.12), it can be seen that the kernel function, $k(\mathbf{x}_i, \mathbf{x}_j)$, arises naturally as part of the dual formulation and the dependence on $\boldsymbol{\phi}$ disappears. The original regression problem (8.3.5) can now be expressed in terms of the kernel functions. The solution to the dual problem (8.3.12)–(8.3.13) is

$$
f(\mathbf{x}) = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)k\left(\mathbf{x}_i,\mathbf{x}\right) + w_0, \qquad (8.3.14)
$$

which is also the solution of the ε-SVR problem. In (8.3.14), the values of α_i and α_i^* are zero for all points inside the ε-tube. Those training data points for which either $\alpha_i \neq 0$ or $\alpha_i^* \neq 0$ are called *support vectors*. The larger the tolerance to deviations, the fewer support vectors are needed to estimate $f(\mathbf{x})$. The intercept term can be estimated using one of the support vectors (say, the *m*-th support vector),

$$w_0 = z_m - \varepsilon - \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) k(\mathbf{x}_i, \mathbf{x}_m).$$

This completes the derivation of the standard ε-SVR.

In ε-SVR, model complexity is controlled by adjusting parameters ε and *c*, as well as hyperparameters of the chosen kernel function. In general, these parameters have to be determined through separate methods. Cross-validation, for example, is commonly used and is implemented by many SVM packages mentioned at the end of this subsection. However, cross-validation becomes computationally intensive when the number of trial values of ε, *c*, and the kernel function hyperparameters is large. Therefore, heuristic methods exist for the selection of ε and *c*. Interested readers may refer to Mattera and Haykin (1999) and Cherkassky and Ma (2004) for setting ε and *c* using analytical formulae.

When $q = 2$, the resulting SVR is called the least-squares SVR, or LS-SVR. In this case, the minimization problem becomes

$$\min_{\mathbf{w}} \left\{ \frac{1}{2} c \sum_{i=1}^{N} e_i^2 + \frac{1}{2} \|\mathbf{w}\|_2^2 \right\}, \tag{8.3.15}$$

in which a new set of unknowns is introduced to represent the fitting error,

$$e_i = z_i - \mathbf{w}^T \boldsymbol{\phi} - w_0. \tag{8.3.16}$$

The Lagrangian corresponding to the minimization problem is obtained by incorporating the constraint (8.3.16)

$$L = \frac{1}{2} \|\mathbf{w}\|_2^2 + \frac{1}{2} c \sum_{i=1}^{N} e_i^2 - \sum_{i=1}^{N} \alpha_i (\mathbf{w}^T \boldsymbol{\phi} + w_0 + e_i - z_i), \tag{8.3.17}$$

where α_i are Lagrange multipliers. After taking partial derivatives with respect to the primal variables (**w**, w_0, e_i, and α_i), a linear system of equations is obtained, from which the unknowns α_i and w_0 can be solved for. The final solution is given in the form of linear combination of kernel functions

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + w_0. \tag{8.3.18}$$

Thus, unlike the dual optimization problem in ε-SVR, the dual problem in LS-SVR is linear and easier to solve. The disadvantage of the LS-SVR, however, is that its solution is no longer sparse because of the replacement of the ε-insensitive cost function by the sum of squares cost function. In this sense, the difference between

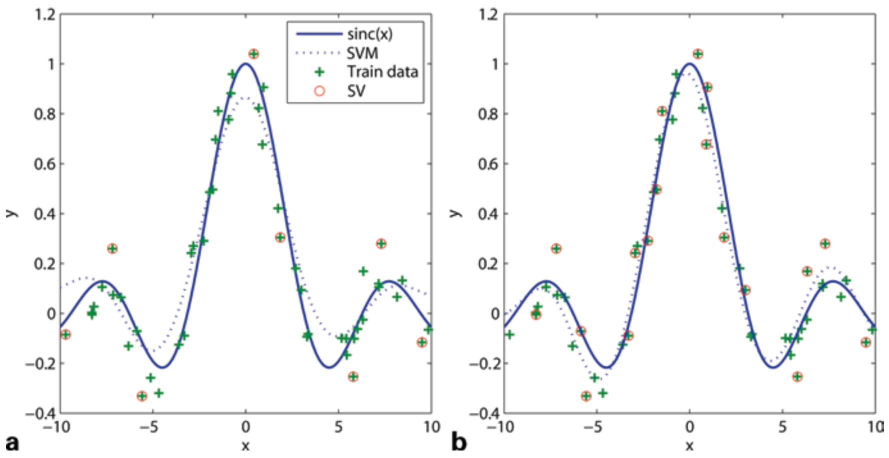


Fig. 8.7 Effect of ε on model complexity: **a** ε=0.2 (8 support vectors). **b** ε=0.1 (18 support vectors)

ε-SVR and LS-SVR is analogous to that between Lasso and ridge regression: one yields sparse weights and the other does not. More details of the LS-SVR method can be found in Basak et al. (2007).

Example 8.7 *Application of ε-SVR*
In this example, we illustrate the effect of loss tolerance parameter ε on the complexity of resulting models. Let us consider the *sinc* function that is commonly used for benchmarking statistical learning machines in the literature (Cherkassky and Ma 2004). Our test dataset is generated by using the following equation

$$y = \text{sinc}(x),$$

which is "contaminated" by a measurement noise $\mathcal{N}(0, \sigma^2)$. For this example, the range of *x* is $[-10, 10]$ and is discretized uniformly into 0.02 intervals. The standard deviation of noise term σ is 0.1. The training data are selected randomly from the test dataset. Figure 8.7a shows the original function sinc(*x*) (solid line) and the 50 data (+) selected for training. The kernel function used is the Gaussian kernel. The SVM toolbox LibSVM (Chang and Lin 2011) is used to solve this problem.

In the first experiment, ε is set to 0.2. The value of *c* is set to 1.5 and the Gaussian kernel spread parameter is set to 0.12. Figure 8.7a shows the ε-SVR predictions. The number of support vectors selected by the algorithm is 8 (circles), and dotted line represents prediction given by the trained ε-SVR model for the whole *x* range. In the second experiment, ε is set to a smaller value of 0.1 and all other parameters are fixed. The results are shown in Fig. 8.7b. Not surprisingly, the sparseness is reduced and the number of support vectors is increased to 18 in the second case. The RMSE values corresponding to the two cases are 0.0874 and 0.0574, respectively. The results indicate that ε-SVR can yield relatively sparse solutions while giving acceptable performance. ■

The SVR has gained popularity in many fields including the EWR (Ghosh and Mujumdar 2008; Asefa et al. 2006; Kalra and Ahmad 2009). Although the SVR can be formulated as a standard constrained optimization problem, special solvers are now provided in several open-source packages, including LibSVM (C++ with Matlab interface) (Chang and Lin 2011), Pegasos (C++) (Shalev-Shwartz et al. 2011), and CVXOPT (Python) (Andersen et al. 2011). More comprehensive discussion of SVR algorithms and solution methods can be found in a number of literature surveys and monographs (Smola and Schölkopf 2004; Cherkassky and Mulier 2007; Bottou and Lin 2007).

8.3.2 Relevance Vector Machine

SVM represents a powerful data-driven method for learning nonlinear models; however, it has several issues and can be further improved. A noteworthy extension of SVM is the relevance vector machine (RVM), which was originally introduced by Tipping (2001). By design, RVM seeks to address several issues and deficiencies of SVM, such as (i) SVM predictions are not probabilistic and, thus, confidence intervals are not readily accessible, and (ii) ad hoc procedures (e.g., cross-validation) are needed for hyperparameter selection (Tipping 2001). Unlike the deterministic ε-insensitive cost function used in ε-SVR, RVM estimates the weights $\mathbf{w}$ in the Bayesian framework that enforces prior distribution on $\mathbf{w}$ to control the model complexity and, thus, avoids overfitting.

Given a set of training data pairs, $\{(\mathbf{x}_i, z_i)\}_{i=1}^{N}$, the following relationship exists between the observation, z_i, of a scalar target variable z and its model $f(\mathbf{x}_i)$

$$z_i = f(\mathbf{x}_i) + \varepsilon_i, 1 \le i \le N \tag{8.3.19}$$

where ε_i are zero-mean i.i.d. noise, $\varepsilon_i = \mathcal{N}(0, \sigma^2)$, and observations z_i are assumed to be independent of each other. The solution of RVM has the general form of linear regression (8.1.35):

$$f(\mathbf{x}) \approx f(\mathbf{x}; \mathbf{w}) = \sum_{i=0}^{M} w_i \phi_i(\mathbf{x}), \tag{8.3.20}$$

which contains $M+1$ free parameters.

Assume that **w** is a multiGaussian random variable and all its components are independent, the prior distribution of **w**, $p(\mathbf{w})$, can be written as

$$p(\mathbf{w} \mid \boldsymbol{\alpha}) \sim \prod_{i=0}^{M} \mathcal{N}\left(w_i \middle| 0, \alpha_i^{-1}\right), \tag{8.3.21}$$

in which α_i denotes the reciprocal of the variance of w_i (also known as the precision parameter). Eq. (8.3.21) involves M+1 hyperparameters $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \cdots, \alpha_M)^T$ that need to be determined.

To estimate the hyperparameters, RVM uses the hierarchical Bayesian approach. Like in MCMC (Sect. 4.3) and hierarchical Bayesian estimation (Sect. 7.4.2), the prior of $\boldsymbol{\alpha}$ is assumed to follow the Gamma distribution and take the following product form

$$p(\boldsymbol{\alpha}) = \prod_{i=0}^{M} \text{Gamma}\left(\alpha_i \middle| a, b\right), \tag{8.3.22}$$

where the Gamma distribution is defined as

$$\text{Gamma}\left(\lambda \middle| a, b\right) = \frac{b^a}{\Gamma(a)} \lambda^{a-1} e^{-b\lambda}.$$

Similarly, we can write the likelihood function in a product form,

$$p\left(\mathbf{z} \middle| \mathbf{w}\right) \sim \prod_{i=1}^{N} p\left(z_i \middle| \mathbf{w}\right). \tag{8.3.23}$$

Because of the assumed Gaussianity for the additive error ε_i, an analytical form of the likelihood function can be obtained

$$p(\mathbf{z} \middle| \mathbf{w}, \sigma^2) = \left(2\pi\sigma^2\right)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2}\left\|\mathbf{z} - \boldsymbol{\Phi}\mathbf{w}\right\|^2\right\}, \tag{8.3.24}$$

in which the $N \times (M+1)$ design matrix $\boldsymbol{\Phi}$ is defined in (8.1.37).

The posterior of weight, $p(\mathbf{w} \mid \mathbf{z})$, is also Gaussian because both its prior and the likelihood function are Gaussian. The error variance σ^2 in (8.3.24) is another hyperparameter. Again, we introduce a precision parameter β to denote the reciprocal of σ^2 (i.e., $\beta = \sigma^{-2}$). The prior of β is also assumed to follow Gamma distribution

$$p(\beta) = \text{Gamma}\left(\beta \middle| c, d\right), \tag{8.3.25}$$

with parameters c and d.

Having defined PDFs for different pieces of the current hierarchical Bayesian problem, we are now ready to estimate the hyperparameters $\boldsymbol{\alpha}$ and β by using the likelihood function (8.3.24). Examination of (8.3.24) shows that the likelihood function is a function of **w**, which can be integrated out (i.e., marginalization) to obtain the following marginal log-likelihood function (Bishop 2006)

$$p(\mathbf{z} \mid \boldsymbol{\alpha}, \beta) = \int p(\mathbf{z} \mid \mathbf{w}, \beta) p(\mathbf{w} \mid \boldsymbol{\alpha}) d\mathbf{w}, \tag{8.3.26}$$

$$\ln\left(p(\mathbf{z} \mid \boldsymbol{\alpha}, \beta)\right) = -\frac{1}{2}\left\{N\ln(2\pi) + \ln\left|\mathbf{R}\right| + \mathbf{z}^T\mathbf{R}^{-1}\mathbf{z}\right\}, \tag{8.3.27}$$

where the $N \times N$ covariance matrix $\mathbf{R}$ is defined as

$$\mathbf{R} = \beta^{-1}\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T,$$

in which **A** is a $(M+1)\times(M+1)$ diagonal matrix of hyperparameters α_i. Estimation of α and β proceeds using the following algorithm:

1. Choose initial values for $\boldsymbol{\alpha}$ and β, and denote them as $\hat{\boldsymbol{\alpha}}$ and $\hat{\beta}$
2. Calculate covariance and mean of the posterior of **w**, which is also Gaussian

$$\mathbf{C} = \left(\hat{\mathbf{A}} + \hat{\beta}\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}. \tag{8.3.28}$$

$$\mathbf{m} = \hat{\beta}\mathbf{C}\mathbf{\Phi}^T\mathbf{z}. \tag{8.3.29}$$

where $\hat{\mathbf{A}}$ is a diagonal matrix consisting of hyperparameters $\hat{\alpha}_i$.

3. Obtain new estimates of $\boldsymbol{\alpha}$ and β from the following equations

$$\alpha_i^{\text{new}} = \frac{\gamma_i}{m_i^2},\ \gamma_i = 1 - \hat{\alpha}_i C_{ii} \tag{8.3.30}$$

$$(\beta^{\text{new}})^{-1} = \frac{\left\|\mathbf{z} - \mathbf{\Phi}\mathbf{m}\right\|^2}{N - \sum_i \gamma_i}, \tag{8.3.31}$$

where m_i is the *i*-th component of the mean vector **m**, C_{ii} is the *i*-th diagonal component of the covariance matrix $\mathbf{C}$ in (8.3.28).

4. Repeat Steps 2–3 in the above until some convergence criterion is met on the estimates.

After the estimates $\hat{\boldsymbol{\alpha}}$ and $\hat{\beta}$ are obtained, we have an RVM learning machine.

Now we can use the trained RVM to predict output $z*$ for any test input $\mathbf{x}*$

$$p\left(z*\middle|\mathbf{x}*, \hat{\boldsymbol{\alpha}}, \hat{\beta}\right) \sim \mathcal{N}\left(z*\middle|\bar{z}, \sigma_z^2\right), \tag{8.3.32}$$

in which the posterior mean and variance of the prediction are defined using the basis function vector $\boldsymbol{\phi}$

$$\bar{z} = \mathbf{m}^T\boldsymbol{\phi}(\mathbf{x}*), \tag{8.3.33}$$

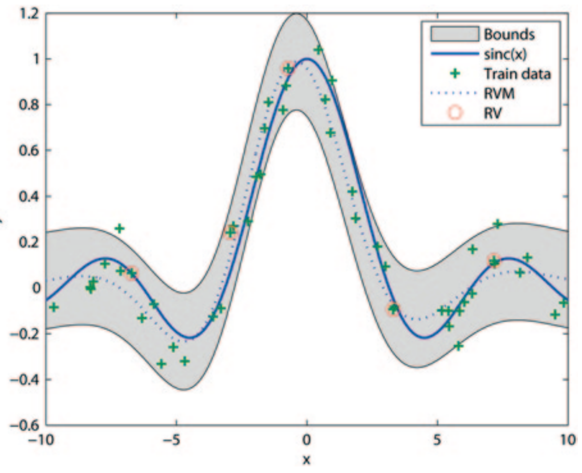


Fig. 8.8 RVM prediction and its prediction bounds on the sinc(x) benchmark problem used in Example 8.7, where RV denotes relevance vectors

$$\sigma_z^2 = \hat{\beta}^{-1} + \boldsymbol{\phi}(\mathbf{x}^*)^T \mathbf{C}\boldsymbol{\phi}(\mathbf{x}^*), \tag{8.3.34}$$

where the predictive variance σ_z^2 is the sum of two terms, the estimated variance of noise $\hat{\beta}^{-1}$ and the uncertainty of weight estimates $\boldsymbol{\phi}(\mathbf{x}^*)^T \mathbf{C}\boldsymbol{\phi}(\mathbf{x}^*)$.

In summary, we see that RVM gives probabilistic solutions, and the hyperparameters are determined by minimizing the log-likelihood function (8.3.27). Thus, there is no need to use cross-validation to determine the hyperparameters as it is done for ε-SVR. RVM basis functions can be general and do not need to be positive-definite kernels like in the case of the SVM. The posterior distributions of many of the weights are sharply peaked around zero and, thus, set to zero. Data that are associated with the remaining nonzero weights are referred to as the *relevance vectors* by Tipping (2001). RVM is found to give models that are typically an order of magnitude more compact (i.e., fewer nonzero weights) than the corresponding ε-SVR for many regression problems and yet, has little or no impact on generalization error (Bishop 2006). However, RVM does not offer explicit control on the structural risk the way SVM does. The derivation of RVM assumes Gaussian-distributed additive errors, which may not always hold in practice. A Matlab implementation of RVM has been made available by Tipping (2009).

Example 8.8 *Application of RVM*
Let us revisit the problem in Example 8.7 and use the same training dataset to approximate the sinc(x) function. The Gaussian kernel is again used as basis function. The spread of the Gaussian kernel is set to 3.0 such that the resulting MSE of RVM (0.0824) is comparable to that of the ε-SVR in the first case of Example 8.7. In this case, RVM selected 5 relevance vectors (circles), as opposed to 8 support vectors selected by the ε-SVR algorithm. The final values of $\hat{\boldsymbol{\alpha}}$ and $\hat{\beta}$ are $[0.159, 0.894, 0.782, 0.004, 0.016]^T$ and 90.34, respectively. The bounds corresponding to $\pm 2\sigma_z$ (~95 % confidence interval) are plotted in Fig. 8.8. Comparing Fig. 8.8 to

Fig. 8.7a, we see that the relevance vectors tend to locate closer to the true function (solid curve), whereas the support vectors tend to deviate significantly from the true curve. This is caused by the difference in the design of the two algorithms: RVM seeks to minimize the empirical risk (log-likelihood) by using as few relevance vectors as possible, whereas ε-SVR minimizes the structural risk by penalizing large deviations from the ε-tube. ■

Note that both SVM and RVM select a subset of actual training data to achieve sparsity. Other learning algorithms use "virtual" data centers, which may or may not coincide with any of the training data. An example is the RBFN described in Sect. 8.2, for which either the K-means clustering algorithm or orthogonal least squares is used to find data centers. The standard RVM algorithm does not include estimation of hyperparameters of basis functions. In the next section, we consider another Bayesian method that streamlines all parameter estimation tasks.

8.4 Gaussian Process Regression

In this section, we turn to *Gaussian Process* (GP) models, which serve as a pillar for many data-driven models commonly employed in machine learning in recent years. Formally speaking, a GP is a collection of random variables for which the joint PDF of any of its subsets is multiGaussian. When extending the definition to function outputs, we say that a random process $f(\mathbf{x})$ is a GP if it is completely specified by its mean and covariance

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), \mathrm{cov}(\mathbf{x}, \mathbf{x}'))$$

$$m(\mathbf{x}) = E[f(\mathbf{x})]$$

$$\mathrm{cov}(\mathbf{x}, \mathbf{x}') = E[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))].$$

The GP can be seen as a framework for unifying several methods that have been introduced in this chapter. The GP can be derived using linear regression. It has been shown that for fixed hyperparameters, a large class of ANN models will converge to a GP in the limit of an infinite number of hidden units because of the central limit theorem (Neal 1996; Williams 1998).

Because of its well-known mathematical properties, the GP has been extensively used in Bayesian inversion algorithms. The main focus here is using GP as a tool to develop data-driven models, a technique commonly referred to as the Gaussian process regression (GPR) in machine learning theory (Rasmussen and Williams 2006). Mathematically, GPR is an extension of kriging methods in classical geostatistics (Chap. 6). However, as it will be shown below, the GPR is not simply a reinvention of kriging. Instead, it is a flexible machine learning framework that emphasizes on easing the training process by combining hyperparameter estimation, model training, and uncertainty quantification in a hierarchical Bayesian framework. The classical kriging or cokriging algorithms usually involve a separate variogram modeling step, which require quite some user intervention and judgment, whereas GPR automates the estimation of hyperparameters.

In the following, two equivalent views and also derivations of the GPR are provided: the weight space view and functional space view. The derivations follow those given by Rasmussen and Williams (2006).

8.4.1 Weight Space View

Let us again start with the generic regression model (8.1.35):

$$f(\mathbf{x};\mathbf{w}) = \sum_{i=0}^{M} w_i \phi_i(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}), \tag{8.4.1}$$

where $\mathbf{w}$ is the unknown weight vector; $\boldsymbol{\phi} = \{\phi_i(\mathbf{x})\}_{i=0}^{M}$ is a set of basis functions that provide mapping from the input space to feature space, and $\boldsymbol{\phi}$ includes $\phi_0 = 1$ for the bias term. The derivation of the GPR under the weight space view is largely on par with that of the RVM. In particular, the problem at hand is to compute the posterior distribution of $\mathbf{w}$ from its likelihood and prior.

Given a set of training data pairs, $\{(\mathbf{x}_i, z_i)\}_{i=1}^{N}$, we again assume the relationship (8.3.19) exists between observations of target variable z_i and model approximation (8.4.1)

$$z_i = f(\mathbf{x}_i;\mathbf{w}) + \varepsilon_i,\ \varepsilon_i \in \mathcal{N}(0,\sigma^2),\ i = 1,\cdots,N. \tag{8.4.2}$$

As a result, the likelihood function is the same as that used in RVM

$$\begin{aligned} p(\mathbf{z} \mid \mathbf{w}) &\sim \prod_{i=1}^{N} p\left(z_i \middle| \mathbf{x}_i, \mathbf{w}\right) \\ &\sim \prod_{i=1}^{N} \mathcal{N}\left(z_i \middle| \mathbf{w}^T \boldsymbol{\phi}\,(\mathbf{x}_i), \sigma^2\right), \end{aligned} \tag{8.4.3}$$

where $\mathbf{z}$ is the target output vector and N is the total number of training data pairs. The predictive distribution is defined as the PDF of target output z^* for any test input value $\mathbf{x}^*$ and integrated over weights. Eliminating $\mathbf{w}$ by marginalization gives

$$p(z^* \mid \mathbf{x}^*, \mathbf{z}) = \int p(z^* \mid \mathbf{x}^*, \mathbf{w}) p(\mathbf{w} \mid \mathbf{z}) d\mathbf{w}. \tag{8.4.4}$$

To evaluate (8.4.4), however, some knowledge of the prior distribution over $\mathbf{w}$ is needed. A convenient choice for the prior of $\mathbf{w}$ is a zero-mean Gaussian,

$$p(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_p), \tag{8.4.5}$$

where $\mathbf{C}_p$ is the covariance matrix of the prior distribution. Comparing (8.4.5) with the prior used in RVM (8.3.18), we see that the latter, which assumes independent Gaussian-distributed weights, can be considered a special case of the former when $\mathbf{C}_p$ is diagonal. The posterior of **w**, which can be expressed as a product of (8.4.3) and (8.4.5), is also Gaussian

$$p\left(\mathbf{w}\middle|\,\mathbf{z}\right) \propto p\left(\mathbf{z}\middle|\,\mathbf{w}\right) p(\mathbf{w}). \tag{8.4.6}$$

Expanding the likelihood function $p\left(\mathbf{z}\middle|\,\mathbf{w}\right)$, we have

$$p\left(\mathbf{z}\middle|\,\mathbf{w}\right) = (2\pi\sigma^2)^{-N/2} \exp\left(-\frac{1}{2\sigma^2} \left\| \mathbf{z} - \mathbf{\Phi}\mathbf{w} \right\|^2 \right). \tag{8.4.7}$$

where $\mathbf{\Phi}$ is the design matrix defined in (8.1.37). Thus, the posterior mean **m** and covariance matrix **C** of **w** are obtained by combining (8.4.7) and (8.4.5) (Rasmussen and Williams 2006)

$$\mathbf{C} = \left(\sigma^{-2} \mathbf{\Phi}^T \mathbf{\Phi} + \mathbf{C}_p^{-1} \right)^{-1} \tag{8.4.8}$$

$$\mathbf{m} = \sigma^{-2} \mathbf{C}^T \mathbf{z} \tag{8.4.9}$$

The predictive distribution, (8.4.4), is the convolution of two Gaussian distributions; thus, it is also a Gaussian. The mean and variance of the predictive distribution are expressed in terms of the mean and covariance of **w**

$$\overline{z} = \mathbf{m}^T \boldsymbol{\phi}(\mathbf{x}^*) = \frac{1}{\sigma^2} \boldsymbol{\phi}(\mathbf{x}^*)^T \mathbf{C}\ \mathbf{\Phi}^T \mathbf{z}, \tag{8.4.10}$$

$$\sigma_z^2 = \boldsymbol{\phi}(\mathbf{x}^*)^T \mathbf{C}\ \boldsymbol{\phi}(\mathbf{x}^*), \tag{8.4.11}$$

where $\boldsymbol{\phi}(\mathbf{x}^*) = [1, \phi_1(\mathbf{x}^*), \cdots, \phi_M(\mathbf{x}^*)]^T$. Using the Woodbury matrix identity (Petersen and Pedersen 2012), we can rewrite the prediction statistics in slightly different forms

$$\overline{z} = \boldsymbol{\phi}(\mathbf{x}^*)^T \mathbf{C}_p \mathbf{\Phi} \left(\sigma^2 \mathbf{I} + \mathbf{\Phi} \mathbf{C}_p \mathbf{\Phi}^T \right)^{-1} \mathbf{z}, \tag{8.4.12}$$

$$\begin{aligned}\sigma_z^2 &= \boldsymbol{\phi}(\mathbf{x}^*)^T \mathbf{C}_p \boldsymbol{\phi}(\mathbf{x}^*) \\ &-\boldsymbol{\phi}(\mathbf{x}^*)^T \mathbf{C}_p \boldsymbol{\Phi}^T (\sigma^2 \mathbf{I} + \boldsymbol{\Phi}\mathbf{C}_p \boldsymbol{\Phi}^T)^{-1} \boldsymbol{\Phi}\mathbf{C}_p \boldsymbol{\phi}(\mathbf{x}^*),\end{aligned} \tag{8.4.13}$$

from which we see that the matrix entries all appear in the form $\boldsymbol{\phi}(\mathbf{x})^T \mathbf{C}_p \boldsymbol{\phi}(\mathbf{x}')$, where $\mathbf{x}$ and $\mathbf{x}'$ are either testing or training input data. We can define the following symbol to denote the covariance

$$k(\mathbf{x},\mathbf{x}') \triangleq \boldsymbol{\phi}(\mathbf{x})^T \mathbf{C}_p \boldsymbol{\phi}(\mathbf{x}') = \boldsymbol{\psi}(\mathbf{x})^T \boldsymbol{\psi}(\mathbf{x}'), \tag{8.4.14}$$

where $\boldsymbol{\psi} = \mathbf{C}_p^{1/2}\boldsymbol{\phi}$. Eq. (8.4.14) is exactly the kernel function that was described in Sect. 8.3, where the "kernel trick" enables us to work with the covariance function in the input space without having to know the actual form of the basis function.

In GPR, the free parameters include the hyperparameters of kernel functions and σ^2, which can be estimated using methods given in Sect. 7.4.2. The Gaussian kernel (also called squared exponential covariance function) is commonly used in GPR. More complex kernel functions can be constructed out of simple ones, as mentioned in Sect. 6.4. One of the most widely used composite kernels for GPR is given by the superposition of several terms (Rasmussen and Williams 2006)

$$k(\mathbf{x},\mathbf{x}') = \theta_0 \exp\left\{-\frac{\theta_1}{2}\left\|\mathbf{x}-\mathbf{x}'\right\|^2\right\} + \theta_2 + \theta_3 \mathbf{x}^T \mathbf{x}', \tag{8.4.15}$$

where $\theta_i (i = 0,\cdots,3)$ are hyperparameters. Other examples can be found in Rasmussen and Williams (2006, Chap. 4).

In RVM, each weight is individually penalized by its precision parameter (i.e., reciprocal of variance). The weights are effectively set to zero when the precision parameters have large values. Sparseness in RVM is achieved by pruning zero weights and associated basis functions, making RVM faster than a nonsparse GP model. However, RVM can yield undesirable results when a test point is located far from the relevance vectors, in which case the predictive distribution will degenerate into a Gaussian with mean close to zero and variance also close to zero (Rasmussen and Williams 2006). The small predictive variance, of course, gives false confidence about the prediction in this case.

8.4.2 Functional Space View

Under the functional space view of GP, the goal is to estimate the statistics of a random function $f(\mathbf{x})$ using its discrete values $f(\mathbf{x}_i)$ at training inputs $\mathbf{x}_i$, and each $f(\mathbf{x}_i)$ is regarded as a random variable. Let us consider the following additive-error model without the weight $\mathbf{w}$

$$z_i = f(\mathbf{x}_i) + \varepsilon_i,\ \varepsilon_i \in \mathcal{N}(0, \sigma^2),\ i = 1, \cdots, N.$$

The joint PDF of target **z** conditioned on function outputs $\mathbf{f} = [f(\mathbf{x}_1), \cdots, f(\mathbf{x}_N)]^T$ can be obtained by applying Bayes' rule

$$p(\mathbf{f} \mid \mathbf{z}, \sigma^2) \propto p(\mathbf{z} \mid \mathbf{f}, \sigma^2) p(\mathbf{f}), \tag{8.4.16}$$

where $\mathbf{X}$ represents all input training data stored in rows. By definition of GP, the prior of **f** is Gaussian

$$p(\mathbf{f} \big| \boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K}), \tag{8.4.17}$$

where **K** is the covariance matrix of **f**, and $\boldsymbol{\theta}$ is the corresponding hyperparameter. From discussion in the last section, **K** can also be considered as a Gram matrix with its elements defined by kernel functions,

$$k_{nm} = k(\mathbf{x}_n, \mathbf{x}_m),\ n = 1, \cdots, N,\ m = 1, \cdots, N.$$

Then $\boldsymbol{\theta}$ includes hyperparameters used to define all kernel functions. The likelihood of **f** is Gaussian because of the Gaussian additive error assumption

$$p(\mathbf{z} \mid \mathbf{f}, \sigma^2) \sim \mathcal{N}(\mathbf{f}, \sigma^2 \mathbf{I}). \tag{8.4.18}$$

As a result, the posterior of **f** is Gaussian with mean and covariance given by

$$\mathbf{m} = \mathbf{K}^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{z}, \tag{8.4.19}$$

$$\mathbf{C} = \mathbf{K} - \mathbf{K}^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{K}. \tag{8.4.20}$$

To estimate the unknown hyperparameters in (8.4.19)–(8.4.20), we formulate the following marginal PDF,

$$p(\mathbf{z} \big| \sigma^2) = \int p\left(\mathbf{z} \big| \mathbf{f}, \sigma^2\right) p(\mathbf{f}) d\mathbf{f}. \tag{8.4.21}$$

After substituting the prior (8.4.17) and likelihood (8.4.18) into (8.4.21), performing integration, and taking logarithm of the results, we have

$$\begin{aligned} &\log p(\mathbf{z} \big| \sigma^2, \boldsymbol{\theta}) \propto \\ &-\frac{1}{2} \mathbf{z}^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{z} - \frac{1}{2} \log \left| \mathbf{K} + \sigma^2 \mathbf{I} \right| - \frac{N}{2} \log(2\pi). \end{aligned} \tag{8.4.22}$$

The hyperparameters $\{\sigma^2, \boldsymbol{\theta}\}$ can then be estimated from (8.4.22) using a gradient-based algorithm introduced in Chap. 2.

It can be shown that the prediction distribution for any test data $\mathbf{x}^*$ is Gaussian (Quiñonero-Candela and Rasmussen 2005)

$$p(f^* \mid \mathbf{z}, \mathbf{x}^*, \sigma^2) = \int p(f^* \mid \mathbf{f}, \mathbf{x}^*) p(\mathbf{f} \mid \mathbf{z}, \sigma^2) d\mathbf{f} \tag{8.4.23}$$

with mean and covariance defined by

$$\bar{z} = \mathbf{k}_*^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{z}, \tag{8.4.24}$$

$$\sigma_z^2 = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}_*^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_*, \tag{8.4.25}$$

where $\mathbf{k}_*$ denotes a vector of covariance between the test point $\mathbf{x}^*$ and all training inputs. Using the definition of kernel function (8.4.14), we can verify that the prediction statistics obtained here are equivalent to those obtained under the weight space view.

The computational cost associated with learning the full GP model is $O(N^3)$, mainly because of the need to invert the covariance matrix. The costs associated with calculating the prediction mean and variance are $O(N)$ and $O(N^2)$, respectively (Rasmussen and Williams 2006, Chap. 8). Thus, for a large training dataset, some approximation to the full-rank covariance matrix is necessary, which is the subject of study of sparse Bayesian GPR. The key idea is to extract a small number of features that can approximate the original covariance matrix well. Such model reduction techniques are behind PCA and KPCA. Also note that equations (8.4.24)–(8.4.25) are identical to cokriging equations given in Sect. 7.4.3 and the classic Kalman filter equations to be introduced in Chap. 9. To avoid inversion of the full-rank covariance matrix, we know that a practical technique in kriging is to construct a small-sized system of equations using data points located only in a local neighborhood of the estimation point. Similar localization ideas may be applied when constructing sparse Bayesian systems. Indeed, we will show how these techniques are used in constructing reduced-rank Kalman filters in Chap. 9.

Example 8.9 *Using GPR on the Sinc(x) Function*
We use GPR to develop a nonparametric predictive model based on the same dataset that was used in Examples 8.7 and 8.8. We use the first two terms of the composite covariance model given in (8.4.15) (i.e., the covariance term and the constant nugget term). The first term is assumed to be a unit variance, isotropic squared exponential function (Gaussian kernel),

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\left\|\mathbf{x} - \mathbf{x}'\right\|^2}{2l^2} \right)$$

where l is the correlation length. The hyperparameters are the nugget term and the length scale (l) in the squared exponential function, which are initialized to 0 and

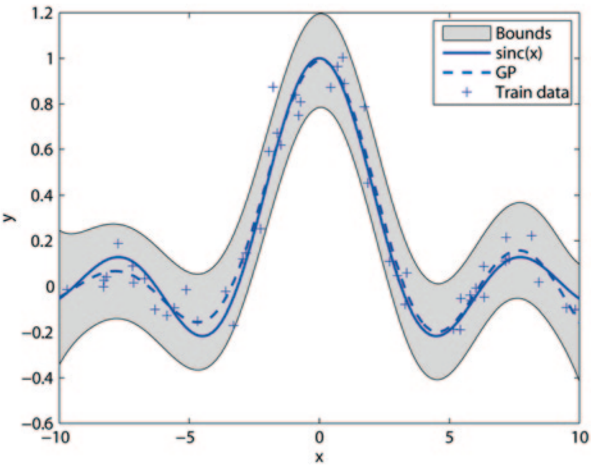


Fig. 8.9 GP prediction on the sinc(x) benchmark problem used in Example 8.7

1.0, respectively. The GPR problem is solved using the GPML Matlab toolbox developed by Rasmussen and Nickisch (2010), which automates the estimation of hyperparameters and model training. In this case, the final hyperparameters are 0.052 and 1.17, respectively. Figure 8.9 shows the GPR prediction and the estimation bounds. The GPR used all 50 data pairs in training. The results are more accurate than those from SVM and RVM, although the performance of the latter two sparse algorithms could be improved by increasing the number of either support vectors or relevant vectors. Setting up the GPR problem in the GPML Matlab toolbox is straightforward, and the user can perform training and prediction in just two function calls.

Example 8.10 *One-Month-Ahead Streamflow Forecast Using SVR, RVM, and GPR* Having demonstrated the learning mechanisms and performance of SVR, RVM, and GPR over a simple regression problem, we now compare the three algorithms for one-month-ahead streamflow forecasting using Neches River streamflow data (see Example 8.1). The daily hydrometeorological observations are first converted into monthly data through either aggregation (for precipitation) or averaging (for T_{max}, T_{min}, and streamflow). The Box-Cox transform is applied to the flow data, and all input data are scaled linearly to the interval [−1, 1], just as we did in the MLP example. The predictors include the following variables at different lags:

Precipitation: $P(t-1),\ P(t-2)$
Temperature: $T_{max}(t-1),\ T_{max}(t-2),\ T_{min}(t-1)$
Streamflow: $Q(t-1),\ Q(t-2)$

The target output is $Q(t)$ for the prediction month t. About 70 % of the data is used for training and the remaining is used for testing. Gaussian kernel is used in all three methods and the spread parameter is set to 1.0. The parameter c in ε-SVR is set to

Table 8.3 Performance metrics of the regression algorithms

Algorithm	NSC	RMSE
ε-SVR	0.616	3.33
RVM	0.560	3.56
GPR	0.657	3.15

0.09 and ε is set to 0.15 on the basis of five-fold cross-validation using LibSVM. The performance metrics on the testing dataset are summarized in Table 8.3, which shows that all three algorithms obtain similar results, with the performance of GPR better than the other two. ■

Recently, Sun et al. (2014b) applied GPR to one-month-ahead streamflow prediction using the MOPEX database, which is a collection of long-term hydrometeorological time series obtained from 438 natural basins across the USA from 1948 to 2003. Comparisons with linear regression (ARMAX) and artificial neural network (MLP) models showed that GPR outperformed both methods in most cases. Elsewhere in EWR, the GPR has been applied to metamodeling (Pau et al. 2013) and monitoring network design (Krause et al. 2008).

8.5 Review Questions

1. What is the general form of data-driven models? In what cases data should we develop data-driven models instead of physical-based models?
2. Why do we need to control the complexity of a regression model? What are the main advantages of the shrinkage method over other methods (e.g., PCR) when used for reducing the dimension of linear models?
3. What is the difference between Lasso and ridge regression?
4. Extend the shrinkage method to the case of two-target regression.
5. What is the basic idea behind the cross-validation method? What is the limitation of the multi-fold cross-validation when the number of possible predictors is large? Discuss how this method can be applied to kriging estimation.
6. Repeat the ANN derivation from equation (8.2.1) to equation (8.2.11) for the three-layer case.
7. What are the advantages of RBFN? Derive equations (8.2.22) and (8.2.23).
8. How do we generate $\mathbf{f}_l(\mathbf{x})$ in equation (8.2.25)? Give examples of different types of methods can be used to generate ANN ensembles. And, how do we use an ensemble to assess the uncertainty of an ANN generalization? Why ensemble methods can outperform prediction of individual members?
9. What are the differences between ε-SVR and LS-SVR in terms of formulation, objective function, algorithm, and the sparsity? Can SVR be used to develop multioutput models?

10. What improvements are made in RVM compared to the SVM in terms of basis functions and uncertainty estimate?
11. What is the main difference between RVM and GPR (use weight space view for the latter)? Can spatial coordinates be used as predictors in GPR?

Chapter 9
Data Assimilation for Inversion

Many EWR applications are data centric and naturally call for the ability to fuse spatial and temporal information from multiple sources and in different formats and scales. With the rapid advance of in situ and remote sensing technologies and cyberinfrastructures, a major EWR research front in recent years has focused on the development of effective algorithms for integrating real-time data into prediction models. Oftentimes, prior knowledge and historic training data are limited in both quality and quantity, leading to uncertainties in calibrated models and estimated parameters. Such issues are relevant not only to distributed EWR models, but also to data-driven models, as we have seen in Chap. 8. A fundamental need in EWR is thus related to systematic and continuous extraction of useful information from new observations to provide updated estimates of model states and parameters. In this chapter, mathematical tools and methods that can be applied to automate such information fusion process will be introduced.

Generally speaking, the aim of *data assimilation* (DA) is to incorporate newly obtained observations into a system model to provide updated estimates of system states and parameters, together with a posteriori error estimates (Nichols 2010). In most cases, DA seeks to sequentially reduce the discrepancy between observations and model predictions by using predefined fitting criteria and model application objectives. Liu and Gupta (2007) suggested that the purpose of DA is to merge uncertain data and information produced by imperfect models in an optimal way such that both uncertainty quantification and uncertainty reduction can be achieved simultaneously. This broad interpretation of DA covers all three objectives of this book, namely experimental design, model structure identification, and parameter estimation. In addition, DA is intertwined with data collection design and data worth (or value of information) evaluation. Because model structure identification and experimental design require a considerable amount of off-line analysis and user intervention, they are typically done separately. With this in mind, the scope of this chapter is restricted to *sequential DA* (SDA) algorithms that are suitable for online update of both model states and parameters for given model structures. Of course, instead of working with a single model or single model structure class, one may use the same forcing data to update an ensemble of models such that the effect of

N-Z. Sun, A. Sun, *Model Calibration and Parameter Estimation,*
DOI 10.1007/978-1-4939-2323-6_9

conceptualization uncertainty is incorporated in an ensemble averaging sense. This latter topic will be covered in Chap. 10, as part of the discussion on uncertainty quantification techniques.

SDA is deeply rooted in system and control engineering, a field that has always had a high demand for recursive online estimators to handle real-time sensor information. The earliest idea of DA came from the German mathematician Karl F. Gauss, who first formulated a recursive least squares method for estimating parameters in linear regression (Gauss 1826). The recursive least squares method was rediscovered by Plackett in 1950s (Plackett 1950). However, a real burgeoning of interests in DA only started in 1960s, after the introduction of the now classical Kalman filter (KF). Wide adoption of DA in EWR applications, especially for distributed systems, is a relatively recent phenomenon. In fact, some early reviews (e.g., McLaughlin and Townley 1996) were dismissive about the use of SDA for updating distributed parameters sequentially, citing large computational burdens. Most DA studies during that time focused on lumped-parameter, low-dimensional systems. A noteworthy early study on real-time river discharge prediction was done by Kitanidis and Bras (1980), who stated that adaptive estimation might be suitable when the forecast lead time is short in comparison with the response time of a watershed. With the advance of computing power, DA has received significantly renewed interest in recent years, especially in numerical weather prediction and hydrometeorology, which feature some of the most advanced DA systems that are in operation today. An example is the Land Data Assimilation Systems (LDAS) developed by the US National Aeronautics and Space Administration. LDAS includes a coarse-resolution (1°) global version and a finer-scale (1/8°) North America version. Both systems operate in near real time and are forced with precipitation gauge observations, satellite data, and radar precipitation measurements. LDAS has been used to support numerous water resources applications, climate prediction studies, and global water- and energy-cycle investigations (Rodell et al. 2004; Su et al. 2008; Reichle et al. 2010; Hain et al. 2012).

Despite the demonstrated success of DA techniques in past decades, questions remain regarding the scalability, rigorousness, and robustness of DA schemes. The field of DA is fast growing. The sheer number of existing DA schemes may already overwhelm novice users, not to mention those schemes for which efficacy has not been adequately tested. For the latter reason, we will focus primarily on the most well-established DA schemes in this chapter while providing references to the latest developments where necessary.

Section 9.1 introduces basic DA concepts. A special emphasis will be given to parameter estimation problems, which are the main theme of this book. The classical KF can be adapted for solving joint state and parameter estimation problems when the system being considered is linear and subjected to white noise (Sect. 9.2). When the system under consideration is nonlinear, the classical KF becomes inappropriate. The extended KF is introduced to handle nonlinearity (Sect. 9.3). When the system of interest has high-dimensional state and parameter space, the extended KF is of limited value. For instance, the dimension of state space in hydrometeorological DA systems is on the order of 10^7–10^8 and the number of observations can be on the order of 10^5–10^6 (Nichols 2010; Kalnay 2003). Therefore, dimension reduction (Chap. 6) becomes necessary to make DA feasible. We introduce the so-called reduced-rank or sparse filters, including the ensemble KF and its variants in Sect. 9.4. As a result of their approximation nature, a key aspect of all reduced-rank filters revolves around maintaining filter stability during operation. Finally, when the system is nonlinear and non-Gaussian, only a few choices are left. The generic particle filter will be introduced in Sect. 9.5.

9.1 Basic Concepts and Methods

SDA is a model calibration procedure that sequentially applies information contained in new measurements to make the prediction of an existing model more accurate. In this section, SDA is formulated as a sequential estimation problem in the statistical framework, as well as a tool for jointly estimating system states and model parameters. This section also provides an overview of various SDA algorithms that will be introduced separately in the remaining sections.

9.1.1 State-Space Model

9.1.1.1 Definition and examples

All SDA problems involve some type of *discrete-time, (continuous) state-space* models, which usually consist of a pair of prediction and measurement equations

$$\mathbf{u}_k = \mathbf{f}(\mathbf{u}_{k-1}, \boldsymbol{\theta}_{k-1}) + \boldsymbol{\eta}_k, \tag{9.1.1}$$

$$\mathbf{z}_k = \mathbf{g}(\mathbf{u}_k, \boldsymbol{\theta}_k) + \boldsymbol{\varepsilon}_k, \tag{9.1.2}$$

where k is an index of discrete-time steps; $\mathbf{u} \in \mathbb{R}^n$ represents the discretized system states concatenated in a vector form; the forward model operator $\mathbf{f}(\cdot)$ evolves the system state from time t_{k-1} to t_k to give $\mathbf{u}_k$; $\mathbf{g}(\cdot)$ maps system state $\mathbf{u}_k$ to observations $\mathbf{z}_k \in \mathbb{R}^m$; $\boldsymbol{\eta}_k$ and $\boldsymbol{\varepsilon}_k$ represent model (or process) error and observation error, respectively; and $\boldsymbol{\theta}$ represents model inputs, which may include model parameters, initial/boundary conditions, and sink/source terms. For forward solution, $\boldsymbol{\theta}$ is assumed to be given.

Equation (9.1.1) is also referred to as the evolution equation, state equation, or forecast equation in different application fields. For EWR modeling, it can be seen as a time-discrete forward solution of a physics-based model governed by ODEs or PDEs. Eq. (9.1.2) is used to compare the model prediction obtained in (9.1.1) with the newly obtained data. If the model state is directly observable, the measurement operator $\mathbf{g}(\cdot)$ in (9.1.2) is a simple matrix with elements equal to either 0 or 1; otherwise, $\mathbf{g}(\cdot)$ can be complex and may well be another nonlinear mathematical model or inverse solution. The latter scenario can be especially true in geophysical and remote sensing applications, where physical variables of interest (e.g., soil types, soil moisture, and water storage anomalies) are inferred indirectly from raw signals (e.g., electric conductivity, radiance, and gravity measurements). Measurement errors arise because of instrumentation limitations and signal conversion uncertainty. Yet another possibility is due to the fact that the model state **u** is discrete, whereas observations represent snapshots of a continuous true state; thus, errors can appear because of the discrepancy between model output scale and observation scale, either in time or space, or both. This type of error is often referred to as the representativeness error (Cohn 1997). For the rest of this discussion, we shall assume that all measurement errors are random (i.e., no systematic error is involved).

Examples of discrete-time state-space EWR models are virtually everywhere, from pore-scale process models to global climate models. In principle, we may convert any dynamic model to a discrete-time, state-space model, using numerical discretization techniques.

Example 9.1 *Formulation of a discrete-time, state-space model*
Example 1.10 gives the numerical solution of the general advection–dispersion–reaction problem governed by PDE (1.1.13),

$$\frac{\partial C}{\partial t} = \nabla\cdot[\mathbf{D}\nabla C] - \nabla\cdot(\mathbf{V}C) - R(C) + E.$$

After time and space discretization, the unknown concentration field $\mathbf{C}_k$ at time step k is obtained forwardly from the solved concentration field $\mathbf{C}_{k-1}$ at time step $k-1$ by using Eq. (1.2.5),

$$\mathbf{C}_k = \left(\frac{\mathbf{B}}{\Delta t_k} + \alpha\mathbf{A}\right)^{-1}\left(\frac{\mathbf{B}}{\Delta t_k} - (1-\alpha)\mathbf{A}\right)\mathbf{C}_{k-1} + \mathbf{r}, \qquad (9.1.3)$$

where $\Delta t_k = t_k - t_{k-1}$ and $0 \le \alpha \le 1$, with $\alpha = 0$ for explicit scheme and $\alpha = 1$ for implicit scheme. Coefficient matrices $\mathbf{A}$, $\mathbf{B}$, and the right-hand-side vector $\mathbf{r}$ depend on parameters $\mathbf{D}$, $\mathbf{V}$, R, and E, boundary conditions, and the method used for spatial discretization.

When the same problem is solved by a particle tracking method, the time-discrete solution is given by Eq. (1.2.7), viz.

$$\mathbf{C}_k = \mathcal{F}(\mathbf{C}_{k-1}; \mathbf{D}, \mathbf{V}, R, E).$$

In this example, the state variable **u** in (9.1.1) becomes concentration field **C** and the operator $\mathbf{f}(\cdot)$ is obtained by the forward solution.

If the concentration is directly observed at a set of locations, then the measurement operator is a matrix $\mathbf{G}$ with elements equal to 1 at observation locations and 0 otherwise. The measurement equation is written as $\mathbf{z}_k = \mathbf{G}\mathbf{C}_k + \boldsymbol{\varepsilon}_k$.

Example 9.2 *Change Detection*
Change detection appears in many EWR applications, such as detecting changes in hydrometeorological time series caused by climate change (slow changes) (Seidou

et al. 2007; Xu et al. 2013) or in monitored pollutant levels (abrupt changes). Formally speaking, change detection is the identification of significant changes in the generative parameters of a dynamic system. In the simplest case, the system is assumed to be controlled by a single parameter θ, which takes different values before and after a change point. The state-space model in (9.1.1)–(9.1.2) now needs to be augmented by a model for θ. For example, in the case of mean shift, we have

$$\theta_k = \begin{cases} \theta_1, & t_k \le \tau \\ \theta_2, & t_k > \tau \end{cases},$$

$$\theta_k = \theta_{k-1} + \upsilon_k,$$

where τ is the unknown change time. Before the change, the residuals are white noise, and after the change, the residuals become large. The goal is to determine a set of rules for determining whether there is a significant change. Change point detection problem can be solved both in the traditional statistical sense (i.e., hypothesis testing) and in the Bayesian framework. A formal discussion on parameter evolution modeling will be given in Sect. 9.1.2.

9.1.1.2 SDA Algorithms

As its name suggests, a SDA algorithm is recursive by nature: From a given initial state $\mathbf{u}_0$, an estimate $\mathbf{u}_1$ is found, and then, $\mathbf{u}_2$ is found from the estimated $\mathbf{u}_1$ and so forth. In general, the process of finding $\mathbf{u}_k$ from $\mathbf{u}_{k-1}$ consists of two stages as sketched in Fig. 9.1. In the first stage, model (9.1.1) is used to generate a prediction of $\mathbf{u}_k$, $\mathbf{u}_k^f$, where the superscript f is used to emphasize that it is a result of model forecast. Because of the existence of model errors and the uncertainty associated with $\mathbf{u}_{k-1}$ and θ_{k-1}, $\mathbf{u}_k^f$ may not be as close as desired to the true $\mathbf{u}_k$. In the second stage, new measurements $\mathbf{z}_k$ in (9.1.2) are used to update $\mathbf{u}_k^f$ to generate a newer and, hopefully, improved estimate of $\mathbf{u}_k$, $\hat{\mathbf{u}}_k$ (note: the posterior estimate $\hat{\mathbf{u}}_k$ is also denoted as analysis $\mathbf{u}_k^a$ in some textbooks). The first stage is called *forecast*; it is

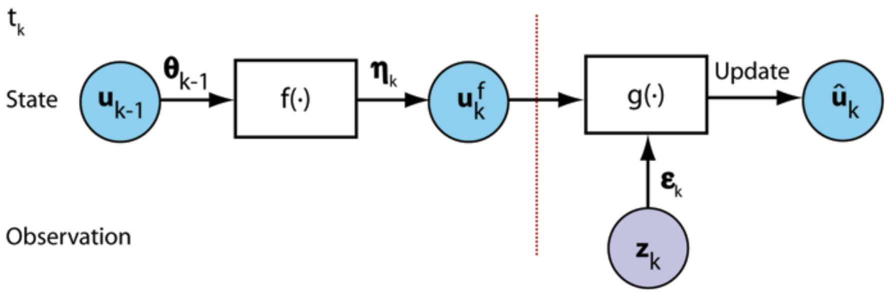


Fig. 9.1 A typical DA step in SDA, where $\mathbf{u}$ denotes model state, $\boldsymbol{\theta}$ the model inputs, $\mathbf{z}$ the observations, and $\boldsymbol{\eta}$ and $\boldsymbol{\varepsilon}$ are error terms. The forward model operator is $\mathbf{f}(\cdot)$ and the measurement operator is $\mathbf{g}(\cdot)$. The *dotted line* splits the forecast and analysis steps

simply a model prediction or a forward solution problem. The second stage is called *update*, *analysis*, *or assimilation* and it is actually an inverse problem: Using $\mathbf{u}_k^f$ as the initial guess, find the best estimate of $\mathbf{u}_k$ by incorporating new observations $\mathbf{z}_k$.

Existing DA frameworks can be divided into two broad categories, *deterministic* and *stochastic*. In the former, the inverse problem is solved by methods introduced in Chaps. 2 and 3. The objective functions to be minimized are usually written in a similar form as Tikhonov regularization, and the gradients of the objective function are evaluated by the adjoint-state method when the forward model is distributed. But, most SDA algorithms, as we shall see below, are based on Bayesian inference introduced in Chap. 4.

9.1.1.3 A Bayesian View of DA

When casting in the stochastic framework, system states $\mathbf{u}$, parameter $\boldsymbol{\theta}$, and measurement $\mathbf{z}$ in (9.1.1) and (9.1.2) are all regarded as stochastic processes. The Bayesian theory then provides a flexible framework for propagating the uncertainty of $\mathbf{u}_{k-1}$ to $\mathbf{u}_k^f$ in the forecast step and subsequently updating the estimate $\hat{\mathbf{u}}_k$ in the assimilation step. When $\boldsymbol{\theta}$ is deterministic, the full posterior distribution of state variables from the initial time to any time t_K is given by

$$p(\mathbf{u}_{0:K} \mid \mathbf{z}_{1:K}) \propto p(\mathbf{z}_{1:K} \mid \mathbf{u}_{0:K})p(\mathbf{u}_{0:K}), \tag{9.1.4}$$

where $\mathbf{u}_{0:K} = \{\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_K\}$, $\mathbf{z}_{1:K} = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_K\}$, $p(\mathbf{u}_{0:K})$ is the joint prior distribution of $\mathbf{u}_{0:K}$, and $p(\mathbf{z}_{1:K} \mid \mathbf{u}_{0:K})$ is the likelihood function. Obviously, updating a full PDF every time when new information becomes available is a demanding task. If we assume that the sequence $\mathbf{u}_{0:K}$ generated by (9.1.1) is a first-order Markov process (i.e., in $\mathbf{u}_{0:k}$, $\mathbf{u}_k$ depends only on $\mathbf{u}_{k-1}$ for $k = 1, 2, \cdots, K$), then using the one-step transition PDF, $p(\mathbf{u}_k \middle| \mathbf{u}_{k-1})$, the prior PDF can be represented by

$$p(\mathbf{u}_{0:K}) = p(\mathbf{u}_0)\prod_{k=1}^{K} p(\mathbf{u}_k \middle| \mathbf{u}_{k-1}), \tag{9.1.5}$$

where $p(\mathbf{u}_0)$ is the PDF of the initial state. Further, if measurement errors from different assimilation times are independent of each other and $\mathbf{z}_k$ depends only on $\mathbf{u}_k$, the likelihood $p(\mathbf{z}_{1:K} \mid \mathbf{u}_{0:K})$ can also be expressed in a product form

$$p(\mathbf{z}_{1:K} \mid \mathbf{u}_{0:K}) = \prod_{k=1}^{K} p(\mathbf{z}_k \middle| \mathbf{u}_k). \tag{9.1.6}$$

Using (9.1.5) and (9.1.6), the Bayes' theorem (9.1.4) can be written as

$$p(\mathbf{u}_{0:K} \mid \mathbf{z}_{1:K}) \propto p(\mathbf{u}_0)\prod_{k=1}^{K} p(\mathbf{z}_k \middle| \mathbf{u}_k)p(\mathbf{u}_k \mid \mathbf{u}_{k-1}). \tag{9.1.7}$$

This equation can be represented in a recursive form

$$p(\mathbf{u}_{0:k}|\mathbf{z}_{1:k}) \propto p(\mathbf{z}_k|\mathbf{u}_k)p(\mathbf{u}_k|\mathbf{u}_{k-1})p(\mathbf{u}_{0:k-1}|\mathbf{z}_{1:k-1}), \text{ for } k=1,2,\cdots,K. \quad (9.1.8)$$

Using this recursive form, the posterior distribution can be calculated step by step during the SDA process. Starting from $p(\mathbf{u}_0)$, after $p(\mathbf{u}_{0:k-1}|\mathbf{z}_{1:k-1})$ is obtained, we can use it to find $p(\mathbf{u}_{0:k}|\mathbf{z}_{1:k})$ for $k = 1, 2, \cdots, K$; thus, in each DA step, we only need to calculate the one-step transition PDF $p(\mathbf{u}_k|\mathbf{u}_{k-1})$ and likelihood $p(\mathbf{z}_k|\mathbf{u}_k)$, in lieu of the full posterior PDF. Equation (9.1.8) is the common base of various SDA algorithms, as we shall see throughout this chapter.

At first glance, the first-order Markov assumption used in the above derivations may seem to be rather restrictive because many physical processes can exhibit long-time memories (i.e., autocorrelation). For instance, we showed in Chap. 8 that data-driven models, such as the ARMA and ANN models, can include any number of previous states. It turns out that we can actually convert an ARMA model to a state-space model by defining a new augmented state vector, and vice versa. Example 9.3 below demonstrates such an application. Even in the case of temporally correlated model errors, it is still possible to reformulate a DA problem as a Markov process by augmenting the model state vector with model error terms (Evensen 2003).

Example 9.3 *Convert linear regression model to state-space model*
Consider the AR(p) model given in Eq. (8.1.7)

$$X_k - \sum_{i=1}^{p} a_i X_{k-i} = V_k,$$

in which the state variable and white noise are denoted by X and V, respectively, and a_i $(i = 1, \cdots, p)$ are coefficients. Let us define the state vector $\mathbf{u}_k$ as

$$\mathbf{u}_k = (X_k, X_{k-1}, \cdots, X_{k-p+1})^T,$$

and the transition matrix as

$$\mathbf{F} = \begin{pmatrix} a_1 & a_2 & \cdots & a_{p-1} & a_p \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix},$$

then the AR(p) model can be written in a discrete-time, state-space form as

$$\mathbf{u}_k = \mathbf{F}\mathbf{u}_{k-1} + \boldsymbol{\eta}_k, \qquad z_k = (1, 0, \cdots, 0)\mathbf{u}_k,$$

where $\boldsymbol{\eta}_k = (V_k, \cdots, 0)^T$. Thus, we see that $\mathbf{u}_k$ "folds" the multilag memory of the original state X_k into a first-order Markov process. Here, it is assumed that the state is observable; if not, then X is a "latent" process and the measurement equation would relate it to an observable quantity. In-depth discussion of the relation between time series models (including ARMA) and the Kalman filter can be found in Brockwell and Davis (2002, Chap. 8).

9.1.1.4 Types of DA

Depending on when and how the observations are assimilated, three types of DA problems can be formally defined:

- *Filtering*, this has been our focus of discussion so far. It is an operation that extracts information about system states using data measured up to and including the current time t_k; this amounts to estimating the posterior PDF, $p(\mathbf{u}_k \middle| \mathbf{z}_{1:k})$. According to the Markov assumption, we have the marginal PDF

$$p(\mathbf{u}_k \middle| \mathbf{z}_{1:k-1}) \propto \int p(\mathbf{u}_k \middle| \mathbf{u}_{k-1}) p(\mathbf{u}_{k-1} \middle| \mathbf{z}_{1:k-1}) d\mathbf{u}_{k-1}. \tag{9.1.9}$$

  After new measurements $\mathbf{z}_k$ arrive, we can obtain

$$\begin{aligned} p(\mathbf{u}_k \middle| \mathbf{z}_{1:k}) &= p(\mathbf{u}_k \mid \mathbf{z}_k, \mathbf{z}_{1:k-1}) \\ &\propto p(\mathbf{z}_k \mid \mathbf{u}_k, \mathbf{z}_{1:k-1}) p(\mathbf{u}_k \middle| \mathbf{z}_{1:k-1}) = p(\mathbf{z}_k \mid \mathbf{u}_k) p(\mathbf{u}_k \middle| \mathbf{z}_{1:k-1}). \end{aligned} \tag{9.1.10}$$

  This equation shows that posterior PDFs can be obtained sequentially during the SDA process. In each time step, a pair of posterior PDFs is given after the forecast step and assimilation step are implemented (i.e., we obtain sequentially $p(\mathbf{u}_1 \middle| \mathbf{z}_1)$; $p(\mathbf{u}_2 \middle| \mathbf{z}_1)$, $p(\mathbf{u}_2 \middle| \mathbf{z}_{1:2})$; $\cdots\cdots; p(\mathbf{u}_k \middle| \mathbf{z}_1), \cdots, p(\mathbf{u}_k \middle| \mathbf{z}_{1:k-1}), p(\mathbf{u}_k \middle| \mathbf{z}_{1:k})$).
- *Prediction*, which provides a priori estimates of future system states at t_{k+s} using data measured up to and including the current time t_k, where $s \geq 1$ denotes the length of lead time; this amounts to estimating the PDF $p(\mathbf{u}_{k+s} \middle| \mathbf{z}_{1:k})$.
- *Smoothing*, which provides a posteriori estimates of the system state at a previous time using all data obtained from interval $[0, t_T]$. The smoothing problem is analogous to batch estimation, off-line analysis, or 4DVAR (four-dimensional variational method). Unlike filtering and prediction, smoothing uses not only past and present information, but also "future" information relative to the actual assimilation time. It requires estimating the PDF $p(\mathbf{u}_k \middle| \mathbf{z}_{1:T})$ for $T \geq k$.

9.1.1.5 Numerical Approximation

In the simplest case, when both the forward model and measurement operators are linear, we have a linear filtering problem. Many EWR systems, however, are nonlinear, non-Gaussian, and high dimensional. Different SDA algorithms are needed depending on assumptions and approximations made regarding the prior, likelihood, transition PDF, model and measurement errors, as well as forward model and measurement operators. Common numerical approximation techniques used in SDA may be classified into the following categories:

- *Linear or higher-order approximation,* in which the forward model and measurement operators are approximated by their first-order or higher-order Taylor series expansion. The best-known filter under this category is the extended Kalman filter (EKF) (Sect. 9.3.1).
- *Deterministic sampling approximation*, which is based on the approximation of a PDF through a fixed set of sampling points. The optimal sampling points and their weights are determined a priori to approximate the first two moments of the prior PDF, and these sampling points are then propagated through the forward model operator to arrive at approximations of the posterior PDF. The best-known filter under this category is the unscented Kalman filter (Sect. 9.3.2).
- *Gaussian sum approximation*, which is based on the classical statistical principle that any non-Gaussian PDF can be approximated to some accurate degree by a sufficiently large number of Gaussian PDFs. A weighted sum of Gaussian PDFs are used for approximating the transition PDF, $p(\mathbf{u}_k \,|\, \mathbf{u}_{k-1})$, and are evolved with time. This approximation leads to the so-called Gaussian sum filters, which can be especially useful for multimodal distributions (Sect. 9.3.3).
- *Monte Carlo sampling approximation*, which is based on the idea of Monte Carlo integration, in which a PDF is approximated by a finite number of random samples drawn from its probability space. The mean of samples converges to the mean of the true PDF by the law of large numbers. Sequential Monte Carlo sampling includes a rich family of methods, which we will discuss in Sects. 9.4 and 9.5.
- *Moment approximation*, which is based on the fact that any PDF can be represented equivalently by an infinite set of its statistical moments, and for each PDF, a moment generation function exists. This method seeks to approximate the posterior PDF by a number of statistical moments, which are evolved using their recursive equations (Zeng and Zhang 2010; Kim et al. 2003).

We remark that all of the above categories, except for linear approximation, seek to approximate the PDFs, instead of the forward model and measurement operators. Also, not all filters are created equal. High dimensionality of distributed models makes the use of many filters difficult, if not infeasible. By the same token, the performance of many SDA algorithms can degrade significantly when switching from low- to high-dimensional space. Reduced-rank filters and localization techniques have been introduced, in part to circumvent these difficulties (Sect. 9.4). In addition, stochastic metamodeling approaches (e.g., polynomial chaos expansion) may be used to approximate the forward model (Li and Xiu 2009; Zhang et al. 2007). A formal discussion of metamodeling techniques will be given in Chap. 10.

9.1.2 Combined State and Parameter Estimation

In the last subsection, SDA is used to update the model output vector $\mathbf{u}$ defined in (9.1.1) and (9.1.2), where model input vector $\boldsymbol{\theta}$ is assumed known. In fact, the information contained in the new measurements on $\mathbf{u}$ and/or $\boldsymbol{\theta}$ can be used to update both of them. In other words, we can use SDA to update the entire model simultaneously, rather than consider the forward and inverse solutions separately. Hereafter, as explained in Sect. 1.2.3, we will assume that $\boldsymbol{\theta}$ consists of only the unknown or poorly known model parameters, initial/boundary conditions, and sink/source terms; all perfectly known model inputs will not be shown explicitly in the model equations; and both state vector **u** and parameter vector $\boldsymbol{\theta}$ are considered as stochastic processes. In this subsection, we will derive the joint PDF of them and formulate a joint SDA problem. This problem generally leads to strongly nonlinear optimization problems, even when the system models are linear.

9.1.2.1 Joint Posterior Distribution of Model States and Parameters

Let us extend the Bayesian view of SDA considered in the last section to include the estimation of both model states **u** and model parameters $\boldsymbol{\theta}$. In this case, the Bayes' theorem (9.1.4) becomes

$$\begin{aligned} p(\mathbf{u}_{0:K}, \boldsymbol{\theta}_{0:K} \mid \mathbf{z}_{1:K}) &\propto p(\mathbf{z}_{1:K} \mid \mathbf{u}_{0:K}, \boldsymbol{\theta}_{0:K}) p(\mathbf{u}_{0:K}, \boldsymbol{\theta}_{0:K}) \\ &= p(\mathbf{z}_{1:K} \mid \mathbf{u}_{0:K}, \boldsymbol{\theta}_{0:K}) p(\mathbf{u}_{0:K} \mid \boldsymbol{\theta}_{0:K}) p(\boldsymbol{\theta}_{0:K}), \end{aligned} \tag{9.1.11}$$

where $\mathbf{u}_{0:K} = \{\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_K\}$, $\boldsymbol{\theta}_{0:K} = \{\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_K\}$, and $\mathbf{z}_{1:K} = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_K\}$. Assume that both $\mathbf{u}_{0:K}$ and $\boldsymbol{\theta}_{0:K}$ are Markov chains, and from (9.1.1), for any *k*, $\mathbf{u}_k$ depends only on $\mathbf{u}_{k-1}$ and $\boldsymbol{\theta}_{k-1}$, and (9.1.11) can then be rewritten as

$$\begin{aligned} p(\mathbf{u}_{0:K}, \boldsymbol{\theta}_{0:K} \mid \mathbf{z}_{1:K}) &\propto \prod_{k=1}^{K} p(\mathbf{z}_k \mid \mathbf{u}_k, \boldsymbol{\theta}_k) \times p(\mathbf{u}_0) \prod_{k=1}^{K} p(\mathbf{u}_k \mid \mathbf{u}_{k-1}, \boldsymbol{\theta}_{k-1}) \\ &\times p(\boldsymbol{\theta}_0) \prod_{k=1}^{K} p(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{k-1}). \end{aligned} \tag{9.1.12}$$

This equation, in turn, can be represented in a recursive form for $k = 1, 2, \cdots, K$:

$$\begin{aligned} p(\mathbf{u}_{0:k}, \boldsymbol{\theta}_{0:k} \mid \mathbf{z}_{1:k}) &\propto p(\mathbf{z}_k \mid \mathbf{u}_k, \boldsymbol{\theta}_k) p(\mathbf{u}_k \mid \mathbf{u}_{k-1}, \boldsymbol{\theta}_{k-1}) p(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{k-1}) \\ &\times p(\mathbf{u}_{0:k-1}, \boldsymbol{\theta}_{0:k-1} \mid \mathbf{z}_{1:k-1}). \end{aligned} \tag{9.1.13}$$

Because

$$p(\mathbf{u}_k, \boldsymbol{\theta}_k \mid \mathbf{u}_{k-1}, \boldsymbol{\theta}_{k-1}) = p(\mathbf{u}_k \mid \mathbf{u}_{k-1}, \boldsymbol{\theta}_{k-1}) p(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{k-1}),$$

the expression in (9.1.13) can be simplified to

$$\begin{aligned} p(\mathbf{u}_{0:k}, \boldsymbol{\theta}_{0:k} \mid \mathbf{z}_{1:k}) &\propto p(\mathbf{z}_k \mid \mathbf{u}_k, \boldsymbol{\theta}_k) p(\mathbf{u}_k, \boldsymbol{\theta}_k \mid \mathbf{u}_{k-1}, \boldsymbol{\theta}_{k-1}) \\ &\times p(\mathbf{u}_{0:k-1}, \boldsymbol{\theta}_{0:k-1} \mid \mathbf{z}_{1:k-1}). \end{aligned} \tag{9.1.14}$$

As this point, we may introduce an augmented state variable $\mathbf{y}$

$$\mathbf{y} \triangleq \begin{pmatrix} \mathbf{u} \\ \boldsymbol{\theta} \end{pmatrix}. \tag{9.1.15}$$

The posterior PDF in (9.1.14) can then be written in terms of $\mathbf{y}$ as

$$p(\mathbf{y}_{0:k} | \mathbf{z}_{1:k}) \propto p(\mathbf{z}_k | \mathbf{y}_k) p(\mathbf{y}_k | \mathbf{y}_{k-1}) p(\mathbf{y}_{0:k-1} | \mathbf{z}_{1:k-1}), \; (k = 1, 2, \cdots, K) \tag{9.1.16}$$

We see that (9.1.16) is exactly identical to (9.1.8) when $\mathbf{u}$ is replaced by $\mathbf{y}$. Therefore, filter algorithms used for estimating state variables $\mathbf{u}$ can be used directly to estimate $\mathbf{u}$ and $\boldsymbol{\theta}$ jointly by replacing $\mathbf{u}$ in the filter with $\mathbf{y}$.

9.1.2.2 Parameter Evolution Equation

In order to estimate the augmented state $\mathbf{y}$ through DA, the prediction equation (9.1.1) and measurement equation (9.1.2) must be augmented to include prediction and measurement equations for $\boldsymbol{\theta}$. These equations are case dependent. For EWR modeling, in most cases, $\boldsymbol{\theta}$ is considered time-invariant or slowly varying parameters. In this case, a commonly used parameter prediction equation has the following form:

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \boldsymbol{\upsilon}_k, \tag{9.1.17}$$

where $\boldsymbol{\upsilon}_k$ is an error term. Equation (9.1.17) is known as the slowly time-varying or drifting parameter for the combined state and parameter estimation problem (Doucet and Tadić 2003). Different rationales for using (9.1.17) have been suggested, depending on the application at hand.

One school of thoughts suggests that $\boldsymbol{\theta}_k$ be treated as a stationary stochastic process driven by a zero-mean artificial noise $\boldsymbol{\upsilon}_k$ (Gordon et al. 1993; Liu and West 2001; Lopes and Tsay 2011) such that $\boldsymbol{\theta}$ becomes a random walk process. Along this line of thinking, van de Merwe (2004) suggested that the covariance structure of parameter error $\boldsymbol{\upsilon}$, $\mathbf{Q}_{\boldsymbol{\upsilon}}$, takes one of the following parametric forms:

- The diagonal structure

$$\text{diag}(\mathbf{Q}_{\boldsymbol{\upsilon}}) = \sigma_{\boldsymbol{\upsilon}}^2, \tag{9.1.18}$$

- The exponential decay structure

$$\mathbf{Q}_{\boldsymbol{\upsilon},k} = (\lambda^{-1} - 1)\mathbf{P}_k = \beta \mathbf{P}_k, \tag{9.1.19}$$

where $\mathbf{P}_k$ is the state covariance at t_k and λ is a "forgetting" factor that imposes exponentially decaying weighting on past data. We can define a new constant,

$\beta \triangleq \lambda^{-1} - 1$, where the role of β is similar to that of the scaling factor used in *covariance inflation*, which is a commonly used heuristic technique to counteract the effect of model errors (see Sect. 9.4). The need for covariance inflation is based on the argument that covariance structure is correct, but the sample variance is too small (Anderson and Anderson 1999).

- Robbins–Monro stochastic approximation

$$\mathbf{Q}_{\boldsymbol{\upsilon},k} = (1-\alpha)\mathbf{Q}_{\boldsymbol{\upsilon},k-1} + \alpha \mathbf{d}_k{}^T \mathbf{d}_k, \tag{9.1.20}$$

where $\mathbf{d}_k$ is the innovation vector (i.e., difference between measurements and prediction) and α is a weighting factor. Equation (9.1.20) approximates the parameter covariance as a weighted sum of the parameter covariance from the previous step and the innovation covariance, $\mathbf{d}_k{}^T\mathbf{d}_k$. Note that (9.1.20) is in the form of regularized objective function we have seen in the shrinkage methods (Sect. 8.1) and SVR (Sect. 8.3).

Parameters σ, λ, and α appearing in (9.1.18)–(9.1.20) can be regarded as additional hyperparameters that need to be tuned adaptively during the DA process. The level of artificial noise $\boldsymbol{\upsilon}$ should be such that a balance is maintained between filter robustness and estimation accuracy. Too much artificial noise will lead to an overly diffuse posterior PDF for $\boldsymbol{\theta}$. Another issue to keep in mind is that the added artificial noise will distort the approximated posterior PDFs. Selection of the appropriate covariance structures and hyperparameters is application specific.

Another school of thoughts suggests that $\mathbf{Q}_{\boldsymbol{\upsilon}}$ should be used to represent parameter uncertainty, but not artificial noise. Parameter uncertainty includes both a reducible component and an irreducible component, as we will learn in Chap. 10. The reducible uncertainty component motivates modelers to reduce uncertainty by gathering more data or by refining models. Then, (9.1.17) signifies that $\boldsymbol{\theta}$ is updated in an iterative manner starting from some initial guess. Of course, "iteration" in the current context means that new information is processed to update the state of knowledge on parameters. Such was the point of view held by Evensen (2007) when presenting the ensemble Kalman filter (EnKF) algorithm for joint state and parameter estimation (see Sect. 9.4). We remark that similar ideas have long been used in geostatistical community. For example, a sequential cokriging method was introduced to the stochastic hydrogeology literature in 1990s for solving combined parameter and state estimation problems (Sun and Yeh 1992; Vargas-Guzmán and Yeh 1999; Harvey and Gorelick 1995) and later incorporated in other geostatistical inversion problems. Detailed discussion of these topics has been provided in Sect. 7.4.

Let us denote the augmented error vector as $\tilde{\boldsymbol{\eta}} \triangleq (\boldsymbol{\eta}^T \quad \boldsymbol{\upsilon}^T)^T$, where $\boldsymbol{\eta}$ and $\boldsymbol{\upsilon}$ denote model and parameter errors, respectively. The covariance matrix of $\tilde{\boldsymbol{\eta}}$ is a block matrix

$$\mathbf{Q}_{\tilde{\boldsymbol{\eta}}} = \begin{pmatrix} \mathbf{Q}_{\boldsymbol{\eta}} & \mathbf{C}_{\boldsymbol{\eta}\boldsymbol{\upsilon}} \\ \mathbf{C}_{\boldsymbol{\eta}\boldsymbol{\upsilon}}^T & \mathbf{Q}_{\boldsymbol{\upsilon}} \end{pmatrix}, \tag{9.1.21}$$

where $\mathbf{Q}$ and $\mathbf{C}$ represent, respectively, the auto- and cross-covariance matrices. Cross-covariance allows interactions between model states and parameters to be modeled and only appears in joint state and parameter filters. Quantification of the covariance and cross-covariance matrices in (9.1.21) is essential. We can identify the following three cases for parameter error specification.

- Case I. The model is subjected to white noise, and parameters are unknown, but constant. In this simple case, $\boldsymbol{\upsilon}$ is artificial and is mainly used as a heuristic measure to improve filter stability. Further, if it can be assumed that $\boldsymbol{\eta}$ and $\boldsymbol{\upsilon}$ are uncorrelated, $\mathbf{Q}_{\tilde{\boldsymbol{\eta}}}$ becomes a diagonal block matrix. The error covariance $\mathbf{Q}_{\boldsymbol{\upsilon}}$ can be modeled using one of the structures provided in (9.1.18) to (9.1.20). Again, $\mathbf{Q}_{\boldsymbol{\upsilon}}$ should be small enough to avoid interference with estimates, but big enough to alleviate filter degeneracy. For reasons mentioned previously, the Robbins–Monro stochastic approximation is expected to be the most robust structure because of its resemblance to regularized objective function. EWR applications that adopt the artificial noise approach include Moradkhani et al. (2005) who assimilated hyperparameters in a lumped-parameter streamflow model, and Gove and Hollinger (2006) who assimilated net CO_2 exchange data into a physiological model with uncertain parameters.
- Case II. Model state error is mainly caused by uncertain parameters, and parameter error structures are deterministic. In this case, $\boldsymbol{\eta}$ and $\boldsymbol{\upsilon}$ are highly correlated, and both have a spatial structure if parameters are distributed. Initially, the prior PDF of $\boldsymbol{\theta}$ is estimated from data to fit to a parameter structure which may be, for example, any of the covariance structures introduced in Sect. 6.2. Starting with the prior of $\boldsymbol{\theta}$, we can recursively estimate all other elements in (9.1.21) by solving either stochastic moment equations or by using a Monte Carlo-based method at each assimilation step. Using the stochastic moment equation approach, one needs to derive and solve coupled stochastic PDEs for auto- and cross-covariance (e.g., McLaughlin and Townley 1996; Sun and Yeh 1992; Zhang 2002; Graham and McLaughlin 1989; Xiu 2010). In contrast, Monte Carlo-based methods typically operate with a large ensemble of sample points or particles, and the population statistics are approximated by ensemble statistics. The efficacy of Monte Carlo methods depends on the statistical representativeness of the ensemble, as well as the feasibility to drive the "swarm" of particles to the true posterior PDF. During SDA, $\mathbf{Q}_{\boldsymbol{\upsilon}}$ is continuously updated by measurements and acts as a regularization mechanism. The latter point can be clarified by using the posterior PDF of $\boldsymbol{\theta}$

$$\begin{aligned} p(\boldsymbol{\theta}|\mathbf{z}) &\propto p(\mathbf{z}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \\ &\propto \exp\left(-\frac{1}{2}(\mathbf{z}-\mathbf{g}(\mathbf{u},\boldsymbol{\theta}))^T\mathbf{R}^{-1}(\mathbf{z}-\mathbf{g}(\mathbf{u},\boldsymbol{\theta}))\right) \\ &\times\exp\left(-\frac{1}{2}(\boldsymbol{\theta}-\bar{\boldsymbol{\theta}})^T\mathbf{Q}_{\boldsymbol{\upsilon}}^{-1}(\boldsymbol{\theta}-\bar{\boldsymbol{\theta}})\right) \end{aligned} \tag{9.1.22}$$

where the likelihood and prior in (9.1.22) were obtained by assuming Gaussian error distributions,

$$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}, \mathbf{Q}_{\upsilon}) \text{ and } p(\mathbf{z} - \mathbf{g}(\mathbf{u}, \boldsymbol{\theta})) = \mathcal{N}(\mathbf{0}, \mathbf{R}).$$

In theory, there is no need for artificial noise in this case. Complexity, however, can arise because of (i) system nonlinearity and (ii) errors in prior statistics caused by, for example, using sampling statistics to approximate population statistics. Researchers have proposed to "repair" problematic covariance matrices by adding artificial noise or by inflating the covariance. A fundamental issue, however, is probably rooted in the higher order interactions among parameters, which are not captured by low-order moments. Therefore, ad hoc fixes of covariance matrices will only have limited effect. Case II has been considered in a large number of EWR applications involving estimation of distributed parameters (e.g., Evensen 2009; Chen and Zhang 2006; Oliver and Chen 2011; Sun et al. 2009a; Agbalaka and Oliver 2008; Xie and Zhang 2010; Elsheikh et al. 2012; Zhou et al. 2011; Zhang et al. 2007; Gu and Oliver 2005; Vrugt and Robinson 2007).

- Case III. Both model structure error and parameter uncertainty exist. Here, the model structure error is caused by conceptualization error. It is fair to say that Case III is probably the most realistic and, unfortunately, the least understood scenario. The fundamental challenge stems from the specification of the prior PDF for model states which, in theory, requires consideration of an infinite number of model classes (a model class defines a group of models having similar mathematical forms, see also Sect. 10.6). Therefore, even with the advent of easily accessible high-performance computing facilities, online analyses are still limited to special cases. One such example is the identification of the uncertain hyperparameters of a parameter structural model through separate MCMC (Del Moral et al. 2006; Polson et al. 2008) or point estimators (Andrieu et al. 2005; Doucet and Tadić 2003). Another example is the multiple model particle filter or reverse-jump MCMC that are capable of jumping or switching between models (Doucet et al. 2001b; Ristic et al. 2004).

For reasons mentioned at the beginning of this chapter, our main focus herein is on Cases I and II.

9.1.2.3 Formulation of the Joint SDA Problem

After the above discussion, let us formulate a joint SDA problem to be solved by various filters introduced in this chapter. Equations (9.1.1) and (9.1.2) will be replaced, respectively, by the following augmented state form:

$$\begin{aligned} \mathbf{y}_k &= \tilde{\mathbf{f}}(\mathbf{y}_{k-1}) + \tilde{\boldsymbol{\eta}}_k \\ \mathbf{z}_k &= \tilde{\mathbf{g}}(\mathbf{y}_k) + \tilde{\boldsymbol{\varepsilon}}_k \end{aligned} \tag{9.1.23}$$

where variable $\mathbf{y}$ may consist of model states $\mathbf{u}$ to be predicted, model parameters $\boldsymbol{\theta}$ to be estimated, or both of them. Accordingly, the forward model operator $\tilde{\mathbf{f}}(\cdot)$ may consist of both state and parameter evolution equations, the measurement operator $\tilde{\mathrm{g}}(\cdot)$ may contain both state and parameter measurements, and the error vector $\tilde{\boldsymbol{\eta}}$ is set accordingly using $\boldsymbol{\eta}$ and $\boldsymbol{\upsilon}$.

In the prediction stage of (9.1.23), the current parameter values are used in the forward solution model to generate the state predictions, and after the parameter values are updated in the assimilation stage, these updated parameter values are immediately used in the next time step for model prediction. The dependence of $\mathbf{u}$ on $\boldsymbol{\theta}$ is directly incorporated into the DA process.

The following common assumptions are needed in the derivation of DA filters for (9.1.23):

- The augmented process error $\tilde{\boldsymbol{\eta}}_k$ is unbiased (i.e., zero mean) and temporally uncorrelated,

$$E(\tilde{\boldsymbol{\eta}}_k) = 0$$
$$E(\tilde{\boldsymbol{\eta}}_i \tilde{\boldsymbol{\eta}}_j^T) = \begin{cases} \mathbf{Q}_k, & i = j \\ \mathbf{0}, & i \neq j \end{cases} \tag{9.1.24}$$

  where the covariance matrix, $\mathbf{Q}_k$, has a block structure as discussed in Sect. 9.1.2.

- The measurement error $\tilde{\boldsymbol{\varepsilon}}_k$ is unbiased and temporally uncorrelated,

$$E(\tilde{\boldsymbol{\varepsilon}}_i) = 0$$
$$E(\tilde{\boldsymbol{\varepsilon}}_i \tilde{\boldsymbol{\varepsilon}}_j^T) = \begin{cases} \mathbf{R}_k, & i = j \\ \mathbf{0}, & i \neq j \end{cases} \tag{9.1.25}$$

- $\tilde{\boldsymbol{\varepsilon}}_k$ is uncorrelated with $\tilde{\boldsymbol{\eta}}_k$ and the initial state $\mathbf{y}_0$.
- The covariance of initial state $\mathbf{y}_0$ is known.

Example 9.4 *Set up a state-space model for 1D porous flow*
Consider 1D flow in a confined porous medium. The governing PDE is

$$S_s \frac{\partial h}{\partial t} = K \frac{\partial^2 h}{\partial^2 x} + q_s \tag{9.1.26}$$

subject to initial and boundary conditions

$$h(t = 0) = h_0,\ h(x = 0) = H_0,\ h(x = L) = H_L$$

where S_s is specific storage, h is the hydraulic head, K is hydraulic conductivity, q_s is a distributed sink/source term, L is the length of the column, and H_0 and H_L are constant head boundaries. A numerical method can be used to find the forward solution of this model. When the finite difference method is used, (9.1.26) is discretized into

$$S_s \frac{h_i^{t+\Delta t} - h_i^t}{\Delta t} = K \frac{h_{i-1}^{t+\Delta t} - 2h_i^{t+\Delta t} + h_{i+1}^{t+\Delta t}}{\Delta x^2} + q_{s,i},\ (i = 1, 2, \cdots, L-1), \tag{9.1.27}$$

where the superscript denotes time step and the subscript labels the coordinate indices; Δt is temporal discretization interval; Δx is spatial discretization interval; and $\{h_i\}_{i=1}^{L-1}$ denote unknown nodal values.

After incorporating boundary conditions into (9.1.27), the head distribution at time step *k*, $\mathbf{h}_k$, can be obtained from $\mathbf{h}_{k-1}$ by solving

$$\mathbf{A}\mathbf{h}_k = a_1\mathbf{h}_{k-1} + \mathbf{c}_k + \mathbf{q}_k, \tag{9.1.28}$$

where $a_1 = S_s / \Delta t$ and

$$\mathbf{A} = \begin{pmatrix} a_1 + 2a_2 & -a_2 & & \\ -a_2 & a_1 + 2a_2 & -a_2 & \\ \cdots & \cdots & \cdots & \cdots \\ & & -a_2 & a_1 + 2a_2 \end{pmatrix}, \mathbf{c}_k = \begin{pmatrix} a_2 H_0 \\ 0 \\ \vdots \\ a_2 H_L \end{pmatrix}, \mathbf{q}_k = \begin{pmatrix} q_{s,1} \\ q_{s,2} \\ \vdots \\ q_{s,L-1} \end{pmatrix}, \tag{9.1.29}$$

where $a_2 = K / \Delta x^2$, $\mathbf{c}_k$ contains known boundary values and $\mathbf{q}_k$ is the unknown distributed sink/source term that we wish to estimate. Both $\mathbf{c}_k$ and $\mathbf{q}_k$ are assumed temporally invariant. From (9.1.28), the forward solution of the model is

$$\mathbf{h}_k = \mathbf{A}^{-1}(a_1\mathbf{h}_{k-1} + \mathbf{q}_k + \mathbf{c}_k). \tag{9.1.30}$$

To perform joint SDA, we can define the augmented state vector as

$$\mathbf{y} \triangleq \begin{pmatrix} \mathbf{h} \\ \mathbf{q} \end{pmatrix}.$$

In this case, the prediction equation in (9.1.23) becomes a linear system given by

$$\begin{aligned} &\begin{pmatrix} \mathbf{h}_k \\ \mathbf{q}_k \end{pmatrix} = \begin{pmatrix} a_1\mathbf{A}^{-1} & \mathbf{A}^{-1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{h}_{k-1} \\ \mathbf{q}_{k-1} \end{pmatrix} + \begin{pmatrix} \mathbf{A}^{-1}\mathbf{c}_k \\ \mathbf{0} \end{pmatrix} + \tilde{\boldsymbol{\eta}}_k, \\ &\text{or } \mathbf{y}_k = \tilde{\mathbf{M}}\mathbf{y}_{k-1} + \tilde{\mathbf{c}}_k + \tilde{\boldsymbol{\eta}}_k. \end{aligned} \tag{9.1.31}$$

When both $\mathbf{h}$ and $\mathbf{q}$ can be measured directly, the measurement equation in (9.1.23) is also a linear system

$$\begin{pmatrix} \mathbf{z}_{\mathbf{h},k} \\ \mathbf{z}_{\mathbf{q},k} \end{pmatrix} = \begin{pmatrix} \mathbf{G}_{\mathbf{h}} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_{\mathbf{q}} \end{pmatrix} \begin{pmatrix} \mathbf{h}_k \\ \mathbf{q}_k \end{pmatrix} + \begin{pmatrix} \boldsymbol{\varepsilon}_{h,k} \\ \boldsymbol{\varepsilon}_{q,k} \end{pmatrix}, \quad \text{or } \mathbf{z}_k = \tilde{\mathbf{G}}\mathbf{y}_k + \tilde{\boldsymbol{\varepsilon}}_k \tag{9.1.32}$$

where $\mathbf{G}_{\mathbf{h}}$ and $\mathbf{G}_{\mathbf{q}}$ are matrices of 0's and 1's, with 1's corresponding to **h** measurement and **q** measurement locations, respectively. In the next section, we will show how the linear SDA problems such as (9.1.31) and (9.1.32) can be solved. ■

Note that for clarity, we will drop the tilde symbol in (9.1.23) hereafter where no confusion should occur. Thus, all filters introduced in the subsequent sections will use the following *extended, discrete-time state-space model* for joint state and parameter estimation:

$$\begin{aligned}\mathbf{y}_k &= \mathbf{f}(\mathbf{y}_{k-1}) + \boldsymbol{\eta}_k,\\ \mathbf{z}_k &= \mathbf{g}(\mathbf{y}_k) + \boldsymbol{\varepsilon}_k.\end{aligned} \tag{9.1.33}$$

9.2 The Kalman Filter

The KF, originally introduced by Rudolph E. Kalman (Kalman 1960), is the most well-known SDA method for linear models and serves as the building block for many modern SDA methods. The basic KF theory for state assimilation has been covered by many textbooks (Lewis et al. 2006; Brockwel and Davis 2002).

In (9.33), when $\mathbf{f}(\mathbf{y}_{k-1}) = \mathbf{M}_k\mathbf{y}_{k-1} + \mathbf{c}_k$, where $\mathbf{c}_k$ is a known control (or forcing) term, and $\mathbf{g}(\mathbf{y}_k) = \mathbf{G}_k\mathbf{y}_k$, we have a *linear stochastic dynamic system*

$$\mathbf{y}_k = \mathbf{M}_k\mathbf{y}_{k-1} + \mathbf{c}_k + \boldsymbol{\eta}_k, \tag{9.2.1}$$

$$\mathbf{z}_k = \mathbf{G}_k\mathbf{y}_k + \boldsymbol{\varepsilon}_k. \tag{9.2.2}$$

The purpose of KF is using the forward model and measurements $\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_{k-1}, \mathbf{z}_k, \cdots\}$ to find the best estimates $\{\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \cdots, \hat{\mathbf{y}}_{k-1}, \hat{\mathbf{y}}_k, \cdots\}$ of the true but unknown system states $\{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_{k-1}, \mathbf{y}_k, \cdots\}$, as well as the uncertainty of estimation measured by posterior covariance matrices $\{\hat{\mathbf{P}}_1, \hat{\mathbf{P}}_2, \cdots, \hat{\mathbf{P}}_{k-1}, \hat{\mathbf{P}}_k, \cdots\}$. The initial state $\mathbf{y}_0$ and its covariance $\mathbf{P}_0$ are assumed known.

As a SDA filter, KF is a recursive process: Starting from the known initial pair $(\hat{\mathbf{y}}_0 = \mathbf{y}_0, \hat{\mathbf{P}}_0 = \mathbf{P}_0)$, find the estimates of unknown pair $(\hat{\mathbf{y}}_1, \hat{\mathbf{P}}_1)$ using data $\mathbf{z}_1$, and then from $(\hat{\mathbf{y}}_1, \hat{\mathbf{P}}_1)$ find $(\hat{\mathbf{y}}_2, \hat{\mathbf{P}}_2)$ using data $\mathbf{z}_2$, and so forth. In general, a single KF step of obtaining $(\hat{\mathbf{y}}_k, \hat{\mathbf{P}}_k)$ from $(\hat{\mathbf{y}}_{k-1}, \hat{\mathbf{P}}_{k-1})$ consists of two stages, the forecast stage (forward solution) and the assimilation stage (inverse solution), as shown in Fig. 9.1 and described in detail below:

- *KF forecast.* Using model (9.2.1) to obtain a prediction pair

$$\begin{aligned}\mathbf{y}_k^f &= \mathbf{M}_k\hat{\mathbf{y}}_{k-1} + \mathbf{c}_k\\ \mathbf{P}_k^f &= \mathbf{M}_k\hat{\mathbf{P}}_{k-1}\mathbf{M}_k^T + \mathbf{Q}_k\end{aligned} \tag{9.2.3}$$

  where $\mathbf{Q}_k$ is the covariance of $\boldsymbol{\eta}_k$ defined in (9.1.24).

- *KF assimilation*: Using measurement model (9.2.2) to modify $\mathbf{y}_k^f$. This is exactly the statistical inverse problem of a linear model that we have considered in Sect. 4.2.3, but here $\mathbf{y}_k$ is the unknown, $\mathbf{y}_k^f$ is its initial guess, $\mathbf{P}_k^f$ is the prior covariance matrix, and $\mathbf{z}_k$ is the observation data to be used for estimation. According to Eqs. (4.2.21) and (4.2.22), the optimal estimation and the covariance matrix of estimation are given by

$$\begin{aligned}\hat{\mathbf{y}}_k &= \mathbf{y}_k^f + \mathbf{P}_k^f\mathbf{G}_k^T(\mathbf{G}_k\mathbf{P}_k^f\mathbf{G}_k^T + \mathbf{R}_k)^{-1}(\mathbf{z}_k - \mathbf{G}_k\mathbf{y}_k^f)\\ \hat{\mathbf{P}}_k &= \mathbf{P}_k^f - \mathbf{P}_k^f\mathbf{G}_k^T(\mathbf{G}_k\mathbf{P}_k^f\mathbf{G}_k^T + \mathbf{R}_k)^{-1}\mathbf{G}_k\mathbf{P}_k^f\end{aligned} \tag{9.2.4}$$

where $\mathbf{R}_k$ is the covariance matrix of $\boldsymbol{\varepsilon}_k$ defined in (9.1.25). The above equation can be rewritten as

$$\begin{aligned}\hat{\mathbf{y}}_k &= \mathbf{y}_k^f + \mathbf{K}_k(\mathbf{z}_k - \mathbf{G}_k\mathbf{y}_k^f)\\ \hat{\mathbf{P}}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{G}_k)\mathbf{P}_k^f\end{aligned} \tag{9.2.5}$$

where

$$\mathbf{K}_k = \mathbf{P}_k^f\mathbf{G}_k^T(\mathbf{G}_k\mathbf{P}_k^f\mathbf{G}_k^T + \mathbf{R}_k)^{-1} \tag{9.2.6}$$

is called the *Kalman gain* matrix. It is easy to show $\mathbf{P}_{\mathbf{y}_k\mathbf{z}_k}^f = \mathbf{P}_k^f\mathbf{G}_k^T$ and $\mathbf{P}_{\mathbf{z}_k\mathbf{z}_k}^f = \mathbf{G}_k\mathbf{P}_k^f\mathbf{G}_k^T + \mathbf{R}_k$, where $\mathbf{P}_{\mathbf{y}_k\mathbf{z}_k}^f$ is the cross-covariance matrix between $\mathbf{y}_k^f$ and $\mathbf{z}_k$. Thus, the Kalman gain matrix can also be written as

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{y}_k\mathbf{z}_k}^f(\mathbf{P}_{\mathbf{z}_k\mathbf{z}_k}^f)^{-1}. \tag{9.2.7}$$

After the pair $(\hat{\mathbf{y}}_k, \hat{\mathbf{P}}_k)$ is found in (9.2.5), we can move to the next time step to find $(\hat{\mathbf{y}}_{k+1}, \hat{\mathbf{P}}_{k+1})$, and so forth.

The length of each assimilation time interval is determined by consistency and stability of the particular spatial and temporal discretization schemes used for deriving a state-space model. To maintain numerical stability, the interval length of t_k is typically small compared to the time window at which successive sets of observations are available (i.e., in those steps without an measurement update, the forward model is simply run to evolve the state).

The a posteriori error covariance, $\hat{\mathbf{P}}_k$, provides a measure of prediction uncertainty at t_k. Propagation of the state covariance $\mathbf{P}_k$ represents the most computationally expensive part of the KF, which becomes even more demanding when the state variable is augmented. From (9.2.5), the Kalman gain determines the uncertainty reduction from the prior covariance $\hat{\mathbf{P}}_k^f$ to the posterior covariance $\hat{\mathbf{P}}_k$, or the amount of information that is extracted from the new measurements $\mathbf{z}_k$. For further interpretation, we use a simpler form of $\mathbf{K}_k$ (Lewis et al. 2006)

$$\mathbf{K}_k = \mathbf{P}_k\mathbf{G}_k^T\mathbf{R}_k^{-1}. \tag{9.2.8}$$

Equation (9.2.8) suggests that $\mathbf{K}_k$ depends on the state covariance and the inverse of measurement covariance. If the measurement errors are large but model and parameters are more certain, $\mathbf{K}_k$ assigns more weight to the model state and vice versa. In practice, however, we seldom have access to actual $\mathbf{Q}$ and $\mathbf{R}$. A rule of thumb is to use more conservative estimates for both to reduce filter divergence and, thus, improve filter stability (Cohn 1997).

For linear systems that is subjected to Gaussian errors, the KF gives unbiased, minimum variance estimate of the state variable. The KF estimates are equivalent to those obtained from a linear Gaussian estimator because both are also maximum-likelihood estimators (McLaughlin and Townley 1996). The KF is not robust and can be affected by outliers in data. If the Gaussianity requirement is removed, the KF still remains an optimal filter in the mean square sense, but its solution is no longer unbiased.

Example 9.5 *Estimate boundary condition using the KF*
Now, assume that the left-hand-side boundary condition (H_0) of the 1D system in Example 9.4 is uncertain and needs to be estimated through SDA. All other parameters are known. We can adopt the evolution equation (9.17) for H_0

$$H_{0,k} = H_{0,k-1} + \upsilon_k, \tag{9.2.9}$$

where υ is white noise. If we define the augmented state vector $\mathbf{y}_k$ as

$$\mathbf{y}_k \triangleq \begin{pmatrix} \mathbf{h}_k \\ H_{0,k} \end{pmatrix}.$$

The state transition matrix $\mathbf{M}_k$ in (9.2.1) becomes

$$\mathbf{M}_k = \begin{pmatrix} a_1\mathbf{A}^{-1} & \mathbf{A}^{-1}\mathbf{b} \\ \mathbf{0} & 1 \end{pmatrix}, \tag{9.2.10}$$

where $\mathbf{b} = [a_2, 0, \cdots, 0]^T$. In this example, the sink/source term is known and is merged into the control term $\mathbf{c}_k$ in (9.2.1) as

$$\mathbf{c}_k = \begin{pmatrix} \mathbf{A}^{-1}\mathbf{q}^\dagger \\ 0 \end{pmatrix}, \tag{9.2.11}$$

where the vector $\mathbf{q}^\dagger$ is defined by

$$\mathbf{q}^\dagger = \begin{pmatrix} q_{s,1} \\ q_{s,2} \\ \vdots \\ a_2 H_L + q_{s,L-1} \end{pmatrix}.$$

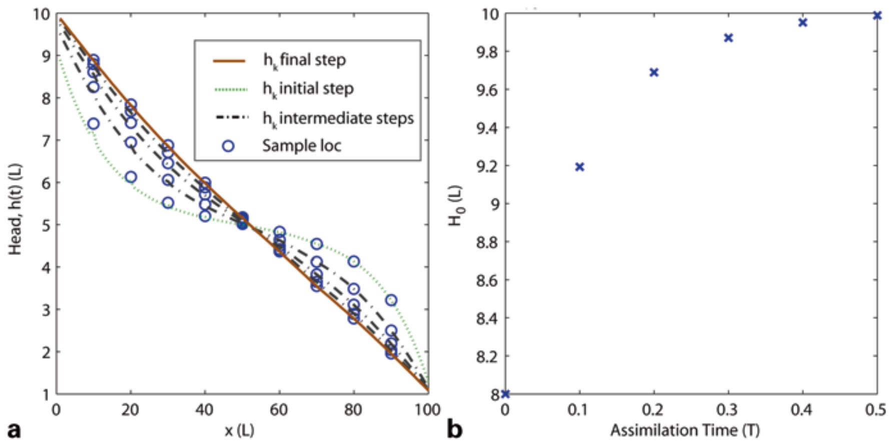


Fig. 9.2 **a** Evolution of head distribution $\mathbf{h}_k$ with time, where the *green dotted line* represents head distribution after the first update step, *brown solid line* represents the final head distribution, *dark dashed lines* in between represent outputs from the intermediate steps, and open circles indicate sample locations; Plot of the assimilated value of unknown boundary condition, H_0, at different assimilation times

The error covariance matrices assume the following structures:

$$\mathbf{Q}_{\tilde{\eta}} = \begin{pmatrix} \sigma_\eta^2 \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \sigma_\upsilon^2 \end{pmatrix}, \quad \mathbf{P}_0 = \begin{pmatrix} \mathbf{P}_{h,0} & \mathbf{0} \\ \mathbf{0} & \sigma_{H_0}^2 \end{pmatrix}, \quad \mathbf{R} = \sigma_R^2 \mathbf{I}$$

in which σ_υ^2 and $\sigma_{H_0}^2$ are variances of υ and H_0, respectively. For demonstration, let us assume the following data for the problem at hand:

$$H_L = 1\,[\mathrm{L}],\ S_s = 1 \times 10^{-5}\ [\mathrm{L}^{-1}],\ K = 0.0183\ [\mathrm{L/T}],$$

$$\sigma_\eta = 0.1,\ \sigma_\upsilon = 0.1,\ \sigma_{H_0} = 0.316,\ \sigma_R = 0.1,\ \mathbf{P}_{h,0} = [\mathbf{0}]_{(L-1)\times(L-1)}.$$

The total length of the 1D system is 100 [L] and the spatial discretization is $\Delta x = 1\,[\mathrm{L}]$. A single pumping well is installed at $x = 70$ [L] and pumps at constant rate of 2×10^{-4} [L/T]. Measurements are taken every 10 [L] along the domain. The total assimilation time is 0.5 [T], and the sampling interval is $\Delta t = 0.1$ [T], which is the same as the assimilation interval. Let us assume that the actual value of H_0 is 10 [L], but the initial guess of H_0 is 8 [L]. Initially, the head is assumed to be 5 [L] everywhere in the domain.

Figure 9.2a shows the evolution of hydraulic head profile **h** with time, and Fig. 9.2b shows the assimilated H_0 with time. Starting with the initial guess, the actual value of H_0, 10.0 [L], is obtained after only five assimilation steps. For this example, the variances are assigned through trial and error. In practice, these variances may be regarded as hyperparameters that need to be adjusted simultaneously with the unknown parameters. Two KF functions, `kf_predict` and `kf_update`, from the MATLAB toolbox EKFUKF (Hartikainen and Särkkä 2008) were used to perform the SDA.

9.3 Nonlinear System Filtering

The KF, when used for joint state and parameter estimation, is subject to several limitations. Firstly, the filtering problem generally becomes nonlinear even for linear systems. In Example 9.4, the left-hand side involves the product of unknown state h and the hydraulic conductivity K (not to be confused with the bold-faced Kalman gain matrix). Thus, if K is unknown, the filtering problem is nonlinear. On the other hand, if K is known, but the sink/source term and boundary conditions are unknown, we still have a linear state and parameter estimation problem, as shown in Example 9.5. Nonlinearity usually leads to non-Gaussianity. Although the KF maintains its minimum variance estimator property, propagation of the first two moments is not expected to approximate the highly skewed or multimodal distributions well. Secondly, it is hard to estimate errors in prior statistics. Model biases may be present, and the assumption that $\mathbf{y}_0$ and $\{\eta_k\}$ are mutually uncorrelated may not always hold. As a result, the KF can easily diverge. Thirdly, the KF is not designed for high-dimensional systems.

Nonlinear problems are ubiquitous in EWR applications. Therefore, filters that encompass a wide range of techniques have been developed for nonlinear SDA problems. The main aim of this section is to discuss several variants of the KF for solving general joint state and parameter estimation problems.

9.3.1 Extended Kalman Filter

The extended KF (EKF) is a straightforward extension of the KF concept to nonlinear problems by linearization (Sect. 2.2.4). After the estimate $\hat{\mathbf{y}}_{k-1}$ is obtained from time step $k-1$, the first-order approximation of the evolution operator in (9.1.33) about $\hat{\mathbf{y}}_{k-1}$ is obtained by Taylor's expansion

$$\mathbf{f}(\mathbf{y}_{k-1}) \approx \mathbf{f}(\hat{\mathbf{y}}_{k-1}) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\hat{\mathbf{y}}_{k-1}} (\mathbf{y}_{k-1} - \hat{\mathbf{y}}_{k-1}), \tag{9.3.1}$$

where $[\partial \mathbf{f} / \partial \mathbf{y}]$ is the Jacobian matrix. When (9.1.17) is used as the evolution equation for parameters $\boldsymbol{\theta}$, we have

$$\mathbf{M}_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\hat{\mathbf{y}}_{k-1}} = \left.\begin{bmatrix} \dfrac{\partial \mathbf{f}}{\partial \mathbf{u}} & \dfrac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}\right|_{(\hat{\mathbf{u}}_{k-1}, \hat{\boldsymbol{\theta}}_{k-1})} \tag{9.3.2}$$

The prediction equation in (9.1.33) can then be replaced approximately by the following linear equation:

$$\mathbf{y}_k = \mathbf{f}(\hat{\mathbf{y}}_{k-1}) + \mathbf{M}_k(\mathbf{y}_{k-1} - \hat{\mathbf{y}}_{k-1}) + \boldsymbol{\eta}_k, \tag{9.3.3}$$

where $(\mathbf{y}_{k-1} - \hat{\mathbf{y}}_{k-1})$ is the residual of the $(k-1)$ step. Similarly, linearization of the measurement equation in (9.1.33) about $\mathbf{y}_k^f$ is given by

$$\mathbf{z}_k = \mathbf{g}(\mathbf{y}_k^f) + \mathbf{G}_k(\mathbf{y}_k - \mathbf{y}_k^f) + \boldsymbol{\varepsilon}_k, \tag{9.3.4}$$

where

$$\mathbf{G}_k = \left.\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right|_{\mathbf{y}_k^f} = \left.\begin{bmatrix} \dfrac{\partial \mathbf{g}}{\partial \mathbf{u}} & \dfrac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}} \end{bmatrix}\right|_{(\mathbf{u}_k^f, \boldsymbol{\theta}_k^f)}. \tag{9.3.5}$$

Combination of (9.3.3) and (9.3.5) leads to a linear DA model that can be solved by KF.

- *EKF forecast.*

$$\begin{aligned} \mathbf{y}_k^f &= \mathbf{f}(\hat{\mathbf{y}}_{k-1}) \\ \mathbf{P}_k^f &= \mathbf{M}_k \hat{\mathbf{P}}_{k-1} \mathbf{M}_k^T + \mathbf{Q}_k \end{aligned} \tag{9.3.6}$$

- *EKF assimilation.*

$$\begin{aligned} \hat{\mathbf{y}}_k &= \mathbf{y}_k^f + \mathbf{K}_k[\mathbf{z}_k - \mathbf{g}(\mathbf{y}_k^f)] \\ \hat{\mathbf{P}}_k &= (\mathbf{I} - \mathbf{K}_k \mathbf{G}_k)\mathbf{P}_k^f \\ \mathbf{K}_k &= \mathbf{P}_k^f \mathbf{G}_k^T (\mathbf{G}_k \mathbf{P}_k^f \mathbf{G}_k^T + \mathbf{R}_k)^{-1} \end{aligned} \tag{9.3.7}$$

Note that a known control term can also be added to the EKF forecast equation, as we have done in the KF. Example 9.6 below shows such a case.

The EKF is one of the most widely used nonlinear DA methods for low-dimensional systems. For high-dimensional systems, the computational effort of generating the Jacobian matrix by numerical differentiation will become unaffordable. Obviously, EKF still assumes Gaussianity of posterior PDFs. This is a fundamental limitation of EKF because PDFs of random variables are usually not Gaussian-distributed after nonlinear transformations by the model and measurement operators. Therefore, although EKF can handle mild nonlinearity, it usually gives poor results or even breaks down when (i) the models are highly nonlinear such that the truncated higher-order terms of the Taylor series expansion become significant or (ii) the posterior PDFs are heavily skewed or multimodal. Methods that use higher order moments may alleviate the problem of EKF to some degree; however, they are still based on the concept of approximating a PDF around its mean.

Example 9.6 *Use EKF to estimate hydraulic conductivity*
Consider the 1D groundwater flow system in Example 9.4. But, this time we assume that the hydraulic conductivity, K, is uncertain, which makes the joint state and parameter assimilation problem nonlinear. The PDF of hydraulic conductivity K is typically considered lognormal; thus, we will work with its log-transform, $Y = \ln K$. The parameter evolution equation for Y is written as

$$Y_k = Y_{k-1} + \upsilon_k, \tag{9.3.8}$$

using which the augmented state vector can be defined as $\mathbf{y}_k \triangleq [\mathbf{h}_k^T \quad Y_k]^T$, where υ is white noise. Substituting the log-transformed hydraulic conductivity into (9.1.26), the governing PDE becomes

$$S_s \frac{\partial h}{\partial t} = e^Y \frac{\partial^2 h}{\partial^2 x} + q_s. \tag{9.3.9}$$

In Example 9.4, constant head boundary conditions were applied. For this example, we assume that the right end of the 1D column is subject to a constant flux boundary. The initial and boundary conditions are revised to

$$h(t=0) = h_0,\ h(x=0) = H_0,\ q(x=L) = q_L,$$

where q_L is specified. The resulting state evolution equation still has the same form as that in (9.1.30), which is recapitulated below:

$$\mathbf{h}_k = \mathbf{A}^{-1}(a_1 \mathbf{h}_{k-1} + \mathbf{c}). \tag{9.3.10}$$

However, the quantities have slightly different definitions because of the change in boundary conditions,

$$\mathbf{A} = \begin{pmatrix} a_1 + 2a_2 & -a_2 & & \\ -a_2 & a_1 + 2a_2 & -a_2 & \\ \cdots & \cdots & \cdots & \cdots \\ & & -a_2 & a_1 + a_2 \end{pmatrix},\ \mathbf{c} = \begin{pmatrix} a_2 H_0 + q_{s,1} \\ q_{s,2} \\ \vdots \\ q_{s,L-1} - q_L / \Delta x \end{pmatrix}, \tag{9.3.11}$$

and $a_2 = e^Y / \Delta x^2$. Note that the matrix $\mathbf{A}$ is now dependent on the unknown Y through a_2. To calculate the coefficient a_2 associated with H_0 in the control term $\mathbf{c}$, the Y estimate from the previous step is used. From (9.3.10), we can find (see Appendix C)

$$\frac{\partial \mathbf{f}}{\partial \mathbf{h}} = a_1 \mathbf{A}^{-1}, \frac{\partial \mathbf{f}}{\partial Y} = a_1 \left(\frac{d\mathbf{A}^{-1}}{dY} \right) \mathbf{h} = a_1 \left(\mathbf{A}^{-1} \frac{d\mathbf{A}}{dY} \mathbf{A}^{-1} \right) \mathbf{h}.$$

From (9.3.11), we can find

$$\frac{d\mathbf{A}}{dY} \triangleq \mathbf{A}_d = \begin{pmatrix} 2a_2 & -a_2 & & \\ -a_2 & 2a_2 & -a_2 & \\ \cdots & \cdots & \cdots & \cdots \\ & & -a_2 & a_2 \end{pmatrix}$$

As a result, the matrix $\mathbf{M}_k$ for this example is

$$\mathbf{M}_k = \begin{pmatrix} a_1\mathbf{A}^{-1} & a_1\mathbf{A}^{-1}\mathbf{A}_d\mathbf{A}^{-1}\mathbf{h} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}, \tag{9.3.12}$$

where all quantities in $\mathbf{M}_k$ are evaluated using $\hat{\mathbf{h}}_{k-1}$ and $\hat{Y}_{k-1}$.

For demonstration, the following data are assumed:

$$H_0 = 10\text{ [L]}, S_s = 1\times10^{-5}\text{ [L}^{-1}\text{]}, q_L = 1\times10^{-4}\text{ [L/T]},$$

$$\sigma_\eta = 1\times10^{-2}, \sigma_\upsilon = 0.3, \sigma_R = 0.1, \mathbf{P}_{h,0} = 1\times10^{-6}\mathbf{I}, P_{Y,0} = 0.1$$

$$K = 0.0183\text{ [L/T] or } Y = -4 \text{ (true value)},$$

$$K = 0.0122\text{ [L/T] or } Y = -4.4 \text{ (initial value)}.$$

where **I** is the identity matrix and the distributed sink/source q_s is assumed zero. Figure 9.3a shows the computed head distributions for different assimilation times. As before, measurements are taken at 10 [L] intervals along x for every 0.05 [T] and the total assimilation time is 1.0 [T]. Figure 9.3b shows the assimilated K values from different time steps. Starting with an initial guess of $Y = -4.4$ (K=0.0122 [L/T]), the largest parameter update happened after the first assimilation step, after which the curve gradually approaches to the "true" value.

The accuracy and stability of the EKF algorithm depend critically on a number of factors, including total assimilation time and assimilation frequency, specification of error covariance, and magnitude of boundary flux, all of which may need to be adjusted simultaneously. A closely related question is when to stop assimilation, which has found surprisingly little discussion in the literature. One possibility is to monitor the variance of parameter being assimilated, as shown in Fig. 9.4 for the current example. The plot indicates that P_Y decreases gradually with time and essentially becomes flat toward the end of assimilation, which signifies that the information content diminishes as the head distribution approaches steady state.

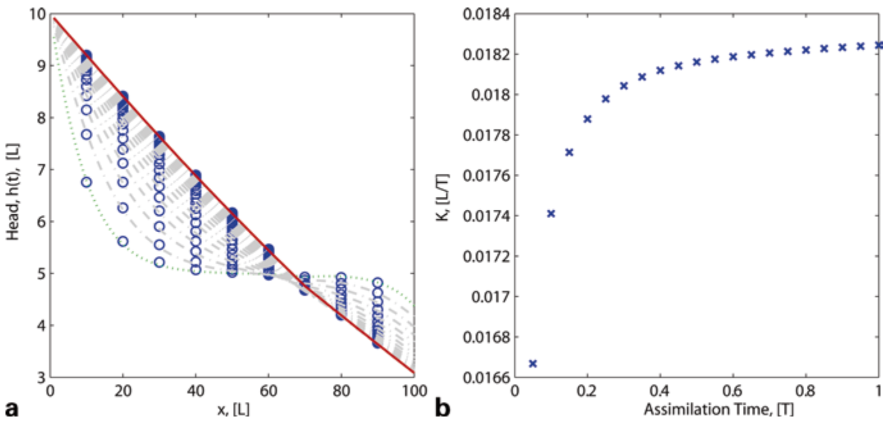


Fig. 9.3 **a** Evolution of head distribution $\mathbf{h}_k$ with time, where the *green dotted line* represents head distribution after first step, *brown solid color line* represents the final head distribution, and *gray* color dashed lines represent the intermediate steps (*open circles* indicate sample locations) and **b** value of hydraulic conductivity, K, as a function of assimilation time

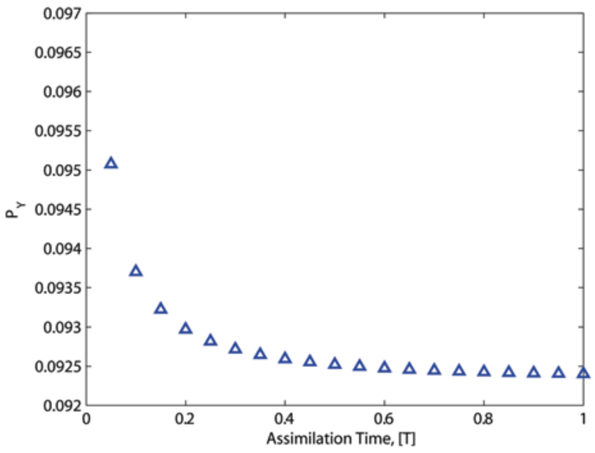


Fig. 9.4 Reduction of the parameter variance P_Y during assimilation

As a result, further updates will provide little additional value or, worse, can even deteriorate existing estimates. Thus, instead of the constant pumping, in practice, it is more meaningful to vary the stimulation to the system so that more useful information can be obtained for assimilation. Such topic has been investigated in hydraulic tomography using numerical, laboratory, and field experiments (e.g., Liu et al. 2007; Li et al. 2005; Leube et al. 2012).

For this example, the EKF was performed using two EKF functions, `ekf_predict1` and `ekf_update1`, from the MATLAB toolbox EKFUKF (Hartikainen and Särkkä 2008).

9.3.2 Unscented Kalman Filter

The EKF linearizes the nonlinear prediction and measurement operators by calculating the Jacobian in each step. A prerequisite is that the models be differentiable. Moreover, the performance of EKF can be poor because the posterior mean and covariance are corrupted by the truncation error. The *unscented Kalman filter* (UKF), introduced in mid-1990s (Julier et al. 2000; Julier et al. 1995; Julier and Uhlmann 2004), is a derivative-free alternative of EKF. It is built upon the idea that approximating a PDF is usually easier than linearizing a nonlinear function. In UKF, the state distribution is obtained by a sampling technique. Because it relinquishes the need to linearize system operators, the UKF is expected to have better accuracy than the EKF.

9.3.2.1 Unscented Transform

The core of the UKF is *unscented transform*, which approximates the PDF of an n-dimensional random variable $\mathbf{x} \in \mathbb{R}^n$ via a set of $2n+1$ sample points of $\mathbf{x}$, $\{\mathcal{X}_i\}$, also known as the *sigma points*. The sigma points can be calculated based on prior statistics

$$\mathcal{X}_0 = \bar{\mathbf{x}},\ w_0 = \frac{\kappa}{n+\kappa}, \tag{9.3.13}$$

$$\mathcal{X}_i = \bar{\mathbf{x}} + \left(\sqrt{(n+\kappa)\mathbf{P}}\right)_i,\ w_i = \frac{1}{2(n+\kappa)}, i = 1,\cdots,n, \tag{9.3.14}$$

$$\mathcal{X}_i = \bar{\mathbf{x}} - \left(\sqrt{(n+\kappa)\mathbf{P}}\right)_i,\ w_i = \frac{1}{2(n+\kappa)},\ i = n+1,\cdots,2n, \tag{9.3.15}$$

where $\bar{\mathbf{x}}$ and $\mathbf{P}$ are the mean and covariance of $\mathbf{x}$, respectively; w_i is the weight associated with the ith sigma point $\mathcal{X}_i$ and the sum of all weights is equal to 1; κ is a scaling parameter; and $\left(\sqrt{(n+\kappa)\mathbf{P}}\right)_i$ is the ith column of the matrix square root of $(n+\kappa)\mathbf{P}$. It can be shown that the $2n+1$ sigma points generated through (9.3.13) to (9.3.15) completely capture the mean and covariance of an n-dimensional Gaussian random variable. When propagated through a nonlinear system, the sigma points also capture the posterior mean and covariance accurately up to the second order for any nonlinearity, with errors introduced only in the third and higher orders (Julier and Uhlmann 2004). The scaling parameter κ is introduced to control the "distance" of sigma points from the mean to better handle nonlinearity. When $\kappa > 0$, the sigma points are shifted away from the mean; conversely, when $\kappa < 0$, the points are moved closer to the mean. In particular, when $\kappa = 3 - n$, the dimensional scaling invariance is achieved. However, when $\kappa = 3 - n < 0$, the scaling could make the resulting covariance non-positive definite. A scaled unscented transform has been introduced to address this issue (Julier and Uhlmann 2004), which results in slightly different weights for mean and covariance approximation.

As a simple example, consider a nonlinear function of two variables

$$s = uv,$$

where *u* and *v* are standard $\mathcal{N}(0,1)$ random variables and are independent of each other. The prior statistics of the bivariate random variable $\mathbf{x} = [u, v]^T$ are

$$\bar{\mathbf{x}} = [0,0]^T \text{ and } \mathbf{P} = \mathbf{I}_{2\times2}$$

and the set of sigma points and associated weights are

$$\mathcal{X}_0 = [0,0]^T, \mathcal{X}_1 = [\sqrt{3},0]^T, \mathcal{X}_2 = [0,\sqrt{3}]^T, \mathcal{X}_3 = [-\sqrt{3},0]^T, \mathcal{X}_4 = [0,-\sqrt{3}]^T,$$

$$w_0 = \frac{1}{3},\ w_i = \frac{1}{6}\ (i = 1,2,3,4),$$

in which the scaling parameter κ is equal to 1. Note: (i) The PDF of *s* also has mean 0 and variance 1, but is non-Gaussian; it can be shown that its PDF is proportional to a zeroth-order-modified Bessel function of second kind (Weisstein 2013) and (ii) as we shall see in Chap. 10, the selected sigma points coincide with zeros of the third-order Hermite polynomials, which are known to provide optimal polynomial bases for Gaussian random variables. The concept of representing a multidimensional parameter space with a parsimonious set of points is also behind many experimental design and reduced-order modeling methods to be described in Chap. 10.

9.3.2.2 UKF Implementation

We now describe the UKF implementation for the joint state and parameter estimation problem (9.1.33), namely

$$\begin{aligned}\mathbf{y}_k &= \mathbf{f}(\mathbf{y}_{k-1}) + \eta_k \\ \mathbf{z}_k &= \mathbf{g}(\mathbf{y}_k) + \varepsilon_k.\end{aligned}$$

According to the general DA procedure, UKF starts from the known initial $(\hat{\mathbf{y}}_0, \hat{\mathbf{P}}_0)$. At any time t_k, UKF uses the following forecast and update steps to find $(\hat{\mathbf{y}}_k, \hat{\mathbf{P}}_k)$ from $(\hat{\mathbf{y}}_{k-1}, \hat{\mathbf{P}}_{k-1})$,

- *UKF Forecast.* Generate $2L+1$ sigma points according to (9.3.13) to (9.3.15), by replacing $\mathcal{X}$, $\bar{\mathbf{x}}$ and $\mathbf{P}$ in those equations with $\mathcal{Y}$, $\hat{\mathbf{y}}_{k-1}$ and $\hat{\mathbf{P}}_{k-1}$, respectively,

$$\begin{aligned}\mathcal{Y}_{k-1,0} &= \hat{\mathbf{y}}_{k-1},\ w_0 = \frac{\kappa}{L+\kappa} \\ \mathcal{Y}_{k-1,i} &= \hat{\mathbf{y}}_{k-1} + \left(\sqrt{(L+\kappa)\hat{\mathbf{P}}_{k-1}}\right)_i,\ w_i = \frac{1}{2(L+\kappa)},\ i = 1,\cdots,L \\ \mathcal{Y}_{k-1,i} &= \hat{\mathbf{y}}_{k-1} - \left(\sqrt{(L+\kappa)\hat{\mathbf{P}}_{k-1}}\right)_i, w_i = \frac{1}{2(L+\kappa)},\ i = L+1,\cdots,2L\end{aligned} \tag{9.3.16}$$

where L is the dimension of the augmented state variable $\mathbf{y}$, the first subscript of each sigma point $\mathcal{Y}$ is the time step index, and the second subscript is the point index. The nonlinear forward operator $\mathbf{f}(\cdot)$ is then used to evolve these sigma points into a set of prediction sigma points

$$\mathcal{Y}_{k,i}^f = \mathbf{f}(\mathcal{Y}_{k-1,i}),\ i = 0,1,\cdots,2L. \tag{9.3.17}$$

The forecast mean and covariance can be approximated using weighted sum of sigma points

$$\mathbf{y}_k^f = \sum_{i=0}^{2L} w_i \mathcal{Y}_{k,i}^f, \tag{9.3.18}$$

$$\mathbf{P}_k^f = \sum_{i=0}^{2L} w_i (\mathcal{Y}_{k,i}^f - \mathbf{y}_k^f)(\mathcal{Y}_{k,i}^f - \mathbf{y}_k^f)^T + \mathbf{Q}_k, \tag{9.3.19}$$

where model and parameter noise is added just like in the case of KF. All weights are given in (9.3.16), but slightly different weights may be used for mean and covariance (Wan and Van Der Merwe 2000).

- *UKF assimilation*. First, using $\mathbf{y}_k^f$ and $\mathbf{P}_k^f$ to replace $\hat{\mathbf{y}}_{k-1}$ and $\hat{\mathbf{P}}_{k-1}$ in (9.3.16) to update the prediction sigma points

$$\begin{aligned}
\mathcal{Y}_{k,0}^f &= \mathbf{y}_k^f, \\
\mathcal{Y}_{k,i}^f &= \mathbf{y}_k^f + (\sqrt{(L+\kappa)\mathbf{P}_k^f})_i, i = 1,\cdots,L \\
\mathcal{Y}_{k,i}^f &= \mathbf{y}_k^f - (\sqrt{(L+\kappa)\mathbf{P}_k^f})_i, i = L+1,\cdots,2L
\end{aligned}$$

The nonlinear measurement operator $\mathbf{g}(\cdot)$ is then used to transfer these prediction sigma points into a set of measurement sigma points

$$\mathcal{Z}_{k,i}^f = \mathbf{g}(\mathcal{Y}_{k,i}^f),\ i = 0,1,\cdots,2L. \tag{9.3.20}$$

The weighted sample mean and sample covariance of these measurement sigma points are

$$\mathbf{z}_k^f = \sum_{i=0}^{2L} w_i \mathcal{Z}_{k,i}^f, \tag{9.3.21}$$

$$\mathbf{P}_{\mathbf{z}_k\mathbf{z}_k}^f = \sum_{i=0}^{2L} w_i (\mathcal{Z}_{k,i}^f - \mathbf{z}_k^f)(\mathcal{Z}_{k,i}^f - \mathbf{z}_k^f)^T + \mathbf{R}_k. \tag{9.3.22}$$

We can also calculate the cross-covariance between $\left\{\mathcal{Z}_{k,i}^f\right\}$ and $\left\{\mathcal{Y}_{k,i}^f\right\}$ as

$$\mathbf{P}^f_{\mathbf{y}_k\mathbf{z}_k} = \sum_{i=0}^{2L} w_i(\mathcal{Y}^f_{k,i} - \mathbf{y}^f_k)(\mathcal{Z}^f_{k,i} - \mathbf{z}^f_k)^T. \tag{9.3.23}$$

After obtaining $\mathbf{P}_{\mathbf{z}_k\mathbf{z}_k}$ and $\mathbf{P}_{\mathbf{y}_k\mathbf{z}_k}$, according to (9.2.7), we have Kalman gain

$$\mathbf{K}_k = \mathbf{P}^f_{\mathbf{y}_k\mathbf{z}_k}(\mathbf{P}^f_{\mathbf{z}_k\mathbf{z}_k})^{-1}. \tag{9.3.24}$$

The required pair $(\hat{\mathbf{y}}_k, \hat{\mathbf{P}}_k)$ at t_k is then given by the Kalman update formula

$$\begin{aligned}\hat{\mathbf{y}}_k &= \mathbf{y}^f_k + \mathbf{K}_k(\mathbf{z}_k - \mathbf{z}^f_k)\\ \hat{\mathbf{P}}_k &= \mathbf{P}^f_k - \mathbf{K}_k\mathbf{P}_{\mathbf{z}_k\mathbf{z}_k}\mathbf{K}^T_k.\end{aligned} \tag{9.3.25}$$

We remark that the UKF can be considered as a special case of the particle filter or the ensemble-based Kalman filter to be discussed shortly. In the UKF, the size of the ensemble is fixed and is at least twice the dimension of the augmented state. Therefore, the computational cost of calculating the covariance matrix and its square root for generating sigma points is prohibitively high when the system dimension is large. We may use dimension reduction techniques described in Chap. 6 (e.g., PCA) to first project the state vector to a low-dimensional space, perform the UKF in the lower-dimensional space, and then reproject the result back to the original state space. Variants of the UKF, including an implementation of the dual sigma-point, state-parameter filter, were discussed in the PhD dissertation of van de Merwe (2004). In particular, van de Merwe proposed a sigma-point Kalman filter framework to combine several "derivativeness" Kalman filters into a single algorithmic framework and verified that UKF achieved consistently better performance than the EKF.

The UKF approximates the low-order moments of a prior PDF; therefore, like the EKF, it works best for Gaussian posterior PDF and will fail when the distribution is highly skewed or multimodal.

Example 9.7 *Estimate hydraulic conductivity using UKF*
We solve the SDA problem in Example 9.6 using the UKF. In this case, the dimension of augmented state vector is $L = 101$, which means the total number of sigma points required is 203. Figure 9.5 shows the evolution of assimilated *K* (recall that the log-transformed *K* is used in the actual assimilation). For comparison, the results of EKF from Example 9.6 are also plotted. It can be seen that the convergence of UKF is significantly faster than that of the EKF, especially for the first update. The UKF also outperforms the EKF in terms of accuracy. At the end of assimilation, the exact value of *K* is recovered.

For this example, we used two functions `ukf_predict1` and `ukf_update1`, from the MATLAB toolbox EKFUKF to perform the UKF. All hyperparameters (i.e., error variances) of the UKF are set the same as those used in Example 9.6.

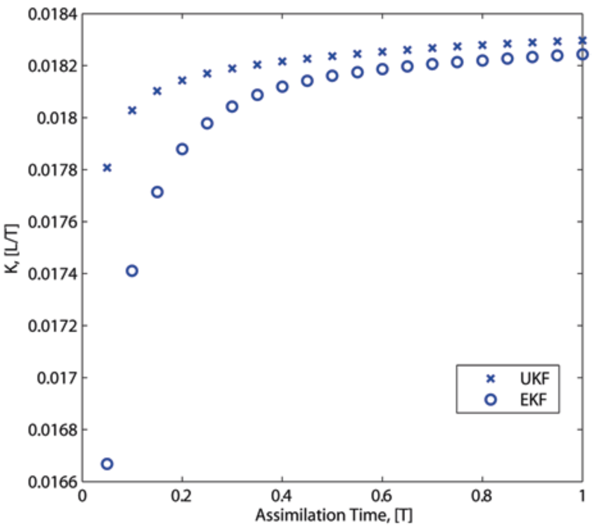


Fig. 9.5 Comparison of hydraulic conductivity K assimilated by using UKF (*crosses*) and EKF (*open circles*)

9.3.3 Gaussian Sum Filter

The *Gaussian sum filter* (GSF), originally introduced in 1970s (Sorenson and Alspach 1971), is a technique for approximating an arbitrary PDF. Let $\mathcal{N}(\boldsymbol{\mu},\mathbf{C})$ denote a multivariate Gaussian distribution for $\mathbf{x} \in \mathbb{R}^n$,

$$\mathcal{N}(\boldsymbol{\mu},\mathbf{C}) = (2\pi)^{-n/2}\left|\mathbf{C}\right|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{C}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right). \tag{9.3.26}$$

It can be shown that any PDF $p(\mathbf{x})$ can be approximated as closely as desired by a weighted sum of Gaussian distributions (Anderson and Moore 1979)

$$p(\mathbf{x}) = \sum_{i=1}^{N} w_i \mathcal{N}(\boldsymbol{\mu}_i, \mathbf{C}_i), \tag{9.3.27}$$

where the weights $\{w_i\}$ are nonnegative and the sum of all weights is equal to 1. Eq. (9.3.27) is known as the *Gaussian mixture model* (GMM), for which the mean and covariance of the mixture can be calculated as

$$\mathbf{m} = \sum_{i=1}^{N} w_i \boldsymbol{\mu}_i \tag{9.3.28}$$

$$\mathbf{P}_{\mathbf{x}} = \sum_{i=1}^{N} w_i (\mathbf{C}_i + (\mathbf{x}-\boldsymbol{\mu}_i)(\mathbf{x}-\boldsymbol{\mu}_i)^T) \tag{9.3.29}$$

The parameters of the GMM are $\{\boldsymbol{\mu}_i\}$, $\{\mathbf{C}_i\}$, and $\{w_i\}$ ($i = 1, \cdots, N$), where N is the number of mixture components.

We now describe the formulation of the GSF for the SDA model (9.1.33) and for the case when both model and measurement error distributions are Gaussian. The starting point of the GSF is approximation of the prior PDF of the augmented state $\mathbf{y}$ at the *k*th step using a GMM

$$p(\hat{\mathbf{y}}_{k-1} | \mathbf{z}_{1:k-1}) \approx \sum_{i=1}^{N} w_{k,i} \mathcal{N}(\hat{\mathbf{y}}_{k-1}; \boldsymbol{\mu}_{k-1,i}, \mathbf{C}_{k-1,i}), \tag{9.3.30}$$

where symbols $\boldsymbol{\mu}$ and $\mathbf{C}$ represent the mean and covariance of augmented state variables for each mixture component. Like other filters that have been introduced so far, the GSF also consists of recursive forecast and measurement update steps. During the forecast step, each component of the GMM is propagated using the model operator. This can be accomplished using an array of either EKF or UKF, one for each component. For the *i*th component, if the EKF is used, the system model is first linearized about the prior mean $\boldsymbol{\mu}_{k-1,i}$, which is then propagated to $\boldsymbol{\mu}_{k,i}^f$ using the model operator, and the forecast state covariance can be calculated using $\mathbf{C}_{k-1,i}$ and error covariance $\mathbf{Q}_k$; if the UKF is used, a set of sigma points is generated using $\boldsymbol{\mu}_{k-1,i}$ and $\mathbf{C}_{k-1,i}$ of the *i*th component and then propagated using the model operator for calculating the forecast statistics. The results are then combined to construct a forecast GMM

$$p(\mathbf{y}_k^f | \mathbf{z}_{1:k-1}) = \sum_{i=1}^{N} w_{k,i} \mathcal{N}(\mathbf{y}_k^f; \boldsymbol{\mu}_{k,i}^f, \mathbf{C}_{k,i}^f). \tag{9.3.31}$$

In the measurement update step, the forecast mean and covariance of each component are updated to $\boldsymbol{\mu}_{k,i}$ and $\mathbf{C}_{k,i}$, using the measurement update scheme of either the EKF or UKF. The weight of each component is updated by the likelihood function, $p_i(\mathbf{z}_k | \mathbf{y}_k = \boldsymbol{\mu}_{k,i})$, which is also Gaussian

$$w_{k+1,i} = \frac{w_{k,i} p_i(\mathbf{z}_k | \mathbf{y}_k)}{\sum_{j=1}^{N} w_{k,j} p_j(\mathbf{z}_k | \mathbf{y}_k)}, \tag{9.3.32}$$

where the denominator normalizes each mixture component's weight.

We will see in Sect. 9.5 that GSF is a special case of particle filters. The GSF formulation for non-Gaussian model and measurement errors is more involved. At each DA step, two additional GMMs are needed to represent the non-Gaussian model and measurement error distributions, respectively. Let us assume that the number of components used in each of these GMMs is, I and J, respectively. Beginning with N in the original mixture model, the total number of mixture components becomes NI after the first forecast step and NIJ after the first measurement update. As a result, the total number of components will grow exponentially during the DA process. For practical purpose, after measurement update, we want to retain only N components that have the highest weights and eliminate the rest. This is essentially importance sampling, which we will describe in Sect. 9.5.

So far, we have assumed that N for the GMM is known and the parameters for initializing a GMM are given. Methods for learning GMM (i.e., identifying its parameters) have been discussed extensively in the data clustering and pattern recognition literature (Figueiredo and Jain 2002; Fraley and Raftery 1998). Starting with some initial guess of N and a set of samples generated from the prior of GMM, an unsupervised GMM learning routine determines the optimal GMM parameters by minimizing a cost function. The process of selecting the number of mixture components is thus similar to model complexity control. We can apply model selection criteria such as AIC, BIC, and Kullback–Leibler distance to determine N (Bouveyron et al. 2007). For the nonlinear non-Gaussian case, this GMM learning step needs to be repeated after each measurement update to determine GMM parameters. Such reinitialization can be very expensive for online analysis. Another problem commonly encountered in practice is that the component covariance $\mathbf{C}$ is often ill-conditioned, causing numerical instability during the GMM learning process.

Despite its computational complexity, the GSF remains one of the most powerful schemes for nonlinear filtering. The flexibility of the filter lies in that (i) the design of the GSF readily lends itself to parallel processing and (ii) the core algorithm for evolving each mixture component can be tuned for different applications. In addition to the EKF and UKF, the GSF has been combined with the importance sampling particle filters (Kotecha and Djuric 2003; van de Merwe 2004; Rings et al. 2012) and the reduced-rank ensemble KFs to be introduced in the next section (Kim et al. 2003; Smith 2007; Sun et al. 2009b).

9.4 Reduced-Rank Filters

9.4.1 The Classical Ensemble Kalman Filter

9.4.1.1 Derivation

The ensemble KF or EnKF is a sequential Monte Carlo approach originally introduced by Evensen (1994) for DA applications in oceanography. The EnKF has drawn wide attention from researchers in different fields in the last decade. Unlike the classical KF, the EnKF is a reduced-rank filter designed to handle high-dimensional state space and mild nonlinearity, and unlike variational methods, the EnKF requires much less effort to implement and can be readily applied to many different problems. The latter advantage partly explains the popularity of the EnKF for joint state and parameter estimation, especially in the context of Case II described in Sect. 9.1.2. Toward that end, the EnKF is an efficient SDA algorithm for estimating distributed parameters, providing an alternative to the traditional batch parameter estimation algorithm. Recall that a full-Bayesian algorithm, the GPR model, was introduced in Chap. 8 for solving low-dimensional data-driven models. GPR requires inversion of a full-rank covariance, which is computationally prohibitive for high-dimensional systems. The EnKF can be considered a reduced-rank counterpart of the GPR for data assimilation.

The basic implementation of the classical EnKF is almost identical to that of the UKF, except that deterministic sigma points are now replaced by user-provided random realizations to generate ensemble statistics. During initialization of EnKF, the prior distribution of the unknown parameters $\boldsymbol{\theta}$ is sampled to generate an N-member ensemble of initial parameter estimates $\left\{\boldsymbol{\theta}_{0,1}, \boldsymbol{\theta}_{0,2}, \cdots, \boldsymbol{\theta}_{0,N}\right\}$, where the second subscript of $\boldsymbol{\theta}_0$ is the ensemble member index. An N-member ensemble of initial states $\left\{\mathbf{u}_{0,i} = \mathbf{u}(\boldsymbol{\theta}_{0,i}) \mid i = 1, 2, \cdots, N\right\}$ is then obtained by running the forward model N times, after which we obtain an N-member ensemble of initial augmented state vectors: $\left\{\hat{\mathbf{y}}_{0,1}, \hat{\mathbf{y}}_{0,2}, \cdots, \hat{\mathbf{y}}_{0,N}\right\}$, where $\hat{\mathbf{y}}_{0,i} = \left[\mathbf{u}_{0,i}^T \quad \boldsymbol{\theta}_{0,i}^T\right]^T$ (Fig. 9.6). The main underlying assumption of the EnKF is that the number of ensemble members required for approximating state/parameter statistics is generally much smaller than that required by a full-Bayesian treatment. For distributed parameter models, this means using an ensemble of ~10^2 members to approximate an augmented state space on the order of 10^{12-16}. Therefore, before using the EnKF, main questions to be addressed are (i) whether system dynamics can be effectively captured by a small number of ensemble samples, (ii) how to generate these samples or, equivalently, how to parameterize the unknowns, and (iii) whether the generated parameter fields and error distributions are suitable to be updated in a KF-like procedure. Both the quantity and quality of ensembles will have an important impact on the performance and stability of the EnKF. For joint state and parameter estimation, careful consideration must be given to parameterization and possible transformation of the uncertain distributed parameter field(s), keeping in mind that the EnKF

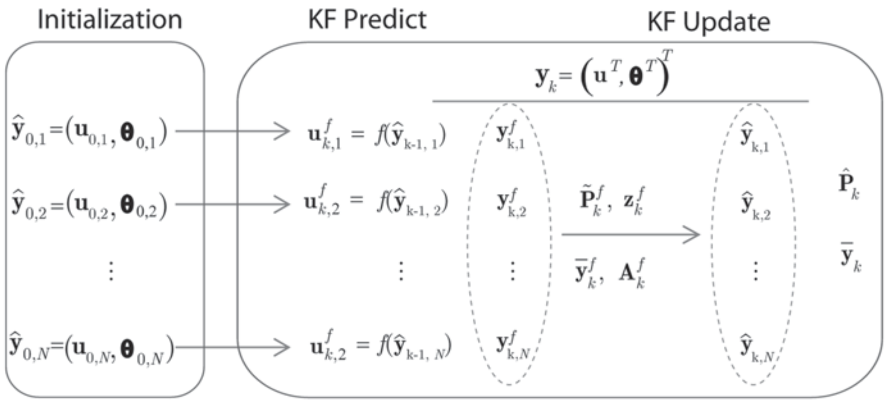

Initialization
KF Predict
KF Update
$\mathbf{y}_k = (\mathbf{u}^T, \boldsymbol{\theta}^T)^T$
$\hat{\mathbf{y}}_{0,1} = (\mathbf{u}_{0,1}, \boldsymbol{\theta}_{0,1})$
$\hat{\mathbf{y}}_{0,2} = (\mathbf{u}_{0,2}, \boldsymbol{\theta}_{0,2})$
$\hat{\mathbf{y}}_{0,N} = (\mathbf{u}_{0,N}, \boldsymbol{\theta}_{0,N})$
$\mathbf{u}^f_{k,1} = f(\hat{\mathbf{y}}_{k-1,1})$
$\mathbf{u}^f_{k,2} = f(\hat{\mathbf{y}}_{k-1,2})$
$\mathbf{u}^f_{k,2} = f(\hat{\mathbf{y}}_{k-1,N})$
$\mathbf{y}^f_{k,1}$
$\mathbf{y}^f_{k,2}$
$\mathbf{y}^f_{k,N}$
$\tilde{\mathbf{P}}^f_k$, $\mathbf{z}^f_k$
$\bar{\mathbf{y}}^f_k$, $\mathbf{A}^f_k$
$\hat{\mathbf{y}}_{k,1}$
$\hat{\mathbf{y}}_{k,2}$
$\hat{\mathbf{y}}_{k,N}$
$\hat{\mathbf{P}}_k$
$\bar{\mathbf{y}}_k$

Fig. 9.6 Main stages of the EnKF for joint state and parameter estimation, in which prediction and analysis steps are applied recursively

still assumes Gaussian statistics. This is the place where parameterization schemes introduced in Chap. 6 can be applied to generating random realizations of distributed parameters.

During each DA step of the EnKF, the current distribution of the augmented state ensemble is updated by using data and propagated to the next time step. From the posterior distribution, we can find the best point estimates (i.e., ensemble mean) and assess the uncertainty of estimation (ensemble variance). In general, after obtaining

$$\hat{\mathbf{y}}_{k-1,1}, \hat{\mathbf{y}}_{k-1,2}, \cdots, \hat{\mathbf{y}}_{k-1,N}, \tag{9.4.1}$$

the EnKF uses the following steps to find $\left\{\hat{\mathbf{y}}_{k,1}, \hat{\mathbf{y}}_{k,2}, \cdots, \hat{\mathbf{y}}_{k,N}\right\}$ (see also Fig. 9.6):

- *EnKF forecast*. Use the forward model in (9.1.33) to propagate each ensemble member of (9.4.1) to obtain a forecast ensemble

$$\mathbf{y}_{k,i}^f = \mathbf{f}(\hat{\mathbf{y}}_{k-1,i}), i = 1, 2, \cdots, N \tag{9.4.2}$$

The mean and covariance of the forecast ensemble (9.4.2) are given, respectively, by

$$\overline{\mathbf{y}}_k^f = \frac{1}{N}\sum_{i=1}^{N} \mathbf{y}_{k,i}^f, \tag{9.4.3}$$

$$\tilde{\mathbf{P}}_k^f = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{y}_{k,i}^f - \overline{\mathbf{y}}_k^f)(\mathbf{y}_{k,i}^f - \overline{\mathbf{y}}_k^f)^T = \frac{\mathbf{A}_k^f(\mathbf{A}_k^f)^T}{N-1}. \tag{9.4.4}$$

The EnFK uses sample mean $\overline{\mathbf{y}}_k^f$ to approximate $\mathbf{y}_k^f$ and sample covariance $\tilde{\mathbf{P}}_k^f$ to approximate $\mathbf{P}_k^f$. Classical statistics tells us that (9.4.3) and (9.4.4) are unbiased estimators of the mean and covariance and converge to those statistics when $N \to \infty$. In (9.4.4), $\mathbf{A}_k^f$ is an ensemble perturbation matrix (or anomaly matrix) obtained by subtracting forecast ensemble mean from each ensemble member (i.e., the *i*th column of $\mathbf{A}_k^f$ is given by $\mathbf{y}_{k,i}^f - \overline{\mathbf{y}}_k^f$).

- *EnKF assimilation*. Use the measurement equation in (9.1.33) to obtain model outputs at observation locations for each ensemble member in (9.4.2), namely

$$\mathbf{z}_{k,i}^f = \mathbf{g}(\mathbf{y}_{k,i}^f), \ i = 1, 2, \cdots, N. \tag{9.4.5}$$

Let $\overline{\mathbf{z}}_k^f$ be the ensemble mean of these $\mathbf{z}_{k,i}^f$'s generated by the model, and we have the following ensemble covariance and cross-covariance matrices:

$$\tilde{\mathbf{P}}_{\mathbf{z}_k\mathbf{z}_k}^f = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{z}_{k,i}^f - \overline{\mathbf{z}}_k^f)(\mathbf{z}_{k,i}^f - \overline{\mathbf{z}}_k^f)^T + \mathbf{R}_k \tag{9.4.6}$$

$$\tilde{\mathbf{P}}^f_{\mathbf{y}_k\mathbf{z}_k} = \frac{1}{N-1}\sum_{i=1}^{N}\left(\mathbf{y}^f_{k,i} - \overline{\mathbf{y}}^f_k\right)\left(\mathbf{z}^f_{k,i} - \overline{\mathbf{z}}^f_k\right)^T \tag{9.4.7}$$

Using $\tilde{\mathbf{P}}^f_{\mathbf{z}_k\mathbf{z}_k}$ and $\tilde{\mathbf{P}}^f_{\mathbf{y}_k\mathbf{z}_k}$ to replace $\mathbf{P}^f_{\mathbf{z}_k\mathbf{z}_k}$, and $\mathbf{P}^f_{\mathbf{y}_k\mathbf{z}_k}$ in (9.2.7), we obtain an approximation of the Kalman gain

$$\tilde{\mathbf{K}}_k = \tilde{\mathbf{P}}^f_{\mathbf{y}_k\mathbf{z}_k}(\tilde{\mathbf{P}}^f_{\mathbf{z}_k\mathbf{z}_k})^{-1}. \tag{9.4.8}$$

Finally, the assimilated members of the augmented state ensemble are obtained using the measurement update formula

$$\hat{\mathbf{y}}_{k,i} = \mathbf{y}^f_{k,i} + \tilde{\mathbf{K}}_k(\mathbf{z}_k + \boldsymbol{\varepsilon}_{k,i} - \mathbf{z}^f_{k,i}), i = 1, 2, \cdots, N, \tag{9.4.9}$$

where $\boldsymbol{\varepsilon}_{k,i}$ is a normally distributed random variable with zero mean and covariance $\mathbf{R}_k$. The updated ensemble (9.4.9) contains the required results of a DA time step

$$\overline{\mathbf{y}}_k \approx \frac{1}{N}\sum_{i=1}^{N}\hat{\mathbf{y}}_{k,i}, \tag{9.4.10}$$

$$\hat{\mathbf{P}}_k \approx \frac{1}{N-1}\sum_{i=1}^{N}(\hat{\mathbf{y}}_{k,i} - \overline{\mathbf{y}}_k)(\hat{\mathbf{y}}_{k,i} - \overline{\mathbf{y}}_k)^T. \tag{9.4.11}$$

In fact, $\boldsymbol{\varepsilon}_{k,i}$ in (9.4.9) is an artificial perturbation added the observation data. Without this term, the ensemble covariance $\hat{\mathbf{P}}_k$ would be underestimated, as we shall show below.

Now, we can replace (9.4.1) by (9.4.9) and move on to the next time step. Although the EnKF algorithm presented herein is easy to interpret, in practice, the following matrix form is used for computational efficiency.

9.4.1.2 The Matrix Form of EnKF

Most implementations of EnKF are based on its matrix form. Let us define the following matrices:

$$\begin{aligned}
\hat{\mathbf{Y}}_k &= \begin{bmatrix}\hat{\mathbf{y}}_{k,1} & \hat{\mathbf{y}}_{k,2} & \cdots & \hat{\mathbf{y}}_{k,N}\end{bmatrix}\\
\mathbf{Y}^f_k &= \begin{bmatrix}\mathbf{y}^f_{k,1} & \mathbf{y}^f_{k,2} & \cdots & \mathbf{y}^f_{k,N}\end{bmatrix}\\
\mathbf{A}^f_k &= \begin{bmatrix}\mathbf{y}^f_{k,1} - \overline{\mathbf{y}}^f_k & \mathbf{y}^f_{k,2} - \overline{\mathbf{y}}^f_k & \cdots & \mathbf{y}^f_{k,N} - \overline{\mathbf{y}}^f_k\end{bmatrix}\\
\mathbf{Z}_k &= \begin{bmatrix}\mathbf{z}_{k,1} & \mathbf{z}_{k,2} & \cdots & \mathbf{z}_{k,N}\end{bmatrix}, \quad \mathbf{z}_{k,i} = \mathbf{z}_k + \boldsymbol{\varepsilon}_{k,i}\\
\mathbf{D}_k &= \begin{bmatrix}\mathbf{z}_{k,1} - \mathbf{z}_k & \mathbf{z}_{k,2} - \mathbf{z}_k & \cdots & \mathbf{z}_{k,N} - \mathbf{z}_k\end{bmatrix}
\end{aligned}$$

Because we have

$$\tilde{\mathbf{P}}_k^f = \frac{\mathbf{A}_k^f(\mathbf{A}_k^f)^T}{N-1}, \tag{9.4.12}$$

the Kalman gain in (9.2.6) can be calculated approximately by

$$\begin{aligned}\tilde{\mathbf{K}}_k &= \tilde{\mathbf{P}}_k^f\mathbf{G}_k^{\ T}[\mathbf{G}_k\tilde{\mathbf{P}}_k^f\mathbf{G}_k^{\ T} + \tilde{\mathbf{R}}_k]^{-1} \\ &= \mathbf{A}_k^f(\mathbf{G}_k\mathbf{A}_k^f)^T[(\mathbf{G}_k\mathbf{A}_k^f)(\mathbf{G}_k\mathbf{A}_k^f)^T + \tilde{\mathbf{R}}_k]^{-1}.\end{aligned} \tag{9.4.13}$$

Here, we assume that the measurement equation is linear, and the measurement error covariance $\mathbf{R}_k$ is replaced approximately by its ensemble covariance

$$\tilde{\mathbf{R}}_k = \frac{\mathbf{D}_k\mathbf{D}_k^T}{N-1}. \tag{9.4.14}$$

The second equality in (9.4.13) suggests that instead of calculating and storing the forecast covariance matrix, we only need to deal with matrix $\mathbf{G}\mathbf{A}_k^f$ when calculating the approximate Kalman gain. The assimilated ensemble (9.4.9) can be represented by

$$\hat{\mathbf{Y}}_k = \mathbf{Y}_k^f + \mathbf{K}_k(\mathbf{Z}_k - \mathbf{G}_k\mathbf{Y}_k^f), \tag{9.4.15}$$

from which the updated ensemble mean in (9.4.10) is calculated to serve as an unbiased estimate of the augmented state and the updated covariance of estimation in (9.4.11) can be represented by

$$\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{G}_k)\tilde{\mathbf{P}}_k^f(\mathbf{I} - \mathbf{K}_k\mathbf{G}_k)^T + \mathbf{K}_k\tilde{\mathbf{R}}_k\mathbf{K}_k. \tag{9.4.16}$$

To show why artificial perturbation is necessary, let us compare (9.4.16) with the theoretical assimilated covariance given in (9.2.5), namely

$$\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{G}_k)\tilde{\mathbf{P}}_k^f. \tag{9.4.17}$$

If observations in the EnKF were not perturbed, the second term on the right-hand side of (9.4.16) would drop. As a result, the assimilated covariance $\hat{\mathbf{P}}_k$ given by (9.4.16) would be less than the theoretical covariance calculated by (9.4.17) because of the extra multiplier matrix $(\mathbf{I} - \mathbf{K}_k\mathbf{G}_k)^T$. In other words, the assimilated covariance is underestimated. Underestimation of covariance can lead to premature reduction in the ensemble spread and result in filter degeneracy (Burgers et al. 1998; Evensen 2009). The role of perturbed measurement covariance $\tilde{\mathbf{R}}_k$ is to counteract underestimation. The main criticism of the random perturbation approach is that it alters actual measurements and inevitably affects accuracy of assimilation results, especially when the ensemble size is small.

Note that (9.4.16) is not used to calculate $\hat{\mathbf{P}}_k$. If necessary, we can first calculate the updated ensemble anomaly matrix

$$\mathbf{A}_k = \mathbf{A}_k^f + \mathbf{K}_k(\mathbf{D}_k - \mathbf{G}_k\mathbf{A}_k^f), \tag{9.4.18}$$

and then calculate $\hat{\mathbf{P}}_k = (\mathbf{A}_k\mathbf{A}_k^T)/(N-1)$. But, (9.4.13) and (9.4.15) show that neither $\tilde{\mathbf{P}}_k^f$ nor $\hat{\mathbf{P}}_k$ needs to be explicitly calculated and stored in actual EnKF implementation. In fact, during each time step of the EnKF, we only need to calculate the Kalman gain by the second identity in (9.4.13) and then update the ensemble for the next time step by (9.4.15).

Example 9.8 *EnKF update of variable hydraulic conductivity*
Let us consider the 1D porous flow problem given in Example 9.5. However, we now assume that the hydraulic conductivity K is heterogeneous along x (i.e., a spatial random process). We would like to estimate the distribution of K using EnKF. This problem is similar to that presented in Sun et al. (2009a). Constant head boundary conditions 10 [L] and 1[L] are imposed to the left and right sides, respectively. Two wells are located at $x = 30$ [L] and 60 [L], both pumping at a constant rate of 3×10^{-2} [L/T]. The log-transform of K, $Y = \ln K$, is assumed second-order stationary with an exponential-type covariance model (see Sect. 6.2)

$$C_Y(r) = \sigma_Y^2 \exp\left(-\frac{r}{I_Y}\right),$$

where r is separation distance, the variance of Y is $\sigma_Y^2 = 0.5$, and the correlation length is 5 [L].

As part of the EnKF initialization process, 500 random realizations of Y are generated by performing eigenvalue decomposition of Y's covariance matrix given in the above. Note that any other method for generating spatially correlated random numbers may be used here. Each augmented state vector consists of 100 states and 100 Y values, corresponding to the discretized 1D grid. Thus, the size of the augmented ensemble matrix is 200×500. During SDA, the forward model is solved for each realization during each assimilation step. Measurements are taken every 10 [L]. The measurement data are artificially perturbed by white noise with a standard deviation, $\sigma_\varepsilon = 0.01$ [L]. The size of each assimilation step is 0.1 [T], and the total number of steps is 6.

Figure 9.7a shows the evolution of the "true" state (i.e., hydraulic head) along the x-axis and at different assimilation time steps. Recall that the initial head is 10 [L]. Figure 9.7b shows the initial and final ensemble mean solutions of Y obtained by EnKF. Starting with an essentially flat initial value, the final solution given by the EnKF is significantly improved from the initial guess. Because of the relative sparsity of head observations and the Gaussian nature of the EnKF, the final ensemble mean only provides a smooth version of the synthetic truth. Figure 9.7c shows the reduction of RMSE with time. The greatest RMSE decrease happens after the first update step, and then, it becomes relatively flat.

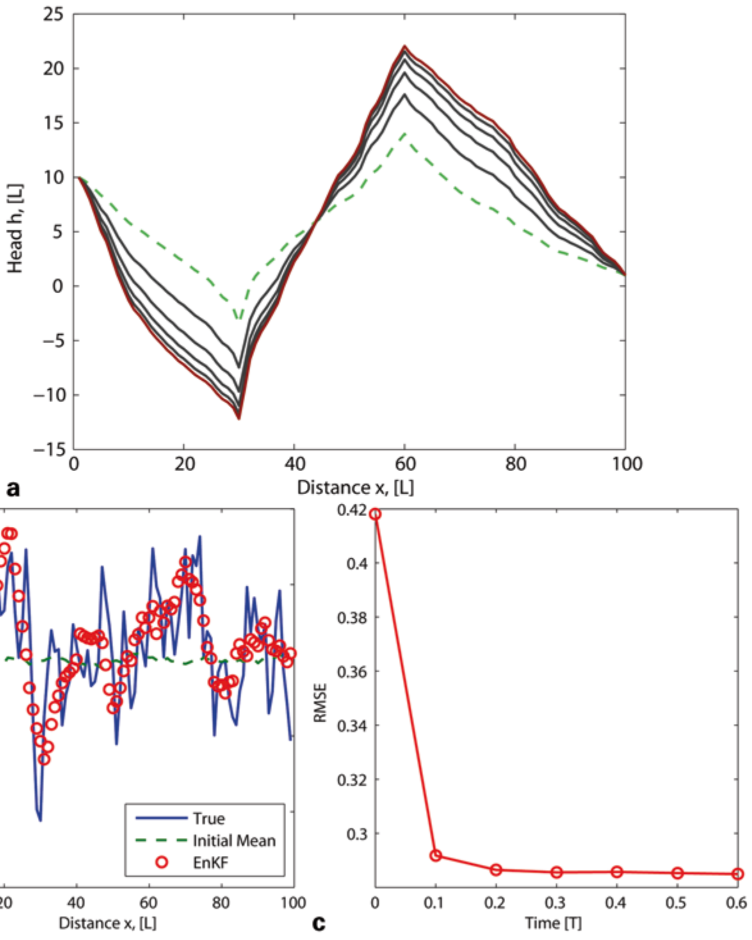


Fig. 9.7 **a** Evolution of head with time (the *dash line* represents head distribution after first time step); **b** history of assimilated $Y = \ln K$ along x, where the *blue solid line* is the synthetic truth, the *green dashed line* corresponds to the initial ensemble mean, and *open circles* represent the final ensemble mean obtained by EnKF; and **c** reduction of RMSE with time. Standard deviation of artificial measurement noise is $\sigma_\varepsilon = 0.01$ [L]

Figure 9.8 shows the effect of increasing the artificial noise level to 1.0 [L]. The assimilated parameter profile becomes more smoothed, and the RMSE reduction is slower than the previous case. During EnKF assimilation, the parameters are modified after each step. The "consistency" school of thoughts advocates that the forward model should be rerun from the initial time to the current time using modified parameters so that the distribution of state variables is "consistent" with current estimates of parameters (Aanonsen et al. 2009). However, several studies indicated that the rerun step is not only unnecessary, but also detrimental to future assimilation results (Zhao et al. 2008; Zafari and Reynolds 2005). Therefore, the rerun step is not done in this example.

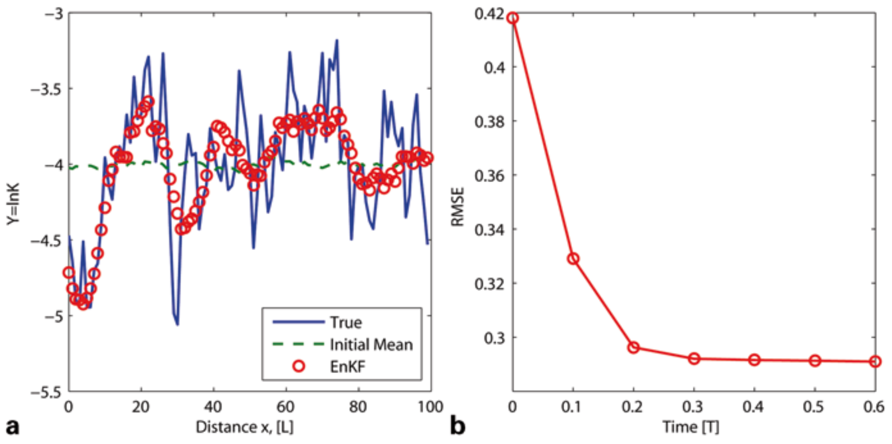


Fig. 9.8 **a** History of assimilated $Y = \ln K$ and **b** reduction of RMSE with time, when standard deviation of artificial measurement noise is set to 1.0 [L]

9.4.2 Variants of the EnKF

The use of perturbed observations in the classical EnKF results in a suboptimal filter. Several variants of the classical EnKF have been proposed in the literature, in part to address this issue.

9.4.2.1 Deterministic EnKF

The deterministic EnKF (DEnKF) attempts to make the assimilated covariance asymptotically match the theoretical KF covariance (9.4.17) without requiring perturbing the actual measurements (Sakov and Oke 2008a). To start, let us revisit the EnKF analysis covariance (9.4.16) without the perturbed noise covariance term

$$\begin{aligned}\hat{\mathbf{P}}_k &= (\mathbf{I}-\mathbf{K}_k\mathbf{G}_k)\mathbf{P}_k^f(\mathbf{I}-\mathbf{K}_k\mathbf{G}_k)^T \\ &= \mathbf{P}_k^f - 2\mathbf{K}_k\mathbf{G}_k\mathbf{P}_k^f + \mathbf{K}_k\mathbf{G}_k\mathbf{P}_k^f\mathbf{G}_k^T\mathbf{K}_k^T.\end{aligned} \tag{9.4.19}$$

Sakov and Oke (2008a) observed that if the product of Kalman gain and measurement operator, $\mathbf{K}_k\mathbf{G}_k$, is small, the quadratic term $\mathbf{K}_k\mathbf{G}_k\mathbf{P}_k^f\mathbf{G}_k^T\mathbf{K}_k^T$ becomes small, and one can asymptotically match the KF-assimilated covariance by halving the Kalman gain $\mathbf{K}_k$. In the DEnKF, the updated ensemble anomaly matrix $\mathbf{A}_k$ in (9.4.18) is changed to (Sakov and Oke 2008a)

$$\mathbf{A}_k = \mathbf{A}_k^f - \frac{1}{2}\mathbf{K}_k\mathbf{G}_k\mathbf{A}_k^f, \tag{9.4.20}$$

and the corresponding assimilated covariance becomes

$$\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{G}_k)\mathbf{P}_k^f + \frac{1}{4}\mathbf{K}_k\mathbf{G}_k\mathbf{P}_k^f\mathbf{G}_k^T\mathbf{K}_k^T. \tag{9.4.21}$$

From (9.4.21), we see that the DEnKF scheme always overestimates the KF covariance because of the second term appearing on the right-hand side of the equation; thus, it essentially imposes a covariance inflation mechanism without perturbing the measurements artificially. Tests of DEnKF on different problems (Sakov and Oke 2008a; Sun et al. 2009a; Simon and Bertino 2012) have indicated that DEnKF consistently outperforms the classical EnKF. We remark that the name "deterministic" is largely a misnomer because DEnKF is still a Bayesian algorithm.

9.4.2.2 Square Root EnKFs

Square root EnKFs represent a class of reduced-rank EnKFs that involve certain type of decomposition of the covariance matrix. Examples of such filters include the singular evolutive interpolate Kalman (SEIK) filter (Pham 2001), ensemble adjustment KF (EAKF) (Anderson 2001), and the ensemble transform KF (ETKF) (Bishop et al. 2001). A general review of the ensemble square root filters can be found in Sakov and Oke (2008c).

The basic idea underlying square root EnKFs is to multiply the ensemble perturbation matrix $\mathbf{A}_k^f$ with a transform matrix such that the analysis ensemble covariance matches theoretical KF covariance without needing to perturb measurements artificially (Sakov and Oke 2008b; Tippett et al. 2003). Toward this end, let us define a transformed anomaly matrix as

$$\tilde{\mathbf{A}}_k^f = \mathbf{A}_k^f\mathbf{T}_k, \tag{9.4.22}$$

where $\mathbf{T}_k$ is transform matrix to be determined. Substituting (9.4.22) into the theoretical KF covariance in (9.2.5) gives

$$(\mathbf{A}_k^f\mathbf{T}_k)(\mathbf{A}_k^f\mathbf{T}_k)^T = (\mathbf{I} - \mathbf{K}_k\mathbf{G}_k)\mathbf{A}^f(\mathbf{A}^f)^T. \tag{9.4.23}$$

A general form of transform matrix $\mathbf{T}_k$ is (Bishop et al. 2001)

$$\mathbf{T}_k = \mathbf{S}_k\mathbf{U}, \tag{9.4.24}$$

$$\mathbf{S}_k = \left[\mathbf{I} + \frac{1}{N-1}(\mathbf{G}_k\mathbf{A}_k^f)^T\mathbf{R}_k^{-1}\mathbf{G}_k\mathbf{A}_k^f\right]^{-1/2}, \tag{9.4.25}$$

in which $\mathbf{U}$ is an arbitrary orthogonal matrix satisfying $\mathbf{U}\mathbf{U}^T = \mathbf{I}$ and $\mathbf{R}$ is the actual measurement error covariance. Performing eigenvalue decomposition on the term in square brackets of (9.4.25) gives

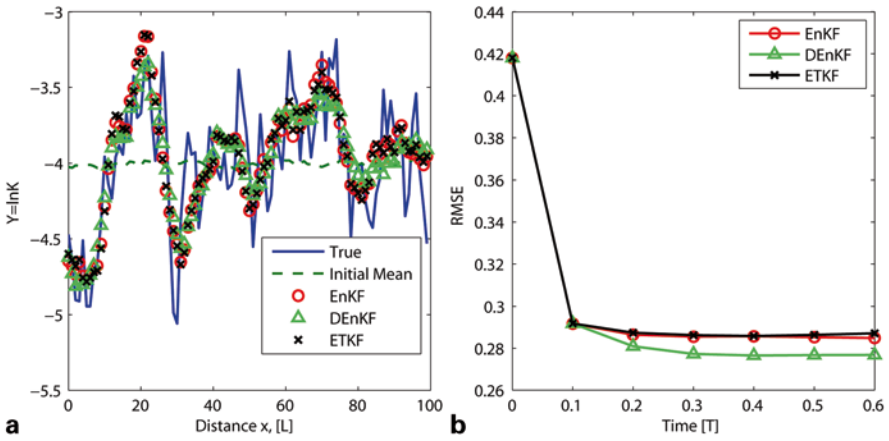


Fig. 9.9 Comparison of the performance of EnKF, DEnKF, and ETKF on the 1D flow problem: **a** Y solutions and **b** reduction of RMSE

$$\mathbf{I} + \frac{1}{N-1}(\mathbf{G}_k\mathbf{A}_k^f)^T\mathbf{R}_k^{-1}\mathbf{G}_k\mathbf{A}_k^f = \mathbf{E}_k\mathbf{\Gamma}_k\mathbf{E}_k^T, \tag{9.4.26}$$

where $\mathbf{\Gamma}_k$ is the diagonal eigenvalue matrix. Thus, $\mathbf{S}_k$ in (9.4.25) can be written in a simplified form after taking the square root of the inverse of $\mathbf{\Gamma}_k$

$$\mathbf{S}_k = \mathbf{E}_k\mathbf{\Gamma}_k^{-1/2}\mathbf{E}_k^T. \tag{9.4.27}$$

The transform matrix for ETKF is (Sakov and Oke 2008b)

$$\mathbf{T}_k = \mathbf{E}_k\mathbf{\Gamma}_k^{-1/2}\mathbf{E}_k^T\mathbf{U}.$$

The computational demand of square root filters is higher than EnKF and DEnKF because of the additional eigenvalue decomposition step.

Example 9.9 *Comparison of EnKF, DEnKF, and ETKF*
The performance of EnKF, DEnKF, and ETKF is compared using the 1D porous flow problem described in Example 9.8. Overall, all three ensemble filters capture the spatial trend of $Y = \ln K$ well (Fig. 9.9a). DEnKF obtained the best performance, whereas the results of EnKF and ETKF are similar (Fig. 9.9b). Sun et al. (2009a) conducted a more comprehensive comparison of filter performance using both 1D and 2D porous flow problems and found that the DEnKF is consistently the most robust filter in all test cases and gives the best performance at relatively small ensemble sizes; also, the detrimental effect of artificial sampling noise on EnKF becomes more pronounced when observations are sparse, whereas DEnKF and square root filters do not suffer from such issue.

9.4.2.3 Covariance Inflation and Localization

So far, we have mentioned covariance inflation several times without a formal discussion. The reduced-rank nature of EnKF (i.e., use of ensemble statistics) inevitably causes insufficient variance in estimated state covariance matrix. Covariance inflation is an ad hoc technique that attempts to prevent filter degeneracy by inflating either forecast or analysis covariance during each assimilation cycle. Note that this covariance inflation is in addition to artificial measurement perturbation in classical EnKF, which is done to compensate for algorithmic deficiency.

Both additive and multiplicative inflation can be applied to the ensemble to boost the ensemble forecast covariance. Multiplicative inflation simply multiplies the forecast ensemble covariance (or each ensemble anomaly matrix) by a scalar. Additive inflation adds random perturbation fields with a certain spatial covariance structure to each ensemble member, which is only feasible if the model error can be well characterized. In many applications, especially surface hydrology, there is often insufficient information to determine the structure of additive error and the assignment then becomes rather arbitrary (DeChant and Moradkhani 2012; Clark et al. 2008).

A major assumption underlying covariance inflation is that the prior correlation structure is correct, but estimates of the variance of individual state vector components are too small (Anderson 2009). To account for spatiotemporal correlation in model errors, some suggested that spatially and temporally varying adaptive covariance inflation be used (Anderson 2009; Evensen 2003). Thus, for the classical EnKF, both forecast covariance and measurement covariance can be inflated, although numerical experiments suggest that multiplicative inflation is more useful for square root EnKFs (Sun et al. 2009a).

Localization is another ad hoc technique for improving filter stability. It has been observed in atmospheric modeling studies that sampling error (due to limited sample size) can result in spurious correlations in space, causing updates to state variables in regions of no real influence (Lorenc 2003; Houtekamer and Mitchell 2001). The basic idea behind localization is to restrict the radius of influence of each observation so that a certain observation only affects the state variables that are close to it in the physical space. An alternative rationale is that localization solves for a small model state in a relatively large ensemble space and, thus, allows for a larger flexibility in the analysis scheme to reach different model solutions (Hamill et al. 2001; Houtekamer and Mitchell 2001). In the literature, two types of location techniques have been used: (i) distance-dependent covariance localization or covariance tapering (Agbalaka and Oliver 2008; Houtekamer and Mitchell 2001; Hamill et al. 2001) and (ii) moving-window-based localization (Sun et al. 2009b; Szunyogh et al. 2008).

The moving-window-based localization approach is physically more intuitive. Sun et al. (2009b) demonstrated that localization can be an effective strategy by itself for reducing parameter uncertainty in multimodal parameter fields, especially when the initial ensemble is well constrained by prior data. However, localization is more computationally intensive because it requires solution of many local ensemble filtering problems, as opposed to a single global filtering problem. Also, discontinuity or "edge effect" in updated state and parameters may appear at local window boundaries because different measurements are used during solution of each local filtering problem. Sun et al. (2009b) suggested that by randomly updating the nodes in a grid, as opposed to updating in regular row/column order, the "edge effect" can be effectively suppressed.

9.5 Particle Filters

Particle filters are based on the simple concept that any integral (in our case PDFs) can be approximated using a finite number of independent random samples called *particles*. This is the same idea behind the Monte Carlo integration that we have introduced in the context of MCMC (Sect. 4.3). Particle filters do not require assumptions about the state model and observation error. The state model can be nonlinear, and the observation error distribution can be non-Gaussian. But, this kind of filter does not perform well for high-dimensional systems due to the limited number of particles that can be used in practice.

We now reformulate the problem using the Bayesian filtering framework considered in Sect. 9.1. Our starting point is the posterior PDF shown in (9.1.16) for augmented state variable $\mathbf{y}_k$ after having measurement $\mathbf{z}_k$, which is written in a recursive form below

$$p(\mathbf{y}_k|\mathbf{z}_{1:k}) \propto p(\mathbf{z}_k|\mathbf{y}_k)\int p(\mathbf{y}_k|\mathbf{y}_{k-1})p(\mathbf{y}_{k-1}|\mathbf{z}_{1:k-1})d\mathbf{y}_{k-1}. \tag{9.5.1}$$

This distribution can be evaluated recursively as new data become available. However, the marginal in (9.5.1) is usually not tractable analytically. It is also difficult to sample directly from the prior, $p(\mathbf{y}_{k-1}|\mathbf{z}_{1:k-1})$, because of its complex shape and high dimensionality. Like in MCMC, we would like to construct an easy-to-implement sampling density, or proposal distribution, $q(\mathbf{y})$, which ideally should be as close to the prior $p(\mathbf{y}_{k-1}|\mathbf{z}_{1:k-1})$ as possible. If $q(\cdot)$ deviates from the prior, some particles from $q(\cdot)$ will not be useful because of their negligible contributions. In the worst case, if $q(\cdot)$ and the prior are completely separated, all particles from $q(\cdot)$ will not contribute to the evaluation of the integral.

To help construct a good proposal distribution, *sequential importance sampling* (SIS) was introduced to enable recursive estimate of particle weights over time (Doucet et al. 2001a). The starting point of SIS is to decompose the proposal distribution in the following form:

$$q(\mathbf{y}_{1:k}|\mathbf{z}_{1:k}) = q(\mathbf{y}_1|\mathbf{z}_1)\prod_{i=2}^{k} q(\mathbf{y}_i|\mathbf{y}_{1:i-1},\mathbf{z}_{1:i}). \tag{9.5.2}$$

Denoting the set of N particles at the kth time step by $\{\mathbf{y}_k^{(i)}\}_{i=1}^N$ and the associated weights by $\{w_k^{(i)}\}_{i=1}^N$, our task is concerned with updating the weights using new data, for which a recursive formula is given as (Doucet et al. 2000):

$$\begin{aligned} w_k^{(i)} &= \frac{p(\mathbf{y}_{1:k}^{(i)} \mid \mathbf{z}_{1:k})}{q(\mathbf{y}_{1:k}^{(i)} \mid \mathbf{z}_{1:k})} \\ &\propto \frac{p(\mathbf{z}_k \mid \mathbf{y}_k^{(i)})p(\mathbf{y}_k^{(i)} \mid \mathbf{y}_{k-1}^{(i)})p(\mathbf{y}_{1:k-1}^{(i)} \mid \mathbf{z}_{1:k-1})}{q(\mathbf{y}_k^{(i)} \mid \mathbf{y}_{1:k-1}^{(i)}, \mathbf{z}_{1:k})q(\mathbf{y}_{1:k-1}^{(i)} \mid \mathbf{z}_{1:k-1})}, \\ &= w_{k-1}^{(i)} \frac{p(\mathbf{z}_k \mid \mathbf{y}_k^{(i)})p(\mathbf{y}_k^{(i)} \mid \mathbf{y}_{k-1}^{(i)})}{q(\mathbf{y}_k^{(i)} \mid \mathbf{y}_{1:k-1}^{(i)}, \mathbf{z}_{1:k})} \end{aligned} \tag{9.5.3}$$

where $w_{k-1}^{(i)}$ is the prior importance weight of the ith particle. Initially, all particles are assigned with the same weight. The weights in (9.5.3) is further normalized by their sums

$$\tilde{w}_i = \frac{w_k^{(i)}}{\sum_{i=1}^N w_k^{(i)}}.$$

Equation (9.5.3) suggests that a suitable proposal distribution would be one that minimizes the variance of weights conditional upon past particle "trajectories" $\mathbf{y}_{1:k-1}^{(i)}$ and observations $\mathbf{z}_{1:k}$. One such possibility was suggested by Doucet et al. (2000)

$$q(\mathbf{y}_k^{(i)} \mid \mathbf{y}_{1:k-1}^{(i)}, \mathbf{z}_{1:k}) = p(\mathbf{y}_k^{(i)} \mid \mathbf{y}_{k-1}^{(i)}, \mathbf{z}_k), \tag{9.5.4}$$

which has also been used extensively in the MCMC literature (e.g., Liu and Chen 1998). However, the proposal distribution in (9.5.4) requires the ability to evaluate the likelihood, $p(\mathbf{z}_k \mid \mathbf{y}_{k-1}^{(i)}) = \int p(\mathbf{z}_k \mid \mathbf{y}_k)p(\mathbf{y}_k \mid \mathbf{y}_{k-1}^{(i)})d\mathbf{y}_k$ and to sample from $p(\mathbf{y}_k^{(i)} \mid \mathbf{y}_{k-1}^{(i)}, \mathbf{z}_k)$, which has no close form in the general case. A suboptimal choice is to set the proposal distribution to state transition PDF, $p(\mathbf{y}_k^{(i)} \mid \mathbf{y}_{k-1}^{(i)})$, which is not conditioned on new data. Under this simplification, the recursive weight update formula given in (9.5.3) becomes

$$w_k^{(i)} = w_{k-1}^{(i)} p(\mathbf{z}_k \mid \mathbf{y}_k^{(i)}). \tag{9.5.5}$$

which simply updates importance weights using likelihood function.

SIS can quickly run into filter degeneracy after several update cycles, whence the particle set is dominated by only a few particles with large weights, while all others have negligible weights. An indicator of SIS degeneracy is the (approximate) effective sample size, N_{eff} (Liu 1996):

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (\tilde{w}_k^{(i)})^2}. \tag{9.5.6}$$

Initially, all particles have equal weights $1/N$ and N_{eff} is equal to N. When N_{eff} falls below a preset threshold value, say $N/2$, it signals that the particle set will need some correction for degeneracy. A naïve remedy to SIS degeneracy is to bootstrap the particle set, the net effect of which is equivalent to duplicating particles with large weights and discarding those with small weights. This is known as the resampling step, and the resulting particle filter is referred to as *sequential importance resampling* (SIR) filter. Unfortunately, resampling does not resolve the filter degeneracy issue completely because the particle set is gradually filled with identical particles, leading to sample impoverishment. A large number of alternative importance resampling schemes have been improvised in the last decade, including multinomial sampling, residual sampling, systematic resampling, and stratified resampling, for which reviews and comparison studies can be found in (Cappé et al. 2007; Douc and Cappé 2005; Hol et al. 2006). The biggest challenge to practical use of particle filters is sampling the high-dimensional parameter space, which partly explains why applications of particle filter in large-scale EWR problems are nonexistent. Recently, a number of studies have demonstrated the merits of particle filters for conceptual hydrologic models (DeChant and Moradkhani 2012; Moradkhani et al. 2012; Rings et al. 2012; Salamon and Feyen 2009; Vrugt et al. 2013). Liu et al. (2012) provided a summary of recent applications of data assimilation and particle filters in hydrology.

9.6 Review Questions

1. Show the pair of prediction and measurement equations that involve system states, parameters, and control variables.
2. Extend the Bayes' theorem in Eq. (9.1.11) to include the estimation of system states, parameters, and control variables. Then, extend Eq. (9.1.14) to this case.
3. Give the details of deriving Eq. (9.2.4) from Eqs. (4.2.21) and (4.2.22) in Sect. 4.2.3. What assumptions do we need to obtain these results?
4. Under what conditions can we use EKF for parameter estimation?
5. What is the difference between EKF and UKF in handling the model nonlinearity?

6. Summarize the forecast and assimilation algorithms of GMM described in Sect. 9.3.3.
7. Discuss how EnKF is related to KF and UKF and describe its main advantages over the two SDA algorithms.
8. Why the classical EnKF requires artificial noise perturbation? What is the adverse side effect of the artificial noise perturbation approach?
9. What is covariance inflation and why is it needed?
10. Use vector and matrix notations to show a flowchart that describes the step-by-step algorithm of the classical EnKF.
11. Explain how the MCMC approach is used in a particle filer to explore the joint posterior distribution. Explain the cause of SIR collapse.

Chapter 10
Model Uncertainty Quantification

Uncertainty quantification (UQ) is the analytic process of determining the effect of input uncertainties (both their magnitudes and sources) on system outcomes. Traditionally applied in engineering reliability analysis, UQ now plays a significant role in environmental and water resource (EWR) applications as environmental engineers and modelers are increasingly involved in designing or permitting complex systems, including examining their long-term environmental impacts and providing decision support under risks. *Risk analysis* is the process of determining the consequence of uncertain and often undesirable outcomes. UQ provides inputs to risk analysis which, in turn, provides bases for decision making and for improving data collection design (Fig. 10.1). UQ and risk analysis are integral and often required components of EWR applications nowadays. Examples of high-risk and high-profile EWR projects are many, such as siting of radioactive waste repositories or geologic carbon sequestration reservoirs. For these projects, the broad aim of UQ and risk analysis is to describe a potentially hazardous situation in a manner as accurate, thorough, and decision-relevant as possible, addressing the significant concerns of the interested and affected parties, and making this information understandable and accessible to public officials and to the stakeholders (Stern et al. 1996).

Advancements in computing and data collection techniques have drastically changed the landscape of EWR modeling in the last several decades. At the same time, with the permeation of social media, societal expectations and scrutiny of EWR models and predictions are becoming greater than ever, with a diminishing tolerance to mistakes. Environmental decision making has shifted to be more of a participatory and iterative process, requiring not only effective communication and collaboration but also compromises among diverse interest groups. The participatory approach to environmental decision making is arguably more effective and transparent than the traditional top-down approach because it attempts to involve stakeholders from the onset of decision-making processes and is, in principle, less prone to ideological clashes. Every step of a decision-making process can involve iterative analysis and deliberation steps. During deliberation, participants collectively decide which harms to analyze and how to describe scientific uncertainty and disagreement (Stern et al. 1996). A system or platform used to support collaborative

N-Z. Sun, A. Sun, *Model Calibration and Parameter Estimation,*
DOI 10.1007/978-1-4939-2323-6_10

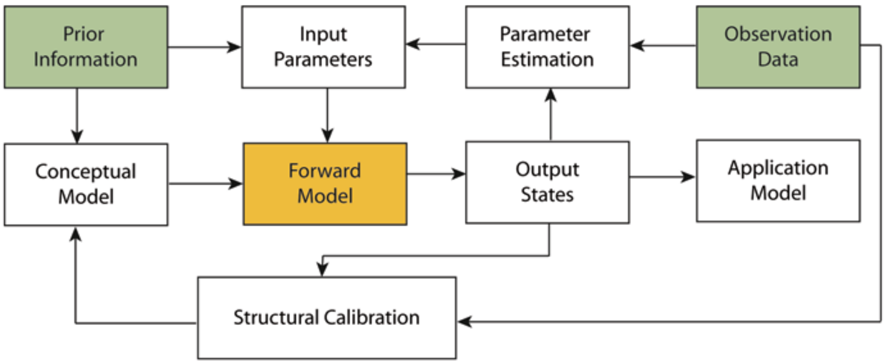


Fig. 10.1 Flowchart of model construction and application, where *arrows* show the directions of uncertainty propagation

decision making must operate on the premise of reliable and mutually agreeable models. However, the performance of a model can only be assessed according to the available data and information (thereby, involving some degree of uncertainty) and to the best of one's knowledge (thereby, involving some aspects of ignorance).

The complexity and challenges of UQ arise from multiple sources, including difficulties in characterizing origins and causes of the unknowns, measuring their magnitudes, and evaluating their impacts on system outputs. Adding to the complexity of UQ is the existence of a wide spectrum of subtly different uncertainty definitions and the lack of a general taxonomy. The main aim of this chapter is to introduce concepts and methods for UQ in EWR applications. Uncertainties to be dealt with under this chapter are assumed recognizable, measureable, and quantifiable. When the uncertainty is not recognizable, it is ignorance. The latter often puts risk analysts on the horn of dilemma—the combination of low-probability extreme events can happen and often leads to disastrous consequences. Arguably, the ability of recognizing limitations of UQ in the context of a specific decision-making activity is as important as conducting the UQ itself.

In Sect. 10.1, taxonomies and classifications of uncertainties used in the literature are reviewed; several uncertainty measures for deterministic and stochastic variables are introduced; and the most general tool used for UQ, the Monte Carlo method and Latin hypercube sampling (LHS) technique are described. Section 10.2 introduces the global sensitivity analysis (GSA) method that finds the total effect of all model input factors to the variance of the model output simultaneously without using linearization. The problem of how to screen noninfluential parameters to model outputs and model applications is addressed. Section 10.3 is dedicated to the stochastic methods of UQ. The stochastic response surface methods (SRSMs), including polynomial chaos and stochastic collocation, are introduced in detail. Finally, Sect. 10.4 tackles the challenging problem of characterizing model structural uncertainty from the model averaging perspective.

While reading this chapter, readers should keep in mind that UQ is not a standalone subject, nor is it a single group of techniques. It is intertwined with many other topics, several of which have been either explicitly or implicitly touched upon in previous chapters. The purpose of uncertainty characterization and reduction has been mentioned in Chaps. 2–4 in the context of parameter estimation and model calibration. UQ is closely related to sensitivity methods discussed in Chap. 5. Chapters 6 and 7 discussed parameterization techniques and model structure identification methods that are needed to perform UQ, while Chaps. 8 and 9 discussed methods for continuous model uncertainty reduction and for dealing with high-dimensional systems. In practice, we envision that UQ and inversion are done in a closed-loop fashion—after model calibration, we quantify its prediction uncertainty which then provides insights into future data collection strategies (Fig. 10.1). This is the broad definition of data assimilation mentioned in Chap. 9. UQ also plays an important role in experimental design and goal-oriented modeling, which will be covered in Chaps. 11 and 12.

10.1 Basic Concepts

10.1.1 Definitions and Taxonomy

Definitions of uncertainty vary in different disciplines, depending on the subject of study and purpose of use. Some authors broadly define uncertainty as "any deviation from the unachievable ideal of completely deterministic knowledge of the relevant system" (Walker et al. 2003). Others make distinction between knowledge incompleteness caused by absence (incompleteness in kind) and uncertainty (incompleteness in degree); along this vein, *absence* refers to gaps in knowledge, whereas *uncertainty* refers specifically to that part of "knowledge incompleteness caused by inherent deficiencies in acquired knowledge" (Ayyub and Klir 2006; Bammer and Smithson 2008; Smithson 1989). A subjective interpretation of uncertainty has also been proposed, for which the main focus is on the degree of confidence (or lack of confidence) that a decision maker has about possible outcomes and/or probabilities of these outcomes (Refsgaard et al. 2007).

Given the many definitions of uncertainty, it is not surprising that different uncertainty taxonomies exist. For example, if classified on the basis of its origins, uncertainty can include (Refsgaard et al. 2007):

- *Context uncertainty*, which is the uncertainty regarding problem domain boundaries and external circumstances (e.g., social, economic, and environmental) that form the problem context;
- *Input uncertainty*, which refers to the uncertainty in external driving forces (e.g., precipitation, land use/land cover pattern, and recharge rates) and data that drive the model;
- *Model structural uncertainty*, which is caused by incomplete understanding of physical processes and/or due to simplification of a system under study;
- *Parameter uncertainty*, which is a result of incomplete or imprecise knowledge about model parameters; and

- *Technical implementation uncertainty*, which arises mostly during implementation and solution of numerical models.

A taxonomy that is commonly used in risk analyses classifies uncertainty according to its reducibility:

- *Aleatory uncertainty* (also known as stochastic uncertainty), which is caused by inherent randomness that is perceived to be nondeterministic and irreducible in nature;
- *Epistemic uncertainty* (also known as subjective uncertainty), which is a result of incomplete knowledge and which can be potentially reduced when new information is gained.

Aleatory uncertainty is usually quantifiable by using probabilistic methods. On the other hand, epistemic uncertainty is more subjective, making the use of probability distribution assertion questionable when data are sparse. Therefore, nonprobabilistic methods based on set-theoretic or interval analysis are often used for the latter.

A taxonomy that is often used in social science classifies uncertainty into

- *Vagueness*, which originates from the imprecise nature of the membership of elements in a set or a notion of interest;
- *Likelihood*, which stems from the randomness of outcomes; and
- *Ambiguity*, which results from the possibility of having multiple outcomes for processes or systems.

Finally, four major types of uncertainties have been identified in environmental and ecological decision making (Ascough et al. 2008)

- *Knowledge uncertainty*, which is caused by model and data limitations;
- *Variability uncertainty*, which is caused by the inherent variability manifested in natural and societal systems;
- *Decision uncertainty*, which is related to ill-defined goals, objectives, and performance measures; and
- *Linguistic uncertainty*, which arises because natural human language can be ambiguous and subject to multiple interpretations.

The notion of multiple outcomes is an important one to recognize in UQ. If all possible outcomes of a system are known, we deal with *bounded* uncertainty, and probabilities can be assigned to each outcome. On the other hand, if only a subset of outcomes can be recognized, we deal with *unbounded* uncertainty. Identifying all system outcomes and subsequently assigning probability to each outcome in an undisputed manner is a daunting task.

Which model to use? Questions like this are often at the center of debate in a decision-making process, especially when differences in probabilities are perceived as elevated risks and financial losses down the road. Thus, in UQ more is less—the use and the correct use of more than one UQ methods may greatly lessen the controversial nature of environmental decision-making processes and clear some major hurdles to the use of uncertainty analysis in practice. A risk analyst should strive for an accurate, balanced, and informative uncertainty analysis, one that incorporates state-of-the-art UQ tools and still remains comprehensible and usable by decision makers and other participants. Although not covered here, visual analytics is an emerging discipline that can be tremendously useful for enhancing interpretation of uncertainty analysis results (Heer, Agrawala 2008; Keim et al. 2008; Andrienko et al. 2007; Sun 2013).

10.1.2 Model Uncertainty Propagation

This chapter is mainly devoted to model uncertainties. Figure 10.1 shows a flow-chart of modeling from data collection, forward solution, inverse solution, to model application. During this process, the following uncertainties are involved

1. Prior information uncertainty
2. Data uncertainty
3. Input parameter uncertainty
4. Conceptual model uncertainty
5. Model output uncertainty
6. Parameter estimation uncertainty
7. Structure identification uncertainty
8. Model application uncertainty

From Fig. 10.1, we can see that (i) the uncertainties associated with prior information and observation data are the sources of all other uncertainties (the uncertainty caused by numerical error is not shown explicitly in the figure), (ii) the uncertainty propagates from one module to the next module one by one, and (iii) the propagation of uncertainty happens in two-way flows, the forward solution flow and the inverse solution flow.

Along the forward solution flow, the uncertainty in prior information is propagated from model inputs to model applications, while the inverse solution flow transfers the information in observation data to decrease the model input uncertainty. The ultimate goal of model construction is to decrease the uncertainty associated with model application to an acceptable level.

As shown in Fig. 10.2, a module can be seen as a function, or more generally, a mapping, $\mathbf{u} = \mathcal{F}(\boldsymbol{\theta})$, that transfers the module input $\boldsymbol{\theta}$ together with its variation $\delta\boldsymbol{\theta}$ to the module output $\mathbf{u}$ and generates a variation $\delta\mathbf{u}$ given by

$$\delta\mathbf{u} = \mathcal{F}(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) - \mathcal{F}(\boldsymbol{\theta}) \tag{10.1.1}$$

The major problems in the UQ study are how to measure different kinds of uncertainty and how to estimate the change in uncertainty through a module or a model.

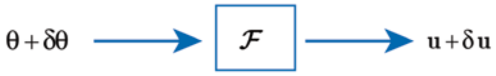


Fig. 10.2 Uncertainty propagation through a module $\mathcal{F}$

10.1.3 Measures of Uncertainty

"Measure," as a theoretical problem, has been systematically studied in mathematics and extensively used in almost all disciplines. Besides measuring size and weight, we need to measure functions, evidence, risks, information, satisfaction, uncertainty, and much more. The measure theory in mathematics provides concepts, principles, and methods for measuring a "set" if it is "measureable." Generally speaking, a measure of a set A, denoted by $\mu(A)$, is a number in the interval $[0, +\infty)$ and satisfies at least two requirements:

$$\begin{aligned} &\text{(i) } \mu(A) = 0, \text{when } A = \varnothing \text{ (an empty set)} \\ &\text{(ii) } \mu(A) \le \mu(B), \text{ when } A \subseteq B \text{ } (A \text{ belongs to } B) \end{aligned} \tag{10.1.2}$$

We will not discuss the measure theory in depth. Readers interested in this issue may refer to, for example, Bogachev (2007). This section only introduces several measures that are useful for quantification of model uncertainties. Most of them have been used in the previous chapters.

Range Measure The most often used method of measuring the uncertainty of a variable is the size of its variation range (interval). For example, let $\mathbf{a}$ and $\mathbf{b}$ be the upper and lower bounds, respectively, of the identified parameter vector $\boldsymbol{\theta}$ based on existing evidence, we can use the Euclidian norm $\mu(\mathbf{a}, \mathbf{b}) = \|\mathbf{b} - \mathbf{a}\|$ as an uncertainty measure of $\boldsymbol{\theta}$. If we can find new evidence to support the increase of $\mathbf{a}$ and/or the decease of $\mathbf{b}$, the uncertainty of $\boldsymbol{\theta}$ is decreased.

Hartley Measure Let θ be a discrete variable that has an equal chance to be any element of a set $E = \{\theta_1, \theta_2, \cdots, \theta_n\}$, the nonspecificity uncertainty of θ is measured by the following Hartley measure introduced by R. Hartley (Hartley 1928):

$$\mathcal{H}_0(E) = \log\left(|E|\right), \text{ where } \theta \in E \text{ and } |E| = n. \tag{10.1.3}$$

This measure can be used when the unknown θ is one of the n candidates. For example, when $n = 10$ and the base of logarithm is 2, we have $\mathcal{H}_0 = 3.32$. If we can find new evidence to decrease the number of candidates to three, the uncertainty of x will be decreased to $\mathcal{H}_0 = 1.58$. When the number of candidates reduces to one, there is no uncertainty and $\mathcal{H}_0 = 0$. For more discussions on the Hartley measure, readers may refer to Ayyub and Klir (2006).

Shannon Entropy Measure Shannon entropy measure introduced by C.E. Shannon (Shannon 1948) is widely used to measure uncertainties or conflicts associated probability assignments. For a discrete random variable with n outcomes $\{\theta_1, \theta_2, \cdots, \theta_n\}$, the Shannon entropy is defined as

$$\mathcal{H}_s(p) = -\sum_{i=1}^{n} p(\theta_i) \log p(\theta_i), \tag{10.1.4}$$

where $p(\theta_i)$ is the probability mass function of outcome θ_i. Shannon entropy is an extension of the Hartley measure when there is additional information that can support the assignment of different weights to different candidates. In fact, when all weights are equal (i.e., $p(\theta_i) = n^{-1}$ for all $i = 1, 2, \cdots, n$), we have $\mathcal{H}_s(p) = \log(n)$ which is the same as the Hartley measure. On the other hand, if additional information can support the assignment of more nonuniform probability distribution, the value of $\mathcal{H}_s(p)$ decreases and finally becomes zero when the probability associated with one outcome becomes one. In this case, there is no more uncertainty.

For a continuous random variable θ over the whole range $\mathbb{R}$, once its probability distribution $p(\theta)$ is given based on the existing information, its uncertainty can be measured by rewriting (10.1.4) into the following continuous form

$$\tilde{\mathcal{H}}_s(p) = -\int_{\mathbb{R}} p(\theta) \log p(\theta) d\theta. \tag{10.1.5}$$

Unfortunately, the so-defined entropy is not qualified to be a measure because its value may become negative. For example, when $p(\theta)$ is a Gaussian distribution $\mathcal{N}(\mu, \sigma)$, from Example 4.2, we have

$$\tilde{\mathcal{H}}_s(p) = \log \sigma + \frac{1}{2}(1 + \log 2\pi).$$

When σ is small enough, $\tilde{\mathcal{H}}_s(p)$ becomes negative. Other problems associated with (10.1.5) are that the value of $\tilde{\mathcal{H}}_s(p)$ depends on the chosen coordinate system, and the improper integral (10.1.5) may not be convergent (Ayyub and Klir 2006). These problems can be overcome by introducing the relative information entropy

$$\mathcal{H}_s(p) = -\int_{\mathbb{R}} p(\theta) \log \frac{p(\theta)}{p_M(\theta)} d\theta, \tag{10.1.6}$$

where $p_M(\theta)$ is the density of a uniform distribution.

Equation (10.1.6) can be further extended to measure the uncertainty of a random vector. In Chap. 4, the uncertainty of a continuous m-dimensional random vector $\boldsymbol{\theta}$ with joint PDF $p(\boldsymbol{\theta})$ is measured by Eq. (4.1.11),

$$\mathcal{H}_s(p) = -\int_{\mathbb{R}^m} p(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\theta})}{p_M(\boldsymbol{\theta})} d\boldsymbol{\theta}. \tag{10.1.7}$$

The uncertainties associated with the prior distribution $p_0(\boldsymbol{\theta})$ and the posterior distribution $p_*(\boldsymbol{\theta})$ can thus be measured by $\mathcal{H}_s(p_0)$ and $\mathcal{H}_s(p_*)$, respectively, and the uncertainty reduction caused by inversion is their difference $\mathcal{H}_s(p_0) - \mathcal{H}_s(p_*)$.

Variance-Based Measure When a variable with uncertainty is considered a random variable, we can use its variance σ^2 to measure its uncertainty roughly. Especially, when the variable is normally distributed, we can use its mean and variance to find its confidence intervals. For a random vector, its uncertainty can be characterized by its covariance matrix containing also the correlation information between its components.

In general, covariance matrices are obtained by sampling. When the number of parameters is high, calculating integral (10.1.7) is computationally expensive. As shown in Chap. 4, after the posterior distribution is found by Bayesian inference, instead of calculating entropy, we can use sample covariance matrix obtained by Markov Chain Monte Carlo (MCMC) to characterize the uncertainty of the estimated parameters.

10.1.4 Monte Carlo Simulation

Monte Carlo methods are the most general tool for characterization and quantification of model uncertainty. They do not impose specific assumptions on the model and input parameters. In previous chapters, we have already used Monte Carlo methods in different contexts, such as MCMC for exploring the posterior distribution (Chap. 4) and sequential Monte Carlo methods for data assimilation (Chap. 9). Using a sample-based method to find the propagation of uncertainty through model (10.1.1), we only need to perform the following steps:

1. Determining a sampling range, such as a multidimensional hypercube, a confidence interval, or a discrete set, according to the given uncertainty and type of the input variables $\boldsymbol{\theta}$.
2. Choosing a sampling method, such as uniform sampling, random sampling, and stratified sampling, to generate a set of samples $\left\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_N\right\}$.
3. Calculating the corresponding model outputs $\left\{\mathbf{f}(\boldsymbol{\theta}_1), \mathbf{f}(\boldsymbol{\theta}_2), \cdots, \mathbf{f}(\boldsymbol{\theta}_N)\right\}$ by running the model.
4. Measuring the uncertainty of the model output set to find, for example, its range, sampling covariance, and other statistics, according to the type of output variables.

In the above process, generating the sample set in step 2 is the most important but also a difficult task. Using a small sample set cannot produce meaningful statistical results; on the other hand, using a very large sample set may make the computational effort unaffordable. Uniform and random sampling methods will generate either too dense or too sparse samples when the dimension of the input parameter space becomes large. An efficient sampling method should use the minimum number of samples to make the statistical results satisfy a preset accuracy requirement.

Stratified sampling techniques have been introduced in attempt to improve sampling efficiency by dividing parameter space into strata. One of the most well-known stratified sampling techniques is LHS, which was proposed originally by McKay et al. (1979). It assumes that the sampling range is a "hypercube" determined by the upper and lower bounds of each input component. The efficiency of the LHS arises from the fact that it can be configured with any number samples.

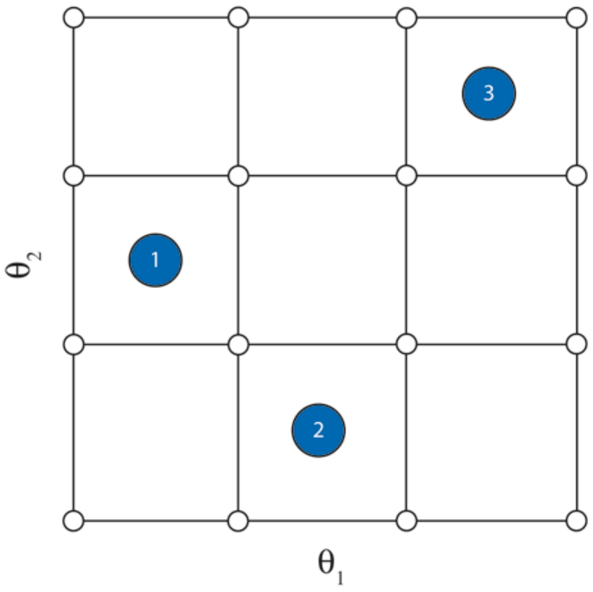


Fig. 10.3 Illustration of the LHS scheme for drawing *three* random samples from a two-dimensional parameter space

For N samples, the LHS method divides each axis of the hypercube into N bins of equal marginal probability, and selects samples from the resulting multidimensional grid. The sampling is done in such a way that for all one-dimensional projections of the N samples, there is one and only one sample in each bin. The LHS scheme is illustrated using a two-dimensional parameter space shown in Fig. 10.3, in which a grid is used for drawing three random samples. The first sample is randomly drawn from the 3×3 grid (nine possibilities). The second sample is drawn from bins that are not on the same row or column as the bin of the first sample (four possibilities). Finally, the last sample can only be sampled from one bin.

By design, LHS guarantees that the ranges of all input variables are adequately represented. For the same number of samples, it has been shown that the sample mean estimated by using LHS has a smaller variance than that estimated by the naïve random sampling (McKay et al. 1979). A number of variants of LHS exist. For example, instead of drawing samples randomly from each bin, some authors suggest using the midpoint of each hyperblock as samples. This is known as the midpoint and lattice LHS (Helton and Davis 2003). It is also possible to generate samples that match a priori sample correlation specified by the user (Iman and Shortencarier 1984). A recent study on the sampling efficiency and convergence of Monte Carlo-based methods for UQ is given by Janssen (2013).

For most situations involving complex models, however, direct Monte Carlo simulation is beyond reach, even with the use of efficient sampling techniques. Although LHS can work with any number of samples, caution must be taken not to make the grid too coarse, implying that a large number of samples are still needed for high-dimensional parameter space.

For a distributed parameter model, the uncertainties caused by parameterization and model structure errors must be considered. In this case, Monte Carlo simulation for UQ would become less efficient after hyperparameters are involved.

10.2 Sensitivity-Based Methods

10.2.1 Local Sensitivity Analysis

Uncertainty propagation through a linear model $\mathbf{u} = \mathbf{A}\boldsymbol{\theta} + \mathbf{b}$ is easy to be quantified, where $\mathbf{A}$ is an $n \times m$ matrix and $\boldsymbol{\theta} = (\theta_1 \ \theta_2 \ \cdots \ \theta_m)^T$. According to (10.1.1), we have $\delta\mathbf{u} = \mathbf{A}\delta\boldsymbol{\theta}$. When the uncertainty of $\delta\boldsymbol{\theta}$ has a range measure $\|\delta\boldsymbol{\theta}\| < \alpha$, the following range measure of $\delta\mathbf{u}$ can be obtained

$$\|\delta\mathbf{u}\| \le \|\mathbf{A}\|\|\delta\boldsymbol{\theta}\| < \|\mathbf{A}\|\alpha \ . \tag{10.2.1}$$

The L_1 and L_2 norms of a vector and the matrix are often used in the above estimation (see Appendix C). When the uncertainty of $\boldsymbol{\theta}$ is represented by its covariance matrix $\mathrm{Cov}(\boldsymbol{\theta})$, the covariance of output **u** is given by

$$\mathrm{Cov}(\mathbf{u}) = (\delta\mathbf{u})(\delta\mathbf{u})^T = \mathbf{A}(\delta\boldsymbol{\theta})(\delta\boldsymbol{\theta})^T\mathbf{A}^T = \mathbf{A}\mathrm{Cov}(\boldsymbol{\theta})\mathbf{A}^T. \tag{10.2.2}$$

For a nonlinear model $\mathbf{u} = \mathbf{f}(\boldsymbol{\theta})$, after linearization around a nominal point $\boldsymbol{\theta}_0$, (10.1.1) gives (see Sect. 2.2.4):

$$\delta\mathbf{u} = \mathbf{f}(\boldsymbol{\theta}_0 + \delta\boldsymbol{\theta}) - \mathbf{f}(\boldsymbol{\theta}_0) \approx \mathbf{J}(\boldsymbol{\theta}_0)\delta\boldsymbol{\theta}, \tag{10.2.3}$$

where $\mathbf{J}(\boldsymbol{\theta}_0)$ is the local sensitivity matrix evaluated at $\boldsymbol{\theta}_0$. Replacing matrix $\mathbf{A}$ by $\mathbf{J}(\boldsymbol{\theta}_0)$ in (10.2.2), we have

$$\mathrm{Cov}(\mathbf{u}) \approx [\mathbf{J}(\boldsymbol{\theta}_0)]\mathrm{Cov}(\boldsymbol{\theta})[\mathbf{J}(\boldsymbol{\theta}_0)]^T. \tag{10.2.4}$$

Example 10.1 *Uncertainty Propagation Through a Linearized Model*
Let us first consider a simple model $u = f(\theta_1, \theta_2)$. According to (10.2.4), we have

$$\begin{aligned}\sigma_u^2 &\approx \begin{bmatrix}\partial u / \partial\theta_1 & \partial u / \partial\theta_2\end{bmatrix}\begin{bmatrix}\sigma_{\theta_1}^2 & \sigma_{\theta_1}\sigma_{\theta_2} \\ \sigma_{\theta_1}\sigma_{\theta_2} & \sigma_{\theta_2}^2\end{bmatrix}\begin{bmatrix}\partial u / \partial\theta_1 \\ \partial u / \partial\theta_2\end{bmatrix} \\ &= \left(\frac{\partial u}{\partial\theta_1}\right)^2\sigma_{\theta_1}^2 + \left(\frac{\partial u}{\partial\theta_2}\right)^2\sigma_{\theta_2}^2 + 2\frac{\partial u}{\partial\theta_1}\frac{\partial u}{\partial\theta_2}\sigma_{\theta_1}\sigma_{\theta_2},\end{aligned} \tag{10.2.5}$$

where the derivatives are evaluated at a point (θ_1^0, θ_2^0) in the input space.

Next, let us consider the following model that has more but independent inputs,

$$u = f(\theta_1, \theta_2, \cdots, \theta_m). \tag{10.2.6}$$

In this case, according to (10.2.4), we have

$$\sigma_u^2 \approx \left(\frac{\partial u}{\partial \theta_1}\right)^2 \sigma_{\theta_1}^2 + \left(\frac{\partial u}{\partial \theta_2}\right)^2 \sigma_{\theta_2}^2 + \cdots + \left(\frac{\partial u}{\partial \theta_m}\right)^2 \sigma_{\theta_m}^2, \tag{10.2.7}$$

where all derivatives are evaluated at a point $\boldsymbol{\theta}_0$ in the input space.

In Chap. 5, the dimensionless (or normalized) sensitivity coefficients are defined in Eq. (5.5.7). If we set $\tau = \sigma_\theta$ and $\lambda = \sigma_u$ in that equation, the variance normalized sensitivity coefficient with respect to θ_i is defined by

$$S_{\sigma,i} = \frac{\sigma_{\theta_i}}{\sigma_u}\left(\frac{\partial u}{\partial \theta_i}\right), \quad i = 1, 2, \cdots, m. \tag{10.2.8}$$

Substituting (10.2.8) into (10.2.7) gives

$$S_{\sigma,1}^2 + S_{\sigma,2}^2 + \cdots + S_{\sigma,m}^2 \approx 1. \tag{10.2.9}$$

By ranking the values of $S_{\sigma,i}^2 (i = 1, 2, \cdots, m)$, we can find the contribution of each input factor in percentage to the model output uncertainty.

The above analysis can be extended straightforwardly to multioutput models

$$u_j = f_j(\theta_1, \theta_2, \cdots, \theta_m), j = 1, 2, \cdots, n. \tag{10.2.10}$$

In this case, (10.2.8) is replaced by

$$S_{\sigma,j,i} = \frac{\sigma_{\theta_i}}{\sigma_{u_j}}\left(\frac{\partial u_j}{\partial \theta_i}\right), \quad j = 1, 2, \cdots, n; \ \ i = 1, 2, \cdots, m \tag{10.2.11}$$

and (10.2.9) is replaced by

$$S_{\sigma,j,1}^2 + S_{\sigma,j,2}^2 + \cdots + S_{\sigma,j,m}^2 \approx 1, j = 1, 2, \cdots, n. \tag{10.2.12}$$

■

The limitations of linearization-based UQ are obvious: (i) the model must be differentiable, (ii) the model must be nearly linear, and (iii) the analysis is not robust.

10.2.2 Global Sensitivity Analysis

Local sensitivity analysis considers the effect of input factors one by one through the calculation of partial derivatives. When the effect of a factor θ_i is evaluated around a point $\boldsymbol{\theta}_0$ by $S^2_{\sigma,i}$, the values of all other input factors are fixed at that point without considering their uncertainties. As a result, the input–output relationship is not thoroughly explored for nonlinear and nonmonotonic models. The GSA, on the other hand, attempts to find the total effect of varying all factors simultaneously and identify those factors that contribute the most to the model output variance.

10.2.2.1 Variance Decomposition

GSA can be considered as an extension of the traditional analysis of variance (ANOVA) to the analysis of model output variance. Let us return to model $u = f(\boldsymbol{\theta})$ in (10.2.6) and keep the assumption that all input factors are independent random variables. The univariate *partial variance* of output u due to θ_i varying over its uncertainty range is denoted by V_i and given by

$$V_i = Var\left[E\left(u \middle| \theta_i\right)\right], i = 1, 2, \cdots, m. \tag{10.2.13}$$

For a nonlinear model, V_i is not the total effect of the uncertainty of θ_i to the output variance because of the combined effects of the uncertainty of θ_i with the uncertainties of other factors. This fact can be seen clearly from the *variance decomposition* theorem proposed by the Russian mathematician I. M. Sobol'. The theorem states that if all input variables are independent of each other, the *total variance* of model output V can be uniquely decomposed into the following finite series (Sobol' 1993):

$$V = Var\left[f(\boldsymbol{\theta})\right] = \sum_{i=1}^{m} V_i + \sum_{i=1}^{m}\sum_{j>i}^{m} V_{i,j} + \cdots + V_{1,2,\ldots m}. \tag{10.2.14}$$

The first term of this decomposition consists of all univariate partial variances due to the uncertainty of each factor, the second term consists of all bivariate partial variances due to the joint uncertainties of any two factors, and the final term is the partial variance due to the joint uncertainties of all factors. In the first term, V_i is given by (10.2.13), and in the second term, $V_{i,j}$ is given by

$$V_{i,j} = Var\left[\mathrm{E}\left(u \middle| \theta_i, \theta_j\right)\right] - V_i - V_j. \tag{10.2.15}$$

When $V_{i,j}$ is nonzero, the two factors θ_i and θ_j are said to be *interactive*. In this case, the output variance contains more contribution from the two factors than the sum of $V_i + V_j$ alone.

The decomposition (10.2.14) has 2^m terms, each of which represents a different conditional expectation of the model output and needs to be calculated by numerical integration. Therefore, using (10.2.14) to find the total variance of the model input is impractical.

10.2.2.2 Global Sensitivity Indices

As we define V_i in (10.2.13), the partial variance of the output contributed by the uncertainties of all factors except for θ_i (or non-θ_i) is denoted by V_{-i} and given by

$$V_{-i} = Var[E(u|\boldsymbol{\theta}_{-i})], \tag{10.2.16}$$

where $\boldsymbol{\theta}_{-i} = \{\theta_1, \theta_2, \cdots, \theta_{i-1}, \theta_{i+1}, \cdots, \theta_m\}$. The quantity $E[Var(u \mid \boldsymbol{\theta}_{-i})]$ represents another partial variance contributed by the uncertainties of all factors jointly with the uncertainty of θ_i (i.e., between θ_i and at least one component from $\boldsymbol{\theta}_{-i}$). The sum of the two quantities is the *total variance* of the output,

$$V = Var\left[E\left(u|\boldsymbol{\theta}_{-i}\right)\right] + E[Var(u \mid \boldsymbol{\theta}_{-i})]. \tag{10.2.17}$$

Note that $E[\mathrm{Var}(u \mid \boldsymbol{\theta}_{-i})]$ can also be explained as the total contribution of the uncertainty of θ_i to the variance of output, and (10.2.17) means that the total variance of output is equal to the sum of the total variance due to non-θ_i and the total variance due to θ_i.

The ratio between V_i and the total variance V is called the *first-order global sensitivity index* (*or first-order Sobol' index*) of θ_i and denoted by

$$S_i = \frac{V_i}{V}. \tag{10.2.18}$$

It measures the *main effect* of the uncertainty of θ_i to the output variance, while the total effect of the uncertainty of θ_i is measured by the *total-order sensitivity index*

$$S_i^T = \frac{E[Var(u \mid \boldsymbol{\theta}_{-i})]}{V} = \frac{V - Var[E(u \mid \boldsymbol{\theta}_{-i})]}{V} = 1 - \frac{V_{-i}}{V}. \tag{10.2.19}$$

The global sensitivity problem thus becomes the calculation of indices S_i and S_i^T for all $i = 1, 2, \cdots, m$. For a linear model, or more generally, a purely additive model, we have $S_i = S_i^T = S_{\sigma,i}^2$, and according to (10.2.12), $\sum S_i = \sum S_i^T \approx 1$. Otherwise, $S_i < S_i^T$, and $\sum S_i^T > 1$ because the interaction between θ_i and θ_j is counted in both S_i^T and S_j^T.

10.2.2.3 Algorithm

The purpose of GSA is to find E, V, V_i, V_{-i} and then use them to calculate S_i and S_i^T for all $i = 1, 2, \cdots, m$. Because E, V, V_i, V_{-i} are defined by integrals, the core part of GSA calculations is numerical integration. In the algorithm proposed by Saltelli (2002) and Saltelli et al. (2008), the Month Carlo method is used for numerical integration. The Saltelli algorithm consists of the following major steps:

1. Generate two sample matrices **A** and **B** and then generate a matrix $\mathbf{C}_i$ from matrix **B** such that its *i*th column is replaced by the *i*th column of A.

$$\mathbf{A} = \begin{bmatrix} \theta_{11} & \theta_{12} & \cdots & \theta_{1m} \\ \theta_{21} & \theta_{22} & \cdots & \theta_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \theta_{N1} & \theta_{N2} & \cdots & \theta_{Nm} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \theta'_{11} & \theta'_{12} & \cdots & \theta'_{1m} \\ \theta'_{21} & \theta'_{22} & \cdots & \theta'_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \theta'_{N1} & \theta'_{N2} & \cdots & \theta'_{Nm} \end{bmatrix} \tag{10.2.20}$$

$$\mathbf{C}_i = \begin{bmatrix} \theta'_{11} & \theta'_{12} & \cdots & \theta_{1i} & \cdots & \theta'_{1m} \\ \theta'_{21} & \theta'_{22} & \cdots & \theta_{2i} & \cdots & \theta'_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \theta'_{N1} & \theta'_{N2} & \cdots & \theta_{Ni} & \cdots & \theta'_{Nm} \end{bmatrix},$$

   where *N* is the number of samples. The Latin Hypercube Sampling method introduced in the last section is frequently used to generate these samples.

2. Use the samples in matrices **A**, **B**, and $\mathbf{C}_i$, respectively, as model inputs to run the model $u = f(\boldsymbol{\theta})$, the following model outputs are obtained

$$\mathbf{u}_A = \left\{u_{A,1}, u_{A,2}, \cdots, u_{A,N}\right\}, \ \mathbf{u}_B = \left\{u_{B,1}, u_{B,2}, \cdots, u_{B,N}\right\},$$
$$\mathbf{u}_{C_i} = \left\{u_{C_i,1}, u_{C_i,2}, \cdots, u_{C_i,N}\right\}$$

3. Calculate the sample statistics

$$\begin{aligned} E &= \frac{1}{N}\sum_{j=1}^{N} u_{A,j}, \quad V = \frac{1}{N}\sum_{j=1}^{N} u_{A,j}^2 - E^2, \\ V_i &= Var[E(u \mid \theta_i)] = \frac{1}{N}\sum_{j=1}^{N} u_{A,j}u_{C_i,j} - E^2, \\ V_{-i} &= Var[E(u \mid \boldsymbol{\theta}_{-i})] = \frac{1}{N}\sum_{j=1}^{N} u_{B,j}u_{C_i,j} - E^2 \end{aligned} \tag{10.2.21}$$

4. Use the above sample statistics to find $S_i = \dfrac{V_i}{V}$ and $S_i^T = 1 - \dfrac{V_{-i}}{V}$.

Compared to the original Sobol' (1993) algorithm, the above Saltelli (2002) algorithm is more effective; to obtain all sensitivity indices, the former needs $N(2m+1)$ model runs, while the latter needs only $N(m+1)$ model runs. Even so, GSA can still become computationally demanding as the dimensions of parameter space increase.

10.2.2.4 Multiple Model Outputs

In the above discussion on GSA, the model to be considered has only one output variable (i.e., $u = f(\boldsymbol{\theta})$). For EWR modeling, we have to deal with multiple output models, such as the numerical solution of a distributed parameter system given by

$$\mathbf{u} = \mathbf{f}(\boldsymbol{\theta}). \tag{10.2.22}$$

Let $\left\{u_1, u_2, \cdots, u_n\right\}$ be the values of state variables at particular locations and/or times of interest. The first- and total-order sensitivity indices associated with these outputs are

$$\left\{S_i(u_k),\ S_i^T(u_k)\right\}, k = 1,2,\cdots,n;\ i = 1,2,\cdots,m. \tag{10.2.23}$$

The total number of sensitivity indices required to be calculated is increased n times, but this does not mean that the total computational effort is increased n times too. With the above Saltelli algorithm, the number of samples used in step 1 is not changed because it depends only on model inputs; the number of model runs to obtain all corresponding model outputs in step 2 is also not changed because all components $\left\{u_1, u_2, \cdots, u_n\right\}$ are obtained simultaneously from the output of the model (10.2.22); and finally, only a little more algebraic operations are added for calculating the sample statistics of all output variables.

Example 10.2 *Using GSA for a Mass Transport Model*
Let us consider one-dimensional mass transport in a semi-infinite porous medium, for which the governing equation is

$$R\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2} - \frac{v}{\phi}\frac{\partial C}{\partial x}, \tag{10.2.24}$$

subject to the following boundary and initial conditions

$$\begin{aligned}
&\left.\left(vC - \frac{D}{\phi}\frac{\partial C}{\partial x}\right)\right|_{x=0} = vC_0,\ t > 0,\\
&\left.\frac{\partial C}{\partial x}\right|_{x=\infty} = 0,\ t > 0\\
&C = 0,\ t = 0
\end{aligned} \tag{10.2.25}$$

where R is a retardation factor [-] that is related to the reaction (e.g., adsorption); D is the dispersion coefficient [m^2/s], v is the Darcy flux [m/s], ϕ is porosity [-], and C is the solute concentration [mg/L]. Equation (10.2.24) has the same form as the water quality model given in Example 1.5. In the current case, the dispersion coefficient is expressed as

$$D = \alpha v + D_e,$$

where α is dispersivity [m] and D_e is the effective diffusion coefficient [m^2/s].

Our task at hand is to find the global sensitivity indices for all input parameters. Assume that we know the following information about model parameter distributions

$$\begin{aligned} &v \sim Tri(1\times10^{-5};1\times10^{-6},5\times10^{-5}),\\ &\phi \sim Tri(0.2;0.10,0.25)\\ &\alpha \sim Tri(5;1,10)\\ &D_e \sim Tri(2\times10^{-9};1.8\times10^{-9},2.2\times10^{-9})\\ &R \sim Tri(7;5,15), \end{aligned} \tag{10.2.26}$$

in which $Tri(c;a,b)$ represents a triangular PDF with mode c (i.e., the peak of triangle), lower-bound a, and upper-bound b. Triangular distributions are often used when there is insufficient information to characterize parameter variability.

The analytical solution to model (10.2.24) is given in van Genuchten, Alves (1982),

$$\begin{aligned} \frac{C}{C_0} &= \frac{1}{2}\operatorname{erfc}\left(\frac{Rx-ut}{2\sqrt{DRt}}\right) + \sqrt{\frac{u^2t}{\pi DR}}\exp\left(-\frac{(Rx-ut)^2}{4DRt}\right)\\ &-\frac{1}{2}\left(1+\frac{ux}{D}+\frac{u^2t}{DR}\right)\exp\left(\frac{ux}{D}\right)\operatorname{erfc}\left(\frac{Rx+ut}{2\sqrt{DRt}}\right). \end{aligned} \tag{10.2.27}$$

in which $u = v/\phi$ is the flow velocity.

Figure 10.4 shows the concentration distribution at $t = 10$ days for various parameter combinations. The thick solid line is simulated using the parameter modes (i.e., the first number in $Tri(c;a,b)$) listed in (10.2.26), whereas the 1000 hairlines are obtained by sampling from the triangular distributions using the LHS. The figure indicates that concentration distributions vary significantly for the given parameter ranges.

To perform GSA using the Saltelli algorithm described in the text, we generate two sets of samples, $\mathbf{A}$ and $\mathbf{B}$, according to (10.2.20). The number of LHS samples used in this example is 2^{13}. Figure 10.5a and b show the global sensitivity indices calculated for three locations, $x = 0.1$, 10, and 20 m, respectively. In all cases, the uncertainty in v contributes the most to the total variability. Dispersivity plays a

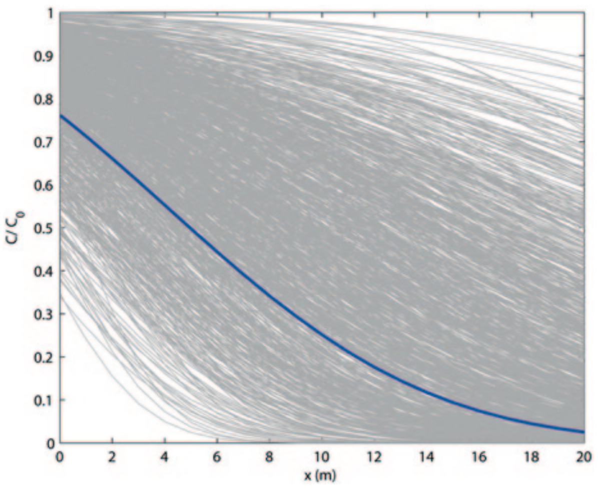


Fig. 10.4 Concentration distribution as a function of distance from source for $t = 10$ days. A *thick line* is generated using modes of uncertain parameters, while *hairlines* are generated using LHS

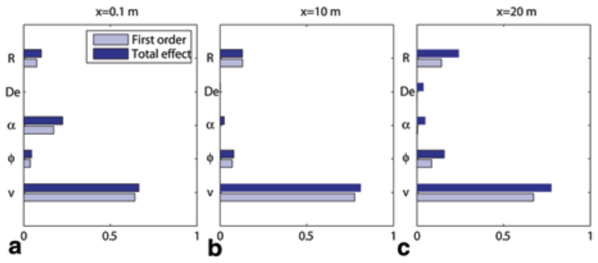


Fig. 10.5 First-order and total-order indices calculated for **a** $x = 0.1$ m, **b** $x = 10$ m, and **c** $x = 20$ m. Time is fixed at $t = 10$ days

more significant role at $x = 0.1$ and then its contribution diminishes at larger distances. This example shows that the global sensitivity indices may depend on both space and time. ■

GSA has become a widely used tool in EWR applications in recent years. Tang et al. (2007) used the Saltelli method (Saltelli 2002) to identify influential hydrological parameters in a distributed watershed model. Liu et al. (2012) applied GSA to a regional groundwater model to identify zonal hydraulic conductivity values that significantly affect the model output. In a geologic carbon sequestration risk assessment study, Wainwright et al. (2013) used GSA to identify parameters that affect the extent of the CO_2 plume and the overpressured zone the most.

A straightforward and less "intrusive" approach for expediting the GSA is to adopt distributed or parallel computing. For example, Liu et al. (2012) used cloud computing to run a regional-scale groundwater model. Metamodeling, a generic term that refers to all techniques for reducing model complexity or for constructing proxy models, may also be used to improve computational efficiency (Fang et al. 2006; Sudret 2008).

10.2.3 Screening Input Factors

For large systems involving many input variables, preliminary analyses need to be performed to determine which variables to include in the UQ. Such an analysis is part of the deliberation process mentioned at the beginning of this chapter. A general purpose of screening is to identify all relevant uncertain variables, but only select those that either significantly affect the model outputs or are risk significant, or both.

Screening of input parameters should avoid two common types of errors; either important parameters are omitted or noninfluential parameters are retained. Model-input screening and model-output uncertainty quantification are thus two complementary tasks. The former can make the latter more effective and, in turn, the latter can make the former more exact. The benefits of eliminating noninfluential factors from further studies are (i) the data collection will be more targeted, (ii) the inverse solution will be easier because of the dimension reduction, and (iii) the model will become more reliable. In Chap. 8, we have learned how to screen the inputs of a data-driven model. This subsection will show how GSA can be used for this purpose.

10.2.3.1 Variance-Based Measure

GSA provides a quantitative basis for input screening, in which, the effects of all input factors are simultaneously considered, the entire region of uncertainties is thoroughly explored, and the effects of interactions are completely accounted. After the global sensitivity indices S_i and S_i^T are calculated for all factors, $i = 1, 2, \cdots, m$, we can use them for input screening. Note that a large first-order index S_i indicates that θ_i is an important factor, but a small S_i does not necessarily mean the factor is noninfluential because its interactions with other factors may be significant. That is why we still need to find the total effect of a factor θ_i, which is represented by the index S_i^T. According to the values of the total effect of all factors, we can select part of them that account for, say, more than 90 % of the total variance of the output while keeping the remaining factors constant. Returning to Example 10.2, the results of GSA suggest that if we are mainly concerned with far-field transport, we would keep only the Darcy flux, porosity, and the retardation factor as important parameters for further studies. Once the uncertainties of these parameters are reduced, the accuracy of the model output will be significantly increased.

As mentioned before, the main practical challenge of applying variance-based GSA is that the computational effort of calculating all global sensitivity indices may become unaffordable when the cost of running the model is high and/or the dimensions of the parameter space are high.

10.2.3.2 Morris Sampling-Based Measure

Morris sampling (Morris 1991) is an alternative method for GSA. It can give a rough estimate of global sensitivities with less computational effort. Thus, it is particularly useful for screening a large number of input factors of a complex model. The basic idea underneath Morris sampling is rather straightforward; the global sensitivity of a factor is estimated by averaging its local sensitivities, which are approximated by finite difference. Let us consider again the uncertainty of the model $u = f(\theta_1, \theta_2, \cdots, \theta_m)$ resulting from the uncertain input factors, and assume that the range forms a hypercube in the m-dimensional parameter space. Partitioning the hypercube by a finite difference grid, the local sensitivity of a factor θ_i at a point $\boldsymbol{\theta}$ can be approximated by

$$D_i = \frac{f(\theta_1, \theta_2, \ldots, \theta_{i-1}, \theta_i + \Delta_i, \theta_{i+1}, \ldots, \theta_m) - f(\boldsymbol{\theta})}{\Delta_i}, \tag{10.2.28}$$

where Δ_i is the step size along the θ_i direction. In the literature, D_i is also called the *elementary effect*. Usually, the step size is set as

$$\Delta_i = p \,/\, (2(p-1)) \times \text{range of} \theta_i,$$

where p is an even number. The method randomly chooses r nodes of the grid as starting points of r trajectories. Each trajectory is generated in the following manner: from the starting point, move one step along the θ_1 direction to arrive at the first point; then from the first point, move one step along the θ_2 direction to arrive at the second point; and so forth, until starting from the ($m-1$)-th point, move one step along the θ_m direction to arrive at the end of the trajectory. Let the nodes on the jth trajectory be

$$\boldsymbol{\theta}_{j,0}, \boldsymbol{\theta}_{j,1}, \cdots, \boldsymbol{\theta}_{j,i}, \cdots, \boldsymbol{\theta}_{j,m}; j = 1, 2, \cdots, r. \tag{10.2.29}$$

For each point $\boldsymbol{\theta}_{j,i}$ on the trajectory, we can use (10.2.28) to calculate an elementary effect $D_{j,i}$. Because the total number of points on all r trajectories is $r(m+1)$, the same number of model runs is needed to obtain all elementary effects. For each factor θ_i, we have r samples $\left\{D_{1,i}, D_{2,i}, \cdots, D_{r,i}\right\}$. Then, we can calculate the following sample mean and sample variance with respect to each factor:

$$\mu_i = \frac{1}{r}\sum_{j=1}^{r} D_{j,i}, \quad \sigma_i^2 = \frac{1}{r}\sum_{j=1}^{r}[D_{j,i} - \mu_i]^2, \quad i = 1, 2, \cdots, m, \tag{10.2.30}$$

where μ_i is an estimate of the global sensitivity of the model output with respect to factor θ_i, and σ_i^2 measures the uncertainty of the estimation. To assure the global requirement of the estimations, the trajectories should be as far apart as possible. After obtaining the statistics in (10.2.30), we can rank the effects of all input factors. A factor with small μ_i and small σ_i^2 can be considered a noninfluential one.

Note that for a nonmonotonic function, $D_{j,i}$'s may have different signs and a small μ_i may be obtained after cancellation. In this case, μ_i is often estimated by $\mu_i^* = \frac{1}{r}\sum_{j=1}^{r}\left|D_{j,i}\right|$ in order to avoid labeling an important factor as a noninfluential one.

Saltelli et al. (2008) provided a review of several other sensitivity analysis methods. In general, the more quantitative a screening method becomes, the more information it requires about characteristics of the unknown parameters.

10.2.3.3 Screening Inputs for Model Application

Obtaining model outputs is generally not the ultimate goal of constructing a model. As shown in Fig. 10.1, the model output is either used as an input to a given model application or used to calculate the fitting residual for inverse solution. Note that a factor that is sensitive to model outputs may be insensitive to a model application. In practice, we care more about the uncertainty of model applications than that of model outputs. The value-of-information principle seeks to address whether reduction of uncertainty in parameters would make a meaningful difference in the decision (Stern et al. 1996; Bratvold et al. 2009).

GSA can be applied to composite modules in Fig. 10.1. For example, to combine the forward solution module $\mathbf{u} = \mathbf{f}(\boldsymbol{\theta})$ with a model application module $g = g(\mathbf{u})$, a composite module $\boldsymbol{\theta} \to h(\boldsymbol{\theta})$ is formed, where $h(\boldsymbol{\theta}) = g[\mathbf{f}(\boldsymbol{\theta})]$. After the global sensitivity indices of $h(\boldsymbol{\theta})$ with respect to all input factors are calculated and the most influential factors to the given model application are identified, we only need to concentrate on how to decrease the uncertainties of those factors (e.g., by collecting more data in the objective-oriented sense). Obviously, this is one of the most important topics of successful modeling and in-depth discussion will be provided in the next two chapters.

10.3 Stochastic Methods for Uncertainty Propagation

Conducting a comprehensive uncertainty analysis using the standard Monte Carlo and LHS techniques may not be feasible for a complex model. Because of the high computational cost of running the forward model, the number of samples is often not large enough to reach convergence, which means the results of uncertainty analysis can be unreliable. The same difficulty can also be seen in the use of global sensitivity methods, where the number of samples required for calculating all global sensitivity indices increases significantly when the accuracy requirement of UQ is increased. The purpose of this section is to introduce several computationally efficient SRSMs for uncertainty propagation. In these methods, both model inputs and outputs are considered as stochastic variables and approximated either by their truncated expansions or by their interpolants. The model uncertainty is then estimated with a smaller number of actual model runs.

10.3.1 Stochastic Response Surface Methods

A stochastic model contains random variables in its inputs and/or structure. As a result, the model output also becomes stochastic. Let us consider the general distributed parameter model used in the EWR fields (Eq. (1.1.14) in Chap. 1). When uncertainty is involved in its physical parameters $\mathbf{p}$, control variables $\mathbf{q}$, and initial/boundary conditions $\mathbf{b}$, the model can be represented by

$$\mathcal{L}[\mathbf{u}, \mathbf{p}(\boldsymbol{\xi}), \mathbf{q}(\boldsymbol{\xi}), \mathbf{b}(\boldsymbol{\xi})] = \mathbf{0}, \tag{10.3.1}$$

where $\boldsymbol{\xi}$ is a set of uncorrelated random variables called "germs." As mentioned in Sect. 10.1, there are two types of randomness, aleatory and epistemic. We will use $\boldsymbol{\theta}(\boldsymbol{\xi})$ to represent all variables in $\{\mathbf{p}, \mathbf{q}, \mathbf{b}\}$ that contain any type of randomness. Equation (10.3.1) can be rewritten as $\mathcal{L}[\mathbf{u}, \boldsymbol{\theta}(\boldsymbol{\xi})] = \mathbf{0}$ and its forward solution is given by

$$\mathbf{u}(\boldsymbol{\xi}) = \mathcal{M}[\boldsymbol{\theta}(\boldsymbol{\xi})], \tag{10.3.2}$$

where $\boldsymbol{\theta}(\boldsymbol{\xi})$ and $\mathbf{u}(\boldsymbol{\xi})$ are inputs and outputs, respectively, and all of them may vary with space and time (i.e., they are stochastic fields). A germ ξ can be a uniform distribution, a standard normal distribution, or others. Let us consider the following cases:

- A stochastic scalar parameter θ is expressed as a function $\theta(\xi)$ of a single ξ.
- A stochastic parameter vector is expressed by several germs. The number of germs may not be the same as the dimensions of the parameter vector.
- A distributed stochastic parameter $\theta(\mathbf{x})$ is parameterized by a set of germs. In Chap. 6, a stochastic field $\theta(\mathbf{x})$ is approximated by truncating its Karhunen-Loève (KL) expansion and represented by a low-dimensional random vector. According to Eq. (6.3.21), we have

$$\theta(\mathbf{x}) \approx \hat{\theta}(\xi_1, \xi_2, \cdots, \xi_k), \tag{10.3.3}$$

  where $k \ll N$ and N is the number of nodes used in the numerical solution.

The total number of germs involved in all inputs of a model (i.e., the dimension of $\boldsymbol{\xi}$) is called the *random dimension* of the model.

Response surface methods (RSMs) refer broadly to all mathematical and statistical techniques used for building surrogate models or *metamodels*. Like interpolation and parameterization that represent a function with a small number of measurements, an RSM attempts to construct the model output space using a small number of model runs. More precisely, if the original model is $\mathbf{u} = \mathcal{M}(\mathbf{x}, \boldsymbol{\theta})$, then we want to find a metamodel $\hat{\mathcal{M}}$ that converges to $\mathcal{M}$ in the mean square sense and is computationally cheaper. In a broad sense, examples of the RSM that we have already seen in previous chapters include:

- Using a parameterized model to replace a distributed parameter for inversion
- Using PCA or FA to reduce the dimensions of a dataset (Chap. 6)
- Using a variogram model to describe the structure of a random field (Chap. 6)
- Applying K-L decomposition to parameterize a random field (Chap. 6)
- Using a shrinkage method to reduce the complexity of a linear model (Chap. 8)
- Training ANN to approximate a physically based model (Chap. 8)
- Using GPR to generate a data-driven model (Chap. 8)

In this section, we introduce stochastic RSMs (SRSMs), an extension of RSMs, to develop metamodels for stochastic modeling. Uncertainty propagation can then be conducted by using metamodels, instead of full model runs. As a consequence, sample-based methods for UQ will become more efficient because a large number of samples can be analyzed when $\mathcal{M}$ is replaced by $\hat{\mathcal{M}}$. For the same token, SRSMs can also significantly improve the efficiency of sensitivity analysis (Buzzard 2012; Sudret 2008).

Several SRSMs that have received broad attention in engineering reliability analyses are the polynomial chaos expansion (PCE) method and the stochastic collocation (SCM) method. In the former, truncated orthogonal polynomial expansions in the stochastic space are used as metamodels, while in the latter polynomial interpolants in the stochastic space are used as metamodels. With polynomial approximation, exponential convergence rates can be achieved for a wide range of probabilistic analysis problems (Xiu and Karniadakis 2002). Let us first introduce the basic theory and methods of PCE and its applications to UQ.

10.3.2 Polynomial Chaos

The original idea of using orthogonal polynomial expansion for uncertainty analysis was established by Wiener in his homogeneous chaos theory (Wiener 1938) and later rediscovered by Ghanem and Spanos (1991). PCE can be used to construct a metamodel that maintains high-order effects of a nonlinear stochastic model. Let us start from a simple model that contains only a single random variable.

10.3.2.1 One Random Dimensional PCE

Orthogonal Polynomial Expansion in Probability Space In order to identify an infinite-dimensional deterministic function $f(x)$, it must be parameterized or approximated by a low-dimensional function. A general parameterization method that we have used many times in the previous chapters is to truncate an infinite-dimensional expansion of the function, namely, let

$$f(x) = \sum_{i=0}^{\infty} f_i \phi_i(x) \approx \sum_{i=0}^{n} f_i \phi_i(x), \tag{10.3.4}$$

where $\{\phi_i(x)\}$ is a selected family of basis functions (see Appendix A). The problem of identifying $f(x)$ then becomes identification of a vector $\mathbf{f} = \{f_1, f_2, \cdots, f_n\}$. Of course, we prefer to select such basis functions that make the series in (10.3.4) converge faster. In PCE, the same structure as (10.3.4) is used to parameterize a random function $f(\xi)$ in the stochastic space and orthogonal polynomials are used as basis functions, viz.

$$f(\xi) = \sum_{i=0}^{\infty} f_i \phi_i(\xi) \approx \sum_{i=0}^{n} f_i \phi_i(\xi), \tag{10.3.5}$$

where $\phi_i(\xi)$ is a polynomial of degree i. A family of polynomials $\{\phi_i(\xi)\}$ is said to be orthogonal and optimal with respect to a probability distribution $p(\xi)$ if

$$\int_{(\Omega)} \phi_i(\xi)\phi_j(\xi)p(\xi)d\xi = c_j \delta_{ij}, \tag{10.3.6}$$

where δ_{ij} is the Kronecker delta function, and c_j is given by

$$c_j = \int_{(\Omega)} [\phi_j(\xi)]^2 p(\xi)d\xi. \tag{10.3.7}$$

In PCE, the truncated expansion (10.3.5) is not used for inversion; instead, it is regarded as a reduced-order model (or metamodel) of $f(\xi)$,

$$f(\xi) \approx \hat{f}(\xi) = \sum_{i=0}^{n} f_i \phi_i(\xi). \tag{10.3.8}$$

Multiplying this equation by $\phi_j(\xi)p(\xi)$ for $j = 1, 2, \cdots, n$, integrating over (Ω), and then using (10.3.6) and (10.3.7), we can find all coefficients

$$f_i = (1/c_i)\int_{(\Omega)} f(\xi)\phi_i(\xi)p(\xi)d\xi, \quad (i = 1, 2, \cdots, n) \tag{10.3.9}$$

Thus, after $p(\xi)$ is given, its optimal orthogonal polynomial family $\{\phi_i(\xi)\}$ is chosen, and the order *n* of approximation is specified, the metamodel $\hat{f}(\xi)$ of a function $f(\xi)$ defined in (10.3.8) is completely determined. We will have $\hat{f}(\xi) \to f(\xi)$ when $n \to \infty$, with an exponential convergence rate in general. Below are several most often used combinations of $p(\xi)$ and $\{\phi_i(\xi)\}$:

- Gaussian distribution & Hermite polynomials (*H*)

  In this case, $\Omega = (-\infty, +\infty)$, $p(\xi) \sim \frac{1}{\sqrt{2\pi}} e^{-\xi^2/2}$, the Hermite polynomial defined by $H_i(\xi) = (-1)^i e^{\xi^2/2} \frac{d^i}{d\xi^i}[e^{-\xi^2/2}]$ is used as $\phi_i(\xi)$, and $c_j = j!$. The first five polynomials are:

$$H_0(\xi) = 1, H_1(\xi) = \xi, H_2(\xi) = \xi^2 - 1,$$

$$H_3(\xi) = \xi^3 - 3\xi, \text{ and } H_4(\xi) = \xi^4 - 6\xi^2 + 3.$$

In general, we have the recurrence relationship

$$H_{i+1}(\xi) = \xi H_i(\xi) - iH_{i-1}(\xi).$$

- Uniform distribution & Legendre polynomials (P)

  In this case, $\Omega = [-1, +1]$, $p(\xi) \sim 1$, $P_i(\xi) = \frac{1}{2^i i!}\frac{d^i}{d\xi^i}[(\xi^2 - 1)^i]$ is used as $\phi_i(\xi)$, and $c_j = \frac{2}{2i+1}$. The first five polynomials are:

$$P_0(\xi) = 1, P_1(\xi) = \xi, \ P_2(\xi) = \frac{1}{2}(3\xi^2 - 1),$$

$$P_3(\xi) = \frac{1}{2}(5\xi^3 - 3\xi), \text{ and } P_4(\xi) = \frac{1}{8}(35\xi^4 - 30\xi^2 + 3).$$

  In general, a recurrence formula is

$$P_{i+1}(\xi) = \frac{2i+1}{i+1}\xi P_i(\xi) - \frac{i}{i+1}P_{i-1}(\xi).$$

- Exponential distribution & Laguerre polynomials (L)

  In this case, $\Omega = [0, +\infty)$, $p(\xi) \sim e^{-\xi}$, $\phi_i(\xi) \equiv L_i(\xi) = \frac{e^{\xi}}{i!}\frac{d^i}{d\xi^i}[e^{-\xi}\xi^i]$, and $c_j = 1$. The first five polynomials are:

$$L_0(\xi) = 1, \ L_1(\xi) = -\xi + 1, \ L_2(\xi) = \frac{1}{2}(\xi^2 - 4\xi + 2),$$
$$L_3(\xi) = \frac{1}{6}(-\xi^3 + 9\xi^2 - 18\xi + 6), \text{ and}$$
$$L_4(\xi) = \frac{1}{24}(\xi^4 - 16\xi^3 + 72\xi^2 - 96\xi + 24).$$

  In general, a recurrence formula is

$$L_{i+1}(\xi) = \frac{1}{i+1}((2i + 1 - \xi)L_i(\xi) - iL_{i-1}(\xi)).$$

More combinations of probability distributions and their matching orthogonal polynomials can be found, for example, in Xiu (2010). Using a nonoptimal combination of $p(\xi)$ and $\{\phi_i(\xi)\}$ (e.g., using Hermite polynomials with uniform distribution) will cause the convergence rate of $\hat{f}(\xi) \to f(\xi)$ to decrease.

The PCE of a Model Input Parameter Let $\theta(\xi)$ be a random input parameter that depends on a germ ξ with a known distribution $p(\xi)$. According to $p(\xi)$, we can find a family of optimal orthogonal polynomials, e.g., using Hermite polynomials when $p(\xi)$ is Gaussian. Applying (10.3.8) to $\theta(\xi)$, we have

$$\theta(\xi) \approx \hat{\theta}(\xi) = \theta_0\phi_0(\xi) + \theta_1\phi_1(\xi) + \cdots + \theta_i\phi_i(\xi) + \cdots + \theta_n\phi_n(\xi). \quad (10.3.10)$$

Using (10.3.9), we can find all coefficients of the expansion,

$$\theta_i = (1/c_i)\int_{(\Omega)} \theta(\xi)\phi_i(\xi)p(\xi)d\xi, (i = 0,1,\cdots,n). \quad (10.3.11)$$

These integrals may be calculated numerically by the Gaussian quadrature,

$$\theta_i = (1/c_i)\sum_k \theta(\xi_k)\phi_i(\xi_k)w_k, (i = 0,1,\cdots,n), \quad (10.3.12)$$

where ξ_k and w_k are Gaussian quadrature points and weights, respectively, and $\theta(\xi_k)$ are obtained by function evaluation (we will give more explanation on the Gaussian quadrature later). After all coefficients are calculated, (10.3.10) gives a metamodel $\hat{\theta}(\xi)$ of $\theta(\xi)$.

Uncertainty Propagation Through a Scalar Model Let us consider a model $u(\xi) = \mathcal{M}[\theta(\xi)]$. Our purpose is to find the uncertainty of $u(\xi)$. First, according to the known distribution $p(\xi)$ of ξ, we can find its PCE

$$u(\xi) \approx \hat{u}(\xi) = u_0\phi_0(\xi) + u_1\phi_1(\xi) + \cdots + u_i\phi_i(\xi) + \cdots + u_n\phi_n(\xi). \quad (10.3.13)$$

Using (10.3.9), all coefficients of the expansion are determined by

$$u_i = (1/c_i)\int_{(\Omega)} u(\xi)\phi_i(\xi)p(\xi)d\xi, \ (i = 0,1,\cdots,n) \quad (10.3.14)$$

These integrals can be calculated numerically using the Gaussian quadrature,

$$u_i = (1/c_i)\sum_k u(\xi_k)\phi_i(\xi_k)w_k, (i = 0,1,\cdots,n), \quad (10.3.15)$$

where ξ_k and w_k are quadrature points and weights, respectively, and $u(\xi_k)$ is obtained by running the model $\mathcal{M}[\theta(\xi_k)]$.

Substituting all coefficients in (10.3.15), a metamodel $\hat{u} = \hat{\mathcal{M}}(\hat{\theta})$ in (10.3.13) is obtained, and $\hat{\mathcal{M}} \to \mathcal{M}$ when $n \to \infty$. Using this metamodel, it is easy to estimate the model output uncertainty that is propagated from the uncertain model inputs. Taking expectation of (10.3.13) and noting $E[\phi_0] = 1$ and $E[\phi_i] = 0$ for $i = 1,2,\cdots,n$, we obtain the following statistics of the model output

$$\begin{aligned} &E[u] = u_0, \\ &Var[u] = E[(u - u_0)^2] \approx \sum_{i=1}^{n} c_i(u_i)^2. \end{aligned} \quad (10.3.16)$$

Higher-order moments can also be obtained from $\hat{\mathcal{M}}$ when needed.

Example 10.3 *Convection Model Uncertainty Caused by Uncertain Initial Conditions*
The one-dimensional convection transport model is given by

$$\begin{aligned}&\frac{\partial u}{\partial t}+V\frac{\partial u}{\partial x}=0, 0\le x\le 1, \text{subject to}\\&u(x,t,\xi)\mid_{t=0}=g(x,\xi),\\&u(x,t)\mid_{x=0}=f_0(t), u(x,t)\mid_{x=1}=f_1(t).\end{aligned}\tag{10.3.17}$$

Because the initial conditions are random functions of ξ , the solution will also be a random function of ξ and can be represented by the following expansion

$$u(x,t,\xi)\approx\hat{u}(x,t,\xi)=\sum_{i=0}^{n}u_i(x,t)\phi_i(\xi).\tag{10.3.18}$$

Substituting this expression into the PDEs (10.3.17) yields

$$\sum_{i=0}^{n}\frac{\partial u_i}{\partial t}\phi_i(\xi)+V\left(\sum_{i=0}^{n}\frac{\partial u_i}{\partial t}\phi_i(\xi)\right)=0.\tag{10.3.19}$$

Multiplying by $p(\xi)\phi_j(\xi)$ and integrating over (Ω) in the stochastic space, the above equation generates a set of $n+1$ equations

$$\sum_{i=0}^{n}\frac{\partial u_i}{\partial t}\int_{(\Omega)}\phi_i\phi_j p(\xi)d\xi+V\left(\sum_{i=0}^{n}\frac{\partial u_i}{\partial t}\int_{(\Omega)}\phi_i\phi_j p(\xi)d\xi\right)=0,\quad (j=0,1,\cdots,n)\tag{10.3.20}$$

Using (10.3.6) and (10.3.7), the above equations can be rewritten as $n+1$ uncoupled deterministic convection equations

$$\begin{aligned}&\frac{\partial u_0}{\partial t}+V\frac{\partial u_0}{\partial x}=0\\&\frac{\partial u_1}{\partial t}+V\frac{\partial u_1}{\partial x}=0\\&\vdots\\&\frac{\partial u_n}{\partial t}+V\frac{\partial u_n}{\partial x}=0.\end{aligned}\tag{10.3.21}$$

Initial conditions of these PDEs can be found from the expansion of $g(x,\xi)$:

$$g(x,\xi) \approx \hat{g}(x,\xi) = g_0(x)\phi_0(\xi) + g_1(x)\phi_1(\xi) + \cdots + g_n(x)\phi_n(\xi). \qquad (10.3.22)$$

After the coefficients in this expansion are determined by PCE, we can use $u_i(x,t)\,|_{t=0} = g_i(x)(i = 0,1,\cdots,n)$ as the deterministic initial condition and the original boundary conditions in (10.3.17) to solve each PDE in (10.3.21). Finally, using these solutions as the coefficients $u_i(x,t)$ in (10.3.18), a metamodel of the original model (10.3.17) is found and UQ can be completed by (10.3.16) using the metamodel.

Uncertainty Propagation Through a Multioutput Model Extending the above discussion to the multiple output case is straightforward. Let us consider a vector model $\mathbf{u}(\xi) = \mathcal{M}[\theta(\xi)]$. Our purpose is to find the uncertainties of all its L outputs $\left\{u_j(\xi) \mid j = 1,2,\cdots,L\right\}$. The PCE expansion (10.3.8) for each component of $\mathbf{u}(\xi)$ is

$$u_j(\xi) \approx \hat{u}_j(\xi) = u_{0j}\phi_0(\xi) + u_{1j}\phi_1(\xi) + \cdots + u_{ij}\phi_i(\xi) + \cdots + u_{nj}\phi_n(\xi), \quad (j = 1,2,\cdots,L) \qquad (10.3.23)$$

According to (10.3.9), all coefficients of the expansion are determined by

$$u_{ij} = (1/c_i)\int_{(\Omega)} u_j(\xi)\phi_i(\xi)p(\xi)d\xi,\ (i = 0,1,\cdots,n; j = 1,2,\cdots,L) \qquad (10.3.24)$$

These integrals may be calculated numerically by the Gaussian quadrature,

$$u_{ij} = (1/c_i)\sum_k u_j(\xi_k)\phi_i(\xi_k)w_k, (i = 0,1,\cdots,n; j = 1,2,\cdots,L), \qquad (10.3.25)$$

where $u_j(\xi_k)$ for all $j = 1,2,\cdots,L$ are obtained simultaneously by running model $\mathcal{M}[\theta(\xi_k)]$. Substituting (10.3.25) into (10.3.23), a metamodel $\hat{\mathcal{M}}$ is obtained, and $\hat{\mathcal{M}} \to \mathcal{M}$ when $n \to \infty$. Using this metamodel, it is easy to estimate the uncertainty propagated from model inputs to model outputs. Taking the expectation of (10.3.23) and noting that $E[\phi_0] = 1$ and $E[\phi_i] = 0$ for $i = 1,2,\cdots,n$, we obtain the following statistics for all components of model outputs

$$\begin{aligned} &E[u_j] = u_{0j} \\ &Var[u_j] = E[(u_j - u_{0j})^2] = \sum_{i=1}^{n} c_i (u_{ij})^2 \\ &Cov[u_{j_1}, u_{j_2}] = E[(u_{j_1} - u_{0j_1})(u_{j_2} - u_{0j_2})] = \sum_{i=1}^{n} c_i u_{ij_1} u_{ij_2}. \end{aligned} \qquad (10.3.26)$$

Like in the case of GSA, when the number of model outputs increases, the complexity of PCE is not increased and the computational effort of using PCE for uncertainty analysis is increased only slightly. But, when the number of random dimensions increases, both the complexity of PCE and the computational effort of using PCE will be increased quickly.

10.3.2.2 Multiple Random Dimensional PCE

Hermite Polynomials for Multivariate Gaussian Distribution Let us consider the case that two independent Gaussian germs ξ_1 and ξ_2 are involved in a model. Their joint probability distribution is given by

$$p(\xi_1,\xi_2) = p(\xi_1)p(\xi_2) = \frac{1}{2\pi}\exp\left(-\frac{1}{2}\xi_1^2 - \frac{1}{2}\xi_2^2\right). \tag{10.3.27}$$

A two-dimensional polynomial, according to the order, has the following general forms:

$$\begin{aligned}
&P_0(\xi_1,\xi_2) = a_0^{(0)}\\
&P_1(\xi_1,\xi_2) = a_0^{(1)} + a_1^{(1)}\xi_1 + a_2^{(1)}\xi_2\\
&P_2(\xi_1,\xi_2) = a_0^{(2)} + a_1^{(2)}\xi_1 + a_2^{(2)}\xi_2 + a_3^{(2)}\xi_1^2 + a_4^{(2)}\xi_1\xi_2 + a_5^{(2)}\xi_2^2\\
&\cdots\ \cdots
\end{aligned}$$

The two-dimensional Hermite polynomials are defined specially by

$$\begin{aligned}
&\text{Zeroth-order:}\\
&\qquad H_{0,0}(\xi_1,\xi_2) = H_0(\xi_1)H_0(\xi_2) = 1\\
&\text{First-order:}\\
&\qquad H_{1,0}(\xi_1,\xi_2) = H_1(\xi_1)H_0(\xi_2) = \xi_1;\\
&\qquad H_{0,1}(\xi_1,\xi_2) = H_0(\xi_1)H_1(\xi_2) = \xi_2;\\
&\text{Second-order:}\\
&\qquad H_{2,0}(\xi_1,\xi_2) = H_2(\xi_1)H_0(\xi_2) = \xi_1^2 - 1;\\
&\qquad H_{1,1}(\xi_1,\xi_2) = H_1(\xi_1)H_1(\xi_2) = \xi_1\xi_2;\\
&\qquad H_{0,2}(\xi_1,\xi_2) = H_0(\xi_1)H_2(\xi_2) = \xi_2^2 - 1;\\
&\qquad \cdots\ \cdots\ \cdots
\end{aligned} \tag{10.3.28}$$

where $H_i(\cdot)$ is the *i*th order Hermite polynomial in one-dimensional defined before. From this equation, we can deduce that all *r*-th order two-dimensional Hermite polynomials, for any $r = 0, 1, 2, \cdots$, are defined by one-dimensional Hermite polynomials according to

$$H_{i_1,i_2}(\xi_1,\xi_2) = H_{i_1}(\xi_1)H_{i_2}(\xi_2), \text{where } i_1 + i_2 = r. \tag{10.3.29}$$

We can arrange all Hermite polynomials from low to high orders, viz.

$$H_{0,0}, H_{1,0}, H_{0,1}, H_{2,0}, H_{1,1}, H_{0,2}, H_{3,0}, H_{2,1}, H_{1,2}, H_{0,3}, \cdots\cdots \tag{10.3.30}$$

Orthogonality of two-dimensional Hermite polynomials is easy to show. In fact, when (i_1, i_2) is not equal to (j_1, j_2), we have

$$\begin{aligned} &\iint H_{i_1,i_2}(\xi_1,\xi_2)H_{j_1,j_2}(\xi_1,\xi_2)p(\xi_1,\xi_2)d\xi_1 d\xi_2 \\ &= \int H_{i_1}(\xi_1)H_{j_1}(\xi_1)p(\xi_1)d\xi_1 \int H_{i_2}(\xi_2)H_{j_2}(\xi_2)p(\xi_2)d\xi_2 = 0 \end{aligned} \tag{10.3.31}$$

When $(i_1, i_2) = (j_1, j_2)$, we have

$$\begin{aligned} &\iint H_{j_1,j_2}(\xi_1,\xi_2)H_{j_1,j_2}(\xi_1,\xi_2)p(\xi_1,\xi_2)d\xi_1 d\xi_2 \\ &= \int \left[H_{j_1}(\xi_1)\right]^2 p(\xi_1)d\xi_1 \int \left[H_{j_2}(\xi_2)\right]^2 p(\xi_2)d\xi_2 \\ &= c_{j_1}c_{j_2} = j_1!\, j_2! \end{aligned} \tag{10.3.32}$$

For notational convenience, we change the subscripts of (10.3.30) to a series form

$$\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4, \mathcal{H}_5, \mathcal{H}_6, \mathcal{H}_7, \mathcal{H}_8, \mathcal{H}_9, \cdots\cdots \tag{10.3.33}$$

Keep in mind that all of them are two-dimensional polynomials in the current case. The Hermite orthogonal expansion in two-dimensional random space has the same form as in one-dimensional, viz.

$$f(\xi_1,\xi_2) \approx \hat{f}(\xi_1,\xi_2) = \sum_{i=0}^{N} f_i \mathcal{H}_i(\xi_1,\xi_2) \tag{10.3.34}$$

To reach the same order of approximation, two-dimensional expansion needs more terms than one-dimensional expansion does. For example, the total number of terms up to the third-order in (10.3.34) is $N = 10$.

Extending the above derivations to cases involving more random dimensions is straightforward. Let $\boldsymbol{\xi} = \left\{\xi_1, \xi_2, \cdots, \xi_m\right\}$ be *m* independent Gaussian germs. Similar to (10.3.34), the Hermite orthogonal expansion of $f(\boldsymbol{\xi})$ is given by

$$f(\boldsymbol{\xi}) \approx \hat{f}(\boldsymbol{\xi}) = \sum_{i=0}^{N} f_i \mathcal{H}_i(\boldsymbol{\xi}). \tag{10.3.35}$$

In this expansion, if $\mathcal{H}_i(\boldsymbol{\xi})$ is an *r*-th order Hermite polynomial, for $r = 0, 1, 2, \cdots$, then as (10.3.29) in the two-dimensional case, it must have a form $H_{i_1, i_2, \cdots, i_m}(\xi_1, \xi_2, \cdots, \xi_m)$, and the latter is defined by the product of one-dimensional Hermite polynomials,

$$\mathcal{H}_i(\boldsymbol{\xi}) = \prod_{k=1}^{m} H_{i_k}(\xi_k), \text{ where } \sum_{k=1}^{m} i_k = r. \tag{10.3.36}$$

Finally, we note that in (10.3.35) the total number of terms needed to reach *n*-order approximation for a random dimension of *m* is

$$N + 1 = \frac{(m+n)!}{m!n!}. \tag{10.3.37}$$

This can lead to a large number even when random dimensions are small. Fortunately, it has been shown that the convergence rate of expansion of (10.3.35) is exponential (Xiu and Karniadakis 2002) and thus only its low-order terms are needed.

10.3.2.3 Generalized PCE (GPCE)

The above discussion is limited to the use of Hermite polynomials coupled with Gaussian distribution. We have shown in the one-dimensional case that each probability distribution has its own optimal family of orthogonal polynomials. The Wiener–Askey scheme used by GPCE can generate different families of orthogonal polynomials suitable for various continuous and discrete distributions. Besides the Hermite polynomials (for normal distribution), Legendre polynomials (for uniform distribution), and Laguerre polynomials (for exponential distribution) we have considered in the one-dimensional case, GPCE also includes the Jacobi polynomials for Beta distribution, Charlier polynomials for Poisson distribution, and others (Xiu and Karniadakis 2002). GPCE expansions have the following common form

$$f(\boldsymbol{\xi}) \approx \hat{f}(\boldsymbol{\xi}) = \sum_{i=0}^{N} f_i \Psi_i(\boldsymbol{\xi}), \tag{10.3.38}$$

where $\left\{ \Psi_i(\boldsymbol{\xi}) \right\}$, the basis functions of the expansion, is the optimal orthogonal polynomial family for the given probability distribution of $\boldsymbol{\xi}$. Obviously, expansion (10.3.35) is a special case of (10.3.38) when Hermite polynomials $\left\{ \mathcal{H}_i(\boldsymbol{\xi}) \right\}$ are used as $\left\{ \Psi_i(\boldsymbol{\xi}) \right\}$ and the probability distribution is multivariate Gaussian. Moreover, any $\Psi_i(\boldsymbol{\xi})$ can be obtained from its one-dimensional form by simply applying (10.3.36) but replacing the Hermite polynomials in the equation with the selected polynomials.

Example 10.4 *Deriving Multivariate Legendre Polynomials*
Assuming $\left\{ \Psi_i(\xi) \right\}$ is the Legendre polynomial family, the optimal one for homogeneous distribution in hypercube $[-1, 1]^m$ and the dimension of $\boldsymbol{\xi}$ is $m = 3$. Let

us find $\Psi_{16}(\boldsymbol{\xi})$. Because there are 3 first-order terms, 6 second-order terms, 9 third-order terms, $\Psi_{16}(\boldsymbol{\xi})$ must be a third-order term with indexes $i_1 = 0$, $i_2 = 2$, and $i_3 = 1$. Using the 1D Legendre polynomials listed before, we have

$$\Psi_{16}(\boldsymbol{\xi}) = P_{0,2,1}(\xi_1, \xi_2, \xi_3) = P_0(\xi_1)P_2(\xi_2)P_1(\xi_3) = \frac{1}{2}(3\xi_2^2 - 1)\xi_3.$$

By following this way, we can find $\Psi_i(\boldsymbol{\xi})$ for any polynomial chaos and any subscript i. ■

Although the types of probability distributions that GPCE can handle are commonly used, they are still limited. Recent research studies attempt to construct PCE for arbitrary distributions of input uncertainty (Eldred 2009). Loeven et al. (2007) demonstrated the use of the Golub-Welsh algorithm for arbitrary distributions and showed that exponential convergence can be obtained when the polynomials are orthogonal with respect to the PDF. In Wan and Karniadakis (2006), GPCE is used locally to multielements for arbitrary distributions. A data-driven formulation of PCE introduced in Oladyshkin and Nowak (2012) can construct up to *n*th order orthogonal polynomials of a random variable using a few moments. The distribution of the variable can be arbitrary: continuous, discrete, or given by a histogram. In practice, however, the limited data often does not support the construction of higher-order moments other than the first two.

If the random inputs are correlated, one methodology is to perform eigendecomposition on the covariance matrix of correlated inputs to map them to uncorrelated random variables. This methodology is basically the mechanism underlying KL-expansion or PCA, which can decompose correlated random processes into uncorrelated random variates. Another possible approach is to use nonlinear transformations to convert correlated random variables to uncorrelated standard Gaussian variables. For this purpose, the commonly used statistical transformation methods, such as Rosenblatt, Nataf, and Box-Cox transforms, can be applied (Xiu 2010).

10.3.2.4 Uncertainty Propagation

After having expansion (10.3.38), we can use it to generate metamodels for each input parameter $\theta(\boldsymbol{\xi})$ and each model output component $u(\boldsymbol{\xi})$ as

$$\hat{u}(\boldsymbol{\xi}) = \sum_{i=0}^{N} u_i \Psi_i(\boldsymbol{\xi}) \text{ and } \hat{\theta}(\boldsymbol{\xi}) = \sum_{i=0}^{N} \theta_i \Psi_i(\boldsymbol{\xi}). \quad (10.3.39)$$

Then, use these metamodels to analyze the uncertainty propagation instead of using the original model. All of these steps are exactly the same as those used for one-dimensional PCE, except that the computational effort is increased exponentially with the increase of random dimensions (curse of dimensionality).

When the PCE approach is used for analyzing uncertainty propagation, a model $u = \mathcal{M}[(\theta(\boldsymbol{\xi})]$ is replaced approximatly by $\hat{u} = \mathcal{M}[(\hat{\theta}(\boldsymbol{\xi})]$,

$$\sum_{i=0}^{N} u_i \Psi_i(\boldsymbol{\xi}) = \mathcal{M}\left(\sum_{i=0}^{N} \theta_i \Psi_i(\boldsymbol{\xi})\right). \tag{10.3.40}$$

Here, as usual, we use the same number of terms and the same basis polynomials to represent the uncertainty of both model inputs and outputs. In principle, of course, we might use different expansions for them, but the analysis would become much more complicated. Note that our purpose is to find $\hat{u}(\boldsymbol{\xi})$ given $\hat{\theta}(\boldsymbol{\xi})$, which can be done by finding all expansion coefficients $\{u_0, u_1, \cdots, u_N\}$ from (10.3.40).

To shorten expressions, we define the mathematical expectation of two functions $f(\boldsymbol{\xi})$ and $g(\boldsymbol{\xi})$ as

$$\langle f, g \rangle = \langle fg \rangle = \int_{(\Omega)} f(\boldsymbol{\xi}) g(\boldsymbol{\xi}) p(\boldsymbol{\xi}) d\boldsymbol{\xi}. \tag{10.3.41}$$

Using this notation, all coefficients in the PCE of a function $f(\boldsymbol{\xi})$ can be obtained simply by projecting it onto the orthogonal polynomial basis. For any k, we have

$$\langle f, \Psi_k \rangle = \left\langle \sum_i f_i \Psi_i \Psi_k \right\rangle = f_k \langle \Psi_k^2 \rangle \Rightarrow f_k = \frac{\langle f, \Psi_k \rangle}{\langle \Psi_k^2 \rangle} \tag{10.3.42}$$

Now, let $f(\boldsymbol{\xi})$ in the above equation be $u(\boldsymbol{\xi})$. By projecting it onto the polynomial basis, all coefficients in its PCE can be expressed explicitly by

$$u_k = \frac{1}{\langle \Psi_k^2 \rangle} \langle u, \Psi_k \rangle = \frac{1}{\langle \Psi_k^2 \rangle} \int_{(\Omega)} u(\boldsymbol{\xi}) \Psi_k(\boldsymbol{\xi}) p(\boldsymbol{\xi}) d\boldsymbol{\xi}. \tag{10.3.43}$$

Equation (10.3.43) is a multivariate counterpart of (10.3.14) that is derived for the one-dimensional case. As shown in (10.3.16), once $\{u_0, u_1, \cdots, u_N\}$ are found, the statistics used to quantify the uncertainty of the model output can be calculated easily. The first two moments are

$$E[u(\boldsymbol{\xi})] \approx u_0, Var[u(\boldsymbol{\xi})] \approx \sum_{i=1}^{N} u_i^2 \langle \Psi_i^2 \rangle. \tag{10.3.44}$$

High-order moments can also be calculated from these coefficients when needed. The remaining problem now is how to calculate the integrals in (10.3.43).

Intrusive Methods Calculating the integral (10.3.43) requires that $u(\boldsymbol{\xi}) = \mathcal{M}[\theta(\boldsymbol{\xi})]$ be simple and be given explicitly. When $\mathcal{M}[\theta(\boldsymbol{\xi})]$ is the solution of a stochastic PDE given in (10.3.1), it is usually not given analytically. In this case, we can use the *stochastic Galerkin projection* approach to project the PDE model itself, rather than its solution, onto the orthogonal polynomial basis. In other words, for a model $\mathcal{L}\left[u(\boldsymbol{\xi}),\theta(\boldsymbol{\xi})\right]=0$, we use

$$\left\langle \mathcal{L}\left[\sum_{i=0}^{N} u_i \Psi_i(\boldsymbol{\xi}), \sum_{j=0}^{N} \theta_j \Psi_j(\boldsymbol{\xi})\right], \Psi_k \right\rangle = 0, \ (k=0,1,\cdots,N). \quad (10.3.45)$$

This is a set of $N+1$ coupled deterministic PDEs with $\left\{u_0,u_1,\cdots,u_N\right\}$ as unknowns. In general, the structure of these equations is different from that of the original model equation. From a numerical programming perspective, the Galerkin projection method is an "intrusive" one, meaning new codes must be developed for solving the required PCE coefficients.

Example 10.5 *Using Multivariate PCE for UQ*
Consider the following stochastic advection-dispersion model

$$\begin{aligned}
&\mathcal{L}(u,\theta(\boldsymbol{\xi})) \equiv \frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(D(x,\boldsymbol{\xi})\frac{\partial u}{\partial x}\right) + V(x,\boldsymbol{\xi})\frac{\partial u}{\partial x} = 0\\
&\text{subject to}\\
&u(x,t,\boldsymbol{\xi})\,|_{t=0} = g(x),\\
&u(x,t,\boldsymbol{\xi})\,|_{x=0} = f_0(t),\, u(x,t,\boldsymbol{\xi})\,|_{x=1} = f_1(t).
\end{aligned} \quad (10.3.46)$$

Let the PCEs of $u(\boldsymbol{\xi}), D(x,\boldsymbol{\xi})$ and $V(x,\boldsymbol{\xi})$ be given by

$$\hat{u}(\boldsymbol{\xi}) = \sum_{i=1}^{N} u_i \Psi_i(\boldsymbol{\xi}),\ \hat{D}(x,\boldsymbol{\xi}) = \sum_{j=1}^{N} D_j \Psi_j(\boldsymbol{\xi}),\ \hat{V}(x,\boldsymbol{\xi}) = \sum_{j=1}^{N} V_j \Psi_j(\boldsymbol{\xi}). \quad (10.3.47)$$

Substituting these PCEs into the Galerkin projection equation (10.3.45) yields $N+1$ coupled deterministic equations:

$$\begin{aligned}
&\frac{\partial}{\partial t}\sum_{i=0}^{N} u_i \left\langle \Psi_i \Psi_k \right\rangle - \frac{\partial}{\partial x}\left[\sum_{j=0}^{N}\sum_{i=0}^{N} D_j \frac{\partial u_i}{\partial x}\left\langle \Psi_i \Psi_j \Psi_k \right\rangle\right]\\
&+\sum_{j=0}^{N}\sum_{i=0}^{N} V_j \frac{\partial u_i}{\partial x}\left\langle \Psi_i \Psi_j \Psi_k \right\rangle = 0,\ (k=0,1,\cdots,N).
\end{aligned} \quad (10.3.48)$$

Note that $\left\langle \Psi_i \Psi_k \right\rangle = \left\langle \Psi_k^2 \right\rangle \delta_{ik}$ and $\left\langle \Psi_i \Psi_j \Psi_k \right\rangle \neq 0$ for $i \neq j$, the above equations can be rewritten as

$$\begin{aligned}
&\frac{\partial u_k}{\partial t} - \frac{\partial}{\partial x}\left[\sum_{j=0}^{N}\sum_{i=0}^{N} D_j \frac{\partial u_i}{\partial x} C_{ijk}\right] + \sum_{j=0}^{N}\sum_{i=0}^{N} V_j \frac{\partial u_i}{\partial x} C_{ijk} = 0,\\
&(k=0,1,\cdots,N)
\end{aligned} \quad (10.3.49)$$

Each of these equations is subject to the same deterministic initial and boundary conditions given in (10.3.46). The coefficients C_{ijk} in these equations are

$$C_{ijk} = \frac{\left\langle \Psi_i \Psi_j \Psi_k \right\rangle}{\left\langle \Psi_k^2 \right\rangle} = \frac{1}{\left\langle \Psi_k^2 \right\rangle} \int_{(\Omega)} \Psi_i(\xi)\Psi_j(\xi)\Psi_k(\xi)p(\xi)d\xi \tag{10.3.50}$$
$$(i = 1, 2, \cdots, N; \;\; j = 1, 2, \cdots, N)$$

The form of coupled equations (10.3.49) is different from the original PDE (10.3.46). That is why the applications of intrusive methods are limited in practice.

Nonintrusive Methods In comparison, nonintrusive methods only require running the original deterministic model. The departure point is to take a set of samples in the stochastic space

$$\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \cdots, \boldsymbol{\xi}_M, \tag{10.3.51}$$

and evaluate their corresponding model output values:

$$u(\boldsymbol{\xi}_1), u(\boldsymbol{\xi}_2), \cdots, u(\boldsymbol{\xi}_M). \tag{10.3.52}$$

These values are obtained by running the original model M times with different sample parameter values. We have two choices to use these sample output values to find the required PCE coefficients $\left\{u_0, u_1, \cdots, u_N\right\}$ of $\hat{u}(\boldsymbol{\xi})$:

- *Numerical integration*: Choosing an algorithm to calculate the integral on the right-hand side of (10.3.43) with the sample values in (10.3.52), an approximate value of coefficient u_k can be found.
- *Linear regression*: Using the sample values (10.3.52) as the "observed model output values" ($z_j = u(\boldsymbol{\xi}_j), \; j = 1, 2, \cdots, M$), (10.3.40) gives a set of M linear equations

$$\sum_{i=0}^{N} u_i \Psi_i(\boldsymbol{\xi}_j) = z_j + \varepsilon_j, \;\; (j = 1, 2, \cdots, M). \tag{10.3.53}$$

  Here, the truncation error ε_j plays the role of the observation error, u_i and Ψ_i play the roles of "weights" and "basis functions," respectively (see Sect. 8.1.2). All expansion coefficients $\left\{u_0, u_1, \cdots, u_N\right\}$ can be solved from this linear system.

The above nonintrusive methods are attractive because (i) they are more efficient than the Monte Carlo sampling method and (ii) they can be completed using the original forward solution code without code modifications. Although understanding the two methods is easy, implementing them is not easy. We have to know how many samples are needed and where to take these samples in (10.3.51). This problem is closely related to the stochastic collocation method (SCM) to be introduced in the next subsection.

10.3.3 Stochastic Collocation Method

Collocation points refer to predefined nodes in the deterministic collocation finite element method. In the SCM, the term "collocation points" is borrowed to refer to the sample set in (10.3.51) because of the similarity between choosing optimal computation nodes for numerical solution and choosing optimal sampling locations for PCE and SCM.

SCM is actually an interpolation method for random functions. Assume that $u(\boldsymbol{\xi})$ is a function of an *m*-dimensional random vector $\boldsymbol{\xi}$ defined in a region (Ω). SCM finds a metamodel $\hat{u}(\boldsymbol{\xi})$ by interpolation based on *M* values of $u(\boldsymbol{\xi})$ evaluated at *M* collocation points. In SCM, orthogonal polynomials are used as basis functions over the whole region to guarantee fast convergence, and $\hat{u}(\boldsymbol{\xi})$ is directly used to replace $u(\boldsymbol{\xi})$ for integration and uncertainty analysis.

10.3.3.1 One Random Dimensional SCM

Let us start from one-dimensional case ($m = 1$). Lagrange polynomials for one-dimensional interpolation over M sampling (or collocation) points $P = \left\{\xi_1, \xi_2, \cdots, \xi_M\right\}$ are defined by

$$L_{j,P}(\xi) = \prod_{\substack{k=1 \\ k \neq j}}^{M} \frac{\xi - \xi_k}{\xi_j - \xi_k}, \quad (j = 1, 2, \cdots, M) \tag{10.3.54}$$

where L_j is equal to 1 when $\xi = \xi_j$, and 0 when $\xi = \xi_k$. The order of $L_{j,P}$ is $M - 1$ and each collocation point is associated with one polynomial. Any smooth, one-dimensional function $u(\xi)$ can be approximated using a set of one-dimensional Lagrange polynomials as

$$u(\xi) \approx \hat{u}(\xi) = \sum_{j=1}^{M} u(\xi_j) L_{j,P}(\xi). \tag{10.3.55}$$

The above stochastic interpolation has the same form as PCE (10.3.18), but they are different. In PCE, the coefficients of expansion need to be determined, while in (10.3.55) only collocation points need to be determined, all coefficients are obtained in (10.3.52) by running the deterministic forward model. As a result, after we have the metamodel $\hat{u}(\xi)$ in (10.3.55), statistics of $u(\xi)$ can be estimated immediately, for example, its expectation will be

$$E[u(\xi)] = \int_{(\Omega)} \left(\sum_{j=1}^{M} u(\xi_j) L_{j,P}(\xi) \right) p(\xi) d\xi = \sum_{j=1}^{M} w_{j,P} u(\xi_j) \tag{10.3.56}$$

This is a weighted summation of known function values with weights

$$w_{j,P}=\int_{(\Omega)}L_{j,P}(\xi)p(\xi)d\xi,\ \ (j=1,2,\cdots,M) \tag{10.3.57}$$

This integral is easy to calculate after the collocation points are determined. If the required order of approximation(10.3.55) is *n*, we should use $M=n+1$ collocation points. First, find the optimal *M*-orthogonal polynomial of $p(\xi)$, then find the roots of the polynomial and, finally, use these roots as the collocation point set *P*. This is the best selection for fast convergence.

Example 10.6 *Using Legendre Polynomials for SCM*
Assume $p(\xi)$ is a uniform distribution over interval $[-1,1]$ (when the integral region (Ω) is a general interval $[a,b]$, use variable transformation to transfer it into the interval $[-1,1]$). In this case, Legendre polynomial is the optimal selection. If the third-order approximation is required, we have $M=4$. The four roots of Legendre polynomial $P_4(\xi)=\frac{1}{8}(35\xi^4-30\xi^2+3)$ are

$$\xi_1=-0.8611, \xi_2=-0.3400, \xi_3=0.3400,\ \ \xi_4=-0.8611.$$

Using these points as collocation points in (10.3.54), all third-order Lagrange polynomials $L_{j,P}(\xi)$ ($j=1,2,3,4$) are completely determined. Substituting them into (10.3.57) and completing the integration, we can obtain

$$w_{1,P}=0.3479,\ w_{2,P}=0.6521,\ w_{3,P}=0.6521,\ w_{4,P}=0.3479$$

10.3.3.2 Multiple Random Dimensional SCM

Multidimensional interpolation formula can be derived from the one-dimensional formula simply by the tensor product approach, in which (10.3.55) is applied in sequence to all dimensions and a summation over all possible combinations is created. Rewrite the one-dimensional interpolation formula for the *i*th dimension as an interpolation operator

$$\mathcal{U}^i[u(\xi)]=\sum_{j=1}^{M_i}u(\xi_j^i)L_{j,P}^i(\xi), \tag{10.3.58}$$

where M_i is the number of collocation points used. Lagrange interpolation for an *m*-dimensional stochastic function $u(\boldsymbol{\xi})$ is given by

$$\begin{aligned}u(\boldsymbol{\xi})&\cong(\mathcal{U}^{i_1}\otimes\cdots\otimes\mathcal{U}^{i_m})[u]\\&=\sum_{j_1=1}^{M_{i_1}}\cdots\sum_{j_m=1}^{M_{i_m}}u(\xi_{j_1}^{i_1},\cdots,\xi_{j_m}^{i_m})(L_{j_1}^{i_1}\otimes\cdots\otimes L_{j_m}^{i_m}),\end{aligned} \tag{10.3.59}$$

where M_{i_k} represents the number of collocation points used in the *k*th dimension, $\xi_{j_l}^{i_k}$ is the j_l-th collocation point in the i_k-th dimension, $L_{j_l}^{i_k}$ is the Lagrange polynomial associated with this point, and $\otimes$ means the tensor product. By using one index to renumber all terms in (10.3.59), we have expansion

$$u(\boldsymbol{\xi}) \approx \hat{u}(\boldsymbol{\xi}) = \sum_{j=1}^{M} u(\boldsymbol{\xi}_j)\mathbf{L}_{j,P}(\boldsymbol{\xi}), \tag{10.3.60}$$

where $M = M_1 M_2 \cdots M_m$, and $\mathbf{L}_{j,P}(\boldsymbol{\xi})$, a multidimensional Lagrange polynomial associated with point ξ_j, is a tensor product of one-dimensional Lagrange polynomials (10.3.54). It can be obtained with exactly the same process as that used to derive the multidimensional Hermite polynomials from its one-dimensional form in (10.3.35).

Again, the multidimensional stochastic interpolation equation (10.3.60) has the same form as the multidimensional PCE (10.3.38). In fact, they are closely related but different. In PCE, the coefficients of expansion need to be determined, while in the SCM the collocation points need to be determined. After the roots of PCE bases are assigned as the collocation points, (10.3.60) is completely determined. Because of the use of a tensor product, the number of collocation points needed for interpolation (i.e., the number of required function evaluations) increases exponentially with the increase of dimensions. For example, when the required order of approximation is $n = 5$ and the number of random dimensions is $m = 10$, the number of collocation points generated by the tensor product will be 60466176. Fortunately, we do not need so many collocation points. Points that contribute less to the accuracy of the estimation should be discarded.

Using Sparse Grids

Sparse grid method has been proposed as an alternative to the full tensor product method for high-dimensional random space (Fig. 10.6). It is based on the linear combination of tensor products using an integration formula originally proposed by Russian mathematician Smolyak (1963). The isotropic Smolyak formula for approximating a dependent variable u in an *m*-dimensional space is

$$\mathcal{A}_{w,m}(u) = \sum_{w+1\le|\mathbf{i}|\le w+m} (-1)^{w+m-|\mathbf{i}|} \begin{pmatrix} m-1 \\ w+m-|\mathbf{i}| \end{pmatrix} \cdot (\mathcal{U}^{i_1} \otimes \cdots \otimes \mathcal{U}^{i_m}) \tag{10.3.61}$$

where w denotes the level of the Smolyak grid (or the depth of the interpolation), $\mathbf{i} = (i_1, \cdots, i_m)$, with each index representing the number of collocation points used for each dimension, and $|\mathbf{i}| = i_1 + \cdots + i_m$. Smolyak's work did not attract much attention until Barthelmann et al. (2000) proved that Smolyak's formula is optimal and is exact for all complete polynomials of order w, provided that nested grids are used. The level parameter of Smolyak grids thus represents the order of complete polynomials that can be interpolated.

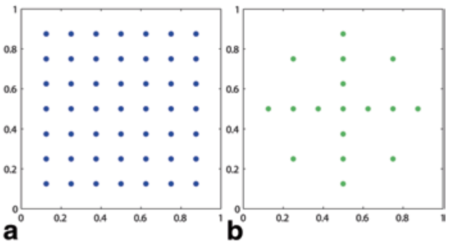


Fig. 10.6 Illustration of a **a** tensor grid and **b** sparse grid

To evaluate $\mathcal{A}_{w,m}(u)$, we only need the knowledge of model outputs at collocation points of the following sparse grid

$$\mathcal{I}(w,m) = \bigcup_{w+1\le|\mathbf{i}|\le w+m} (v^{i_1} \times \cdots \times v^{i_m}) \subset [-1,1]^m, \tag{10.3.62}$$

where $v^i = \{\xi_1^i, \cdots, \xi_{mi}^i\} \subset [-1,1]$ denotes a set of collocation points used by $\mathcal{U}^i$.

The collocation points for constructing Smolyak sparse grids can be generated using several rules. One of the most commonly used is the Clenshaw–Curtis rule, for which the collocations are extrema of Chebyshev polynomials and are given for any choice of M_i as (Xiu and Hesthaven 2005)

$$\xi_j^i = -\cos\left(\frac{\pi(j-1)}{M_i - 1}\right), \; j = 1, \cdots, M_i. \tag{10.3.63}$$

For the special case of $M_i = 1$, only one collocation point is needed (i.e., $\xi^1 = 0$). The number of collocation points grows at each Clenshaw–Curtis grid level according to

$$M_1 = 1, \; M_i = 2^{i-1} + 1, \; for \; i > 1. \tag{10.3.64}$$

With the particular growth rule given by (10.3.64), we obtain a nested set of abscissae, namely $\mathcal{I}(w,m) \subset \mathcal{I}(w+1,m)$. The Clensaw-Curtis rule is appealing because its nestedness leads to computational savings in addition to the sparseness provided by the Smolyak construction.

After the number of required collocation points is reduced significantly, we can use the metamodel $\hat{u}(\xi)$ in (10.3.55) (or its counterpart in (10.3.60) for the multidimensional case) to find the statistics of $u(\xi)$, such as its mean and variance. Moreover, we can simply use the function values evaluated at the collocation points to calculate sample mean and variance, or even to generate a histogram.

Algorithms for constructing sparse grids have been extensively studied. More in-depth discussion on Smolyak sparse grids can be found in Nobile et al. (2008) and Ma and Zabaras (2009). Numerical examples can be found in (Ma and Zabaras 2009; Zhang et al. 2013; Ganapathysubramanian and Zabaras 2007). A Matlab toolbox for constructing sparse grids, SPINTERP, has been developed by Klimke and Wohlmuth (2005). A C++ library for performing sparse grids computations is available in the UQ framework DAKOTA (Eldred et al. 2007).

10.3.3.3 Numerical integration for PCE

Let us return to the nonintrusive method for evaluating the coefficients of PCE by numerical integration. Gaussian quadrature is a natural option for this purpose because it is also based on polynomial approximation. Detailed discussions on this topic can be found in many text books (e.g., Davis and Rabinowitz 2007; Stoer and Bulirsch 2002). The general form of the one-dimensional Gaussian quadrature rule is

$$\int_a^b f(\xi)w(\xi)d\xi \cong \sum_{j=1}^{M} f(\xi_j)w_j, \tag{10.3.65}$$

where $f(\xi)w(\xi)$ is the integrand, $w(\xi)$ is a weighting function. The integral is approximated by a linear combination of function values $f(\xi_j)$ at *M* Gaussian (quadrature) points ξ_j with weights $w_j\,(j = 1, 2, \cdots, M)$, and the accuracy of approximation depends on the number of terms *M*. The fundamental theorem of the Gaussian quadrature states that the optimal abscissae of the *M* quadrature points should be the roots of an *M*-order orthogonal polynomial that matches the weighting function (or the optimal one for the weighting function) and the quadrature weights (w_j) are derived from Lagrange interpolation under the requirement that (10.3.65) becomes exact when $f(\xi)$ is a polynomial with a degree up to $2M-1$.

When the Gaussian quadrature (10.3.65) is used for one-dimensional PCE coefficient estimation in (10.3.14), $f(\xi)$ becomes $u(\xi)\phi_i(\xi)$, the probability distribution $p(\xi)$ plays the role of weighting function $w(\xi)$, and the optimal polynomial can be found according to the given $p(\xi)$. For example, we can use Hermite polynomials for Gaussian distribution over the interval $(-\infty, +\infty)$ and Legendre polynomials for uniform distribution over the interval $(-1, +1)$. Note that if the order of the PCE of $u(\xi)$ is n, then the order of $f(\xi)$ could reach $2n$. Therefore, in order to achieve the *n*th order estimation accuracy, we have to use the roots of $(n+1)$-th order polynomials as the Gaussian quadrature points.

To estimate the coefficients of *m*-dimensional PCE, the *m*-dimensional integral in (10.3.43) has to be calculated. We can use the tensor product approach used in the above to derive the multidimensional Gaussian quadrature formula from its one-dimensional form. Let us rewrite (10.3.65) as an integral operator for the *i*th dimension

$$\mathcal{U}^i[f(\xi)] = \sum_{j=1}^{M_i} f(\xi_j^i)w_j^i. \tag{10.3.66}$$

Then, we have the following *m*-dimensional Gaussian quadrature formula

$$\begin{aligned}\int_{(\Omega)} f(\boldsymbol{\xi})w(\boldsymbol{\xi})d\boldsymbol{\xi} &= (\mathcal{U}^{i_1} \otimes \cdots \otimes \mathcal{U}^{i_m})[f(\boldsymbol{\xi})] \\ &= \sum_{j_1=1}^{M_{i_1}} \cdots \sum_{j_m=1}^{M_{i_m}} f(\xi_{j_1}^{i_1}, \cdots, \xi_{j_m}^{i_m})(w_{j_1}^{i_1} \otimes \cdots \otimes w_{j_m}^{i_m})\end{aligned} \tag{10.3.67}$$

For integration of (10.3.43), $w(\boldsymbol{\xi}) = p(\boldsymbol{\xi}) = p(\xi_1)p(\xi_2)\cdots p(\xi_m)$. For isotropic case, the number of Gaussian points and their abscissae will be the same for all dimensions. The total number of Gaussian points needed in the *m*-dimensional tensor product would be $M > N^m$, where *N* is the order of the PCE. When *m* is small, (10.3.67) can be used as a nonintrusive method for PCE estimation; when *m* is large, however, we have to use the nested Smolyak sparse grids introduced above to complete the numerical integration with much smaller number of collocation points.

10.3.3.4 Regression for PCE

Let us return to the nonintrusive method of using regression to estimate the coefficients of PCE. Random sampling or more efficient stratified sampling techniques can be used to generate sampling points in (10.3.53) (Reagan et al. 2003). For example, LHS can be used to generate samples drawn from equiprobable partitioning of the probability space (see Sect. 10.1.4). It is recommended that the number of collocation points should be at least twice of the number of terms in PCE, (i.e., $M > 2(N+1)$) (Hosder et al. 2007). After forward solutions in (10.3.52) are obtained, (10.3.53) is turned into an overdetermined system

$$\begin{pmatrix} \Psi_0(\xi_1) & \cdots & \Psi_N(\xi_1) \\ \vdots & \ddots & \cdots \\ \Psi_0(\xi_M) & \cdots & \Psi_N(\xi_M) \end{pmatrix} \begin{pmatrix} u_0 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} u(\xi_1) \\ \vdots \\ u(\xi_M) \end{pmatrix}. \tag{10.3.68}$$

All PCE coefficients can then be solved from this system.

The Probability Collocation Method (PCM) This method uses roots of orthogonal polynomials as sampling (or collocation) points to construct a determined, instead of an overdetermined system (i.e., $M = N+1$ in (10.3.68)) (Li and Zhang 2007; Tang et al. 2007). The collocation points are chosen from the roots of polynomials at one degree higher than that used in the PCE of $\hat{u}(\xi)$. As an example, for second-order PCE using Hermite polynomials, the collocation points are chosen from the roots of the third-order Hermite polynomial $\xi^3 - 3\xi$, namely, $-\sqrt{3}, 0, \sqrt{3}$. Each collocation point is a permutation of these roots for each random variable. The number of possible permutations is $(n+1)^m$, which is typically much greater than the number of unknowns ($N+1$). Therefore, only a subset of collocation points is selected in a way that ensures the resulting coefficient matrix stays full rank.

Li and Zhang (2007) suggested a heuristic scheme for selecting collocation points, in which roots corresponding to high probability regions in each dimension are selected first before moving to other combinations involving lower probability regions. Thus, among all three roots in the third-order Hermite polynomial mentioned above, zero corresponds to the highest probability region of the standard Gaussian distribution and should be selected before the other two. Each time before a new collocation point is added, the resulting matrix is tested to ensure it remains full row rank. Otherwise, the new point is discarded and another combination is checked. The process is repeated until $N+1$ collocation points are selected. Obviously, PCM needs less collocation points than other methods of PCE. Critiques of this approach are that it has no explicit control on approximation error and the resulting solution may be nonunique. This issue can be especially problematic for low-order expansion when fewer points are involved. In this case, the truncation error (i.e., the ε term in (10.3.53)), may cause significant estimation error. The other potential problem is the computational time required to check the matrix rank, which becomes prohibitive for high dimensions. Thus, the random dimensions of UQ problems considered are usually limited to be a small number (~10).

Example 10.7 *Solving a One-Dimensional Problem Using PCM*
Let us consider one-dimensional groundwater flow under confined conditions

$$\begin{aligned}&S\frac{\partial h(x,t)}{\partial t}=T_h\frac{\partial^2 h(x,t)}{\partial x^2}+\sum_{i=1}^{N_w}Q_i\delta(x-x_i)\\&h(0)=100\ [\mathrm{L}],\\&h(l)=90\ [\mathrm{L}],\end{aligned}\tag{10.3.69}$$

where $h(x,t)$ is hydraulic head, $S=1\times10^{-4}$ is storativity [-], T_h is transmissivity [L^2/T], and the total domain length is $l=100$ [L]. The domain is divided into three equal-length zones. A pumping well is placed at the boundary between zone 1 and 2, and an injection well is placed at the boundary between zone 2 and 3 (Fig. 10.7). The pumping/injection rates are also labeled in Fig. 10.7. For illustration purposes, we assume that hydraulic conductivity values in all three zones are independent of each other. The random dimension for our problem is thus $m=3$.

Assuming that T_h of all three zones are uniformly distributed random variables, with their respective range of variations defined as

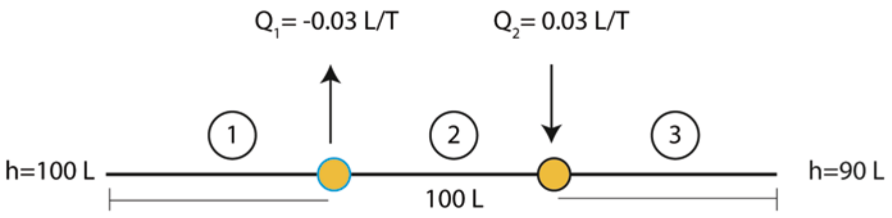


Fig. 10.7 Problem settings for Example 10.10

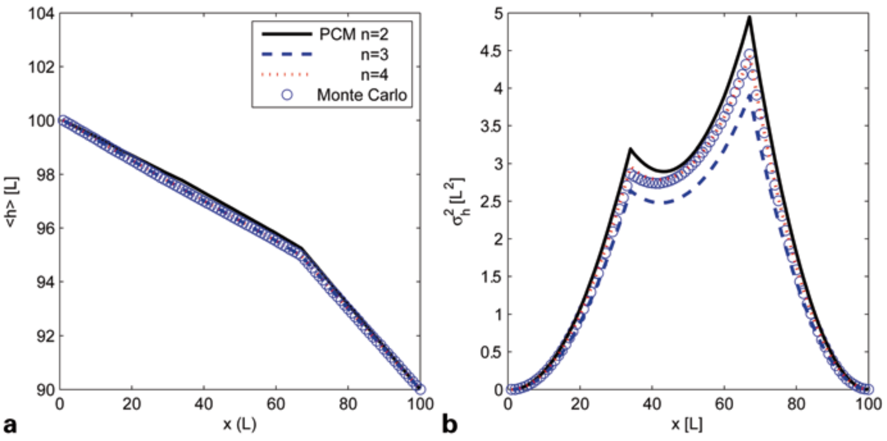


Fig. 10.8 Comparison of **a** mean and **b** variance of hydraulic head obtained by PCM using Legendre polynomial chaos and Monte Carlo simulation (*open circle*)

$$1 \le T_{h,1} \le 20, 1 \le T_{h,2} \le 20, \, and \; 0.1 \le T_{h,3} \le 10. \tag{10.3.70}$$

We are interested in quantifying the uncertainty in hydraulic head, $h(x,t)$. The Legendre polynomial basis will be used here because of the uniform distribution assumption.

Figure 10.8a shows the mean head obtained by PCM using different orders of polynomial expansion. For comparison, the ensemble mean obtained using 30,000 Monte Carlo simulations is also shown (open circles) and is regarded as the "true" solution. For this example, the number of model runs required for orders $n = 2, 3, 4$ is 10, 20, and 35, respectively, according to Eq. (10.3.37). The mean head solution obtained by PCE essentially overlaps with that from the Monte Carlo simulation when $n > 2$. Figure 10.8b compares the head variance obtained using both PCM and Monte Carlo simulation. The head variance obtained by PCM from the fourth-order expansion ($n = 4$) reproduced the Monte Carlo solution well; however, significant deviations can be observed in the other two cases.

This example shows also the importance of using matching (or optimal) basis. Figure 10.9a, b suggest that when Hermite basis (which is not optimal for uniform distribution) is used, a higher-order expansion is needed in-order to achieve the similar accuracy.

At last, we repeated the same exercise using the least squares method (i.e., the regression method) to solve an overdetermined system of equations. Figure 10.10 shows the estimated head variance for different orders of expansion and different numbers of collocation points (indicated on the plot). Compared to Fig. 10.8b, we see that the overdetermined solutions for lower-order expansions are significantly improved. In summary, PCM requires a smaller number of runs, but may need a higher-order expansion, especially for highly nonlinear problems. On the other hand, the overdetermined method works well with low-order polynomial chaos

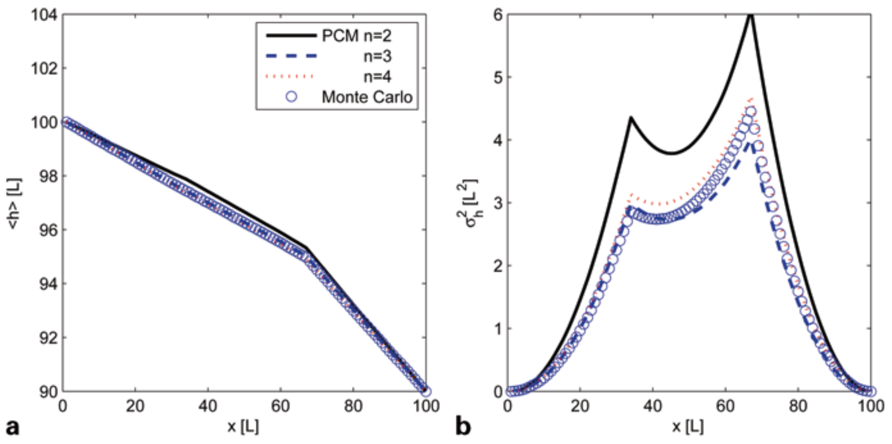


Fig. 10.9 Comparison of **a** mean and **b** variance of hydraulic head obtained by PCM using Hermite polynomial chaos and Monte Carlo simulation (*open circle*)

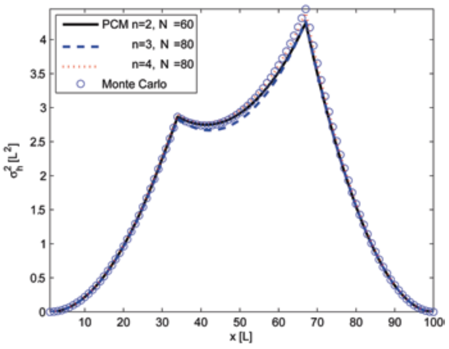


Fig. 10.10 Hydraulic head variance obtained using overdetermined PCM, where the order of expansion and number of collocation points (N) are indicated on the plot

expansion, but usually requires at least twice (or even more) the number of runs than PCM does. When the computational cost is relatively low and the dimensions of random vector are large, we recommend that the user adopt the least squares method. Recently, Liao and Zhang (2014) proposed a location-based transformed probabilistic collocation method (xTPCM), which is shown to outperform the PCM for strongly nonlinear problems. The method uses a transform to map the model output values to a level set.

10.4 Assessment of Model Structure Uncertainty

In Chap. 7, we considered the complicated problem of identifying model structures and unknown parameters simultaneously, namely, the EIP. The goal there was to identify an optimal model among a set of plausible models by solving a min-max problem and using either goodness-of-fit criterion or model selection criterion (e.g., AIC or BIC). There are situations, however, in which no single model exhibits clear superiority over others. Such situations can arise, for example, when data and conceptualization uncertainties preclude selection of a single best model or, in the case of environmental management, when multiple working hypotheses and multiple competing models coexist. It may be advantageous in those situations to perform multimodel inference using a set of models, instead of just a single model. Such is the paradigm behind various model averaging techniques, which are concerned more about achieving the most robust outcome than about the best possible outcome. That being said, however, prediction obtained from model averaging can often outperform that from a single model if the latter only explores partial model space. For example, we have seen in Chap. 8 that ensemble methods can be used to boost the performance of unstable methods such as ANN. The choice of combining multiple models is case-dependent and is ultimately related to how well the model space is represented via the selected model(s). Compared with the single model selection approach, an advantage of the multimodel formulation is that it naturally provides a framework for assessment of model uncertainty.

Breiman (1992) coined a special phase "quiet scandal of statistics" to refer to the lack of recognition of model selection uncertainty in the statistical inference, where confidence intervals were often calculated based on a calibrated single model. Although the topic of combining model prediction has long been studied (Bates and Granger 1969; Leamer 1978), theoretical development only started to flourish in 1990s, which are related to the Bayesian model averaging (BMA) theory (e.g., Gelfand and Dey 1994; Hoeting et al. 1999; Raftery 1996; Draper 1995) and MCMC (Besag and Green 1993; George and McCulloch 1993). Model structure identification and uncertainty quantification also started to appear in the EWR community around the same era, such as in hydrologic modeling (Duan et al. 1992; Beven 1993) and groundwater modeling (Sun and Yeh 1985; Sun 1994).

A main motivation in studying model averaging techniques is related to their ability to convey uncertainties in model structures. Here we distinguish between model structure uncertainty and the so-called deep uncertainty, which is reserved for describing unverifiable probabilities pertaining to long-term events (Lempert et al. 2006). The former is handled by assigning weights to models that describe the same underlying physical process and observations, but with different degrees of complexity. While in the case of the latter, it is often hard to construct a single model, not to mention multiple classes of models; therefore, the best one can do is to conduct a scenario analysis, with the hope that the resulting scenario set, especially the worst-case scenario, can sufficiently capture outcomes of future events in a manner that also minimizes the foreseen risk. Thus, although the two types of uncertainties may both benefit from the use of multiple models, they represent different applications. As a commonly used example, weather forecasting is often based on an ensemble of numerical weather forecast models, which is model averaging; on the other hand, global climate warming forecasts are based on multiple projections into the future that are dependent on different carbon emission scenarios.

Model averaging is an active research area and different methodologies have arisen from Bayesian, frequentist, and information theoretic frameworks. Interestingly, EWR applications have played a prominent role in the evolvement and shaping of model averaging theories. In the following, we will focus on several dominant approaches often used in the EWR literature, including BMA, generalized linear unbiased estimation (GLUE), and multimodel inference.

10.4.1 Bayesian Modeling Averaging

BMA provides a general and elegant framework for incorporating both parameter and model structure uncertainties. It introduces a model space in addition to the parameter space that was introduced in Chap. 2 (Fig. 2.1) and discussed in details in Chaps. 4 and 7. A closed-model space, as defined in Sect. 7.1, consists of all feasible system models as well as the "real system" as its members. In practice, we can have access to at most a subset of the model space that may or may not contain the true model. In the extreme, information theorists completely abandon the notion of a "true model" because the reality is elusive, infinite-dimensional, and because modeling in their eyes is only an "exercise in the approximation of explainable information in the empirical data" (Burnham and Anderson 1998). The model construction process is, thus, driven either by the available data or by the planned model uses. We will mainly focus on the aspect of data-driven modeling in this section and turn to the objective-driven paradigm in Chap. 12.

Let us consider a set of K plausible models, say, $\{\mathcal{M}_k\}_{k=1}^{K}$, where each model $\mathcal{M}_k$ is in the form of the generic model given in $\mathbf{u} = \mathcal{M}(\mathbf{x}, \boldsymbol{\theta})$. For the time being, let us assume that we are able to assign a prior probability mass function $P(\mathcal{M})$ on the model set such that

$$\sum_{k=1}^{K} P(\mathcal{M}_k) = 1.$$

Also, we have knowledge of the prior probability mass function for parameters of each model, $P(\boldsymbol{\theta}_k \mid \mathcal{M}_k)$. For ease of notation, we assume the parameter space is discrete. Switching to continuous parameter space is trivial. By applying the chain rule of conditional probability, the joint probability mass function of the model structure, parameter, and data can be decomposed into a product of three terms

$$P\left(\boldsymbol{\theta}_k, \mathcal{M}_k, \mathbf{d}\right) = P\left(\mathbf{d} \mid \boldsymbol{\theta}_k, \mathcal{M}_k\right) P\left(\boldsymbol{\theta}_k \mid \mathcal{M}_k\right) P(\mathcal{M}_k), \; k = 1, \cdots, K, \quad (10.4.1)$$

where $\mathbf{d}$ denotes all observations. The right-hand side of (10.4.1) is a straightforward extension of the hierarchical Bayesian parameter estimation framework introduced in Sect. 7.4, with the addition of model space. Therefore, BMA includes the hyperparameter estimation problem as a special case, in which all models share the same model structure, but with different hyperparameter values.

Following the data-driven paradigm, the weight of each model is quantified through some distance measure between data and model output (i.e., marginal likelihood of the model $\mathcal{M}_k$), which is obtained by integrating over the model's parameter space (Clyde and George 2004)

$$P(\mathbf{d}|\mathcal{M}_k) = \sum\nolimits_{\Omega} P(\mathbf{d}|\boldsymbol{\theta}_k, \mathcal{M}_k) P(\boldsymbol{\theta}_k|\mathcal{M}_k), \tag{10.4.2}$$

where the summation is performed over the universe of parameters, Ω. The posterior probability of each model under consideration is established in terms of likelihood $P(\mathbf{d}|\mathcal{M}_k)$ and the corresponding model prior probability $P(\mathcal{M}_k)$

$$P(\mathcal{M}_k|\mathbf{d}) = \frac{P(\mathbf{d}|\mathcal{M}_k)P(\mathcal{M}_k)}{\sum_k P(\mathbf{d}|\mathcal{M}_k)P(\mathcal{M}_k)}. \tag{10.4.3}$$

Equation (10.4.3) reflects how well each model performs for the given data and in accordance with prior information. The posterior distribution of a random variable of interest, say, $\mathbf{u}$, can then be expressed using the posterior model probabilities as weights

$$P(\mathbf{u}|\mathbf{d}) = \sum_{k=1}^{K} P(\mathbf{u}|\mathcal{M}_k, \mathbf{d}) P(\mathcal{M}_k|\mathbf{d}). \tag{10.4.4}$$

Finally, the mean of BMA estimate can be obtained as a weighted sum of its mean value obtained under each model (Hoeting et al. 1999)

$$E(\mathbf{u}|\mathbf{d}) = \sum_{k=1}^{K} E(\mathbf{u}|\mathcal{M}_k, \mathbf{d}) P(\mathcal{M}_k|\mathbf{d}) \tag{10.4.5}$$

and the associated BMA variance is

$$Var(\mathbf{u}|\mathbf{d}) = \sum_{k=1}^{K} (Var(\mathbf{u}|\mathcal{M}_k, \mathbf{d}) - E^2(\mathbf{u}|\mathcal{M}_k, \mathbf{d})) P(\mathcal{M}_k|\mathbf{d}). \tag{10.4.6}$$

Equations (10.4.2–10.4.6) consist of the backbone of the BMA framework which, although looks conceptually elegant, can be challenging and costly to implement in practice. In the following, we will briefly discuss some issues related to (i) the specification of model priors, $P(\mathcal{M}_k)$, (ii) calculation of model posteriors, and (iii) the ever-lasting question, "what if the true model is not included in the candidate models."

10.4.1.1 Specification of Models

We start by asking the question: Which models to include in a candidate set? As with any Baysesian approach, the effectiveness of BMA rests strongly on the choice of probability mass functions or, in the continuous case, the PDFs. Ideally, the prior distribution specified for parameters and models should provide a thorough representation of prior uncertainty. If all models are generated in a similar fashion, such as random realizations used in a Monte Carlo simulation, we may choose a uniform prior. In reality, we often have models from different model classes. By definition, a *model class* contains models with the same functional form that is parameterized with the same number and type of parameters.

Example 10.8 *Model Classes in EWR*
A large number of BMA studies in the statistical literature are concerned with multivariate regression. Here we present some examples of model classes in EWR applications. The division of model classes, as we will see from below, is by no means black and white.

In hydrometeorology, the concept of multimodel or ensemble prediction is widely adopted. Duan et al. (2007) and Renard et al. (2010) combined predictions of multiple conceptual rainfall-runoff models to quantify model structure uncertainty. The rainfall-runoff models considered by the authors are lumped-parameter models, each having different conceptualization and parameterization schemes. Thus, each conceptual rainfall-runoff model is considered a separate model class.

In hydrogeology, Ye et al. (2008) considered geostatistical modeling of a permeability dataset. In their study, each permeability variogram model (e.g., exponential, power law, and spherical) is considered a separate model with its own hyperparameters. Variogram models present a unique case because they represent covariance of the random variable under study, but not the direct causal relationship often seen in other BMA applications. On the other hand, variogram represents only one type of parameterization techniques that rely on the assumption of statistical stationarity of the underlying spatial field. Parameterization using any other techniques given in Chap. 6 may yield different model classes. Thus, Ye et al. (2008) largely dealt with one model class.

Singh et al. (2010) considered combining predictions of nine groundwater models. The models collected by Singh et al. (2010) are all distributed groundwater models solved by the code MODFLOW. Although the conceptual models reflect a combination of uncertainties in geologic frameworks and recharge mechanisms, the outcome is flux variations over the top layer and permeability variations in each of the numerical blocks after mapping the conceptual frameworks to the same numerical model. Therefore, on the one hand, one may argue that each model represents a distinct conceptualization; on the other hand, one may argue, from the numerical modeling perspective, that all nine models are largely from one model class (i.e., MODFLOW) and the authors mainly dealt with parametric uncertainty.

The few examples presented here give a glimpse of practical challenges of implementing BMA, especially for distributed models. When the number of degrees of freedom (DOF) gets larger, model averaging can become increasingly challenging because of the inherent difficulty in enumerating and discriminating candidate models. ■

The first task of BMA is filtering out "meaningless" models to form a set of plausible models. Failure to do so will lead to inclusion of many trivial or even wrong model members that could significantly "dilute" the contribution of more significant model members. But how do we tell a "meaningless" model from a plausible model? In the absence of substantive prior information, the model prior probability distribution is often elicited from domain experts, who may have contributed to the generation of models either in the same study or in similar studies. As mentioned before, critics of the Bayesian approach often question the objectivity of expert elicitation processes. Chipman et al. (2001) delineated several practical strategies for stepping away from a pure subjective approach. For example, one strategy involves constructing noninformative, semi-automatic priors and use subjective and empirical Bayes consideration only when needed. Chipman et al. (2001) recommended to start with a uniform prior over models and use imaginary training data to construct a more informative prior. In principle, such a procedure may be performed using historical data or data collected from analog studies.

10.4.1.2 Calculation of Posteriors

When information becomes available, we can calculate posterior model probabilities. The principle underlying this step is essentially the same as that behind the importance-sampling filters (Chap. 9), which consists of repeated elimination and normalization steps. Madigan and Raftery (1994) suggested the use of Occam's window method to eliminate inferior candidates for a given set of observations

$$\frac{\max\{P(\mathcal{M}_l|\mathbf{d})\}}{P(\mathcal{M}_k|\mathbf{d})} \leq C_{\mathcal{M}}, \tag{10.4.7}$$

where $C_{\mathcal{M}}$ is a user-specified threshold and the nominator represents the maximum posterior model probability calculated using all candidate models. A potential issue associated with the one-pass, model-elimination procedure is that good candidates may be accidentally removed because of data limitations. In contrast, the sequential Monte Carlo approaches may be used to evolve an ensemble of models. Similarly, cross-validation may also be used to confirm the selection of models.

Because the model posterior PDF is rarely available in closed form for real applications, a number of MCMC schemes have been proposed to explore the model space. Examples include the direct simulation method (Chib 1995), the MCMC model composition method (MC^3) (Madigan et al. 1995), the stochastic search variable selection (SSVS) (George and McCulloch 1993), and the reversible jump sampler (Green 1995). Various MCMC schemes differ in the specification of proposal distribution and the way model space and parameter space is sampled. For example, the popular reverse jump sampler proposed by Green (1995) jumps "between parameter subspaces of differing dimensionality" while preserving the balance within each move type. Thus, to move from model $\mathcal{M}_k$ with parameter $\boldsymbol{\theta}_k$ to model $\mathcal{M}_k'$ with parameter $\boldsymbol{\theta}_k'$, the acceptance probability is given as

$$\alpha = \min\left(1, \frac{p\left(\mathcal{M}_k', \boldsymbol{\theta}_k' \middle| \mathbf{d}\right) q\left(\mathcal{M}_k, \boldsymbol{\theta}_k; \mathcal{M}_k', \boldsymbol{\theta}_k'\right)}{p\left(\mathcal{M}_k, \boldsymbol{\theta}_k \middle| \mathbf{d}\right) q\left(\mathcal{M}_k', \boldsymbol{\theta}_k'; \mathcal{M}_k, \boldsymbol{\theta}_k\right)}\right) \tag{10.4.8}$$

where $q(\cdot)$ is the proposal distribution. Furthermore, if the proposal distribution is separable, then the acceptance ratio is simplified to (Godsill 2001)

$$\alpha = \min\left(1, \frac{p\left(\mathcal{M}_k', \boldsymbol{\theta}_k' \middle| \mathbf{d}\right) q_1\left(\mathcal{M}_k; \mathcal{M}_k'\right) q_2\left(\boldsymbol{\theta}_k; \boldsymbol{\theta}_k'\right)}{p\left(\mathcal{M}_k, \boldsymbol{\theta}_k \middle| \mathbf{d}\right) q_1\left(\mathcal{M}_k'; \mathcal{M}_k\right) q_2\left(\boldsymbol{\theta}_k'; \boldsymbol{\theta}_k\right)}\right) \tag{10.4.9}$$

One possible situation in which the proposal distribution is separable is when parameter space has a fixed dimension.

10.4.1.3 Model Averaging Without the True Model

Bernado and Smith (1994) defined two types of model sets, the closed set and the open set, with the former including the true model as a member. Many practitioners have associated BMA with the closed-model set and regarded it as the greatest limitation of BMA. It is true that BMA dwells on the idea that the specification of prior model PDF should adequately span the model space and allow the posteriori information to "recover" the true model, which implies that all candidate models should be in the neighborhood of the "true" data-generating model to begin with. The reverse is often not true—a high-probability model does not necessarily imply that the model is close to the true model.

Hoeting et al. (1999) argued that the basic principles of BMA do not restrict the applicability of the method to closed-model sets—BMA simply treats the model as another unknown parameter. Thus, Hoeting et al. (1999) proposed an adaptive setting where, at each stage, models are continuously updated and decisions are made to either include or exclude certain models. Such a view is exactly the one behind sequential Monte Carlo methods. For most practical problems in EWR, it is hard to define a true model because the model structure, parameter, and measurement errors are highly intertwined. Thus, the BMA and other model averaging methods should be regarded mostly as a means for propagating model uncertainty, but not a means for uncovering the true model.

Example 10.9 *Use BMA in Ensemble Forecasting*
Raftery et al. (2005) extended BMA from regression models to dynamical models for ensemble weather forecasting. Sources of uncertainty in numerical weather forecasts may include uncertainty under initial conditions, lateral boundary conditions, model physics, as well as discretization and integration methods. The effect of uncertainty is manifested as bias and additive error in the forecast variables. Raftery et al. (2005) used the following linear regression model to relate the *k*th forecast, f_k, to observations d

$$d = a_k + b_k f_k + \varepsilon_k, \quad k = 1, \cdots, K,$$

where ε_k is the additive error, a_k and b_k are bias-correction parameters. Assuming normal observation errors, the BMA predictive mean is the weighted mean of all forecasts

$$E(u|d) = \sum_{k=1}^{K} w_k (a_k + b_k f_k)$$

in which u is the state variable such as temperature or sea-level pressure and w_k are weights. Bias correction parameters a_k and b_k, as well as weights w_k, are estimated using training data on f_k. The BMA predictive variance is

$$Var\left(u|d\right) = \sum_{k=1}^{K} w_k \left[(a_k + b_k f_k) - \sum_{i=1}^{K} w_i (a_i + b_i f_i)\right]^2 + \sigma^2$$

where σ^2 is the variance of the error term; the first term on the right-hand side accounts for between-forecast variance, and the second term accounts for within-forecast variance.

10.4.2 Other Model Averaging Approaches

10.4.2.1 Multimodel Inference

Multimodel inference uses model selection criteria to rank competing models and to weigh the relative support for each one (Ajami et al. 2007; Butts et al. 2004; Georgakakos et al. 2004; Li and Tsai 2009; Poeter and Anderson 2005). Burnham and Anderson (2002) used AIC to quantify the plausibility of each model. If the AIC difference between the *k*th model and that of the best performer of all candidate models is defined as

$$\Delta_k = AIC_k - AIC_{\min}, \tag{10.4.10}$$

then the Akaike weight of the *k*th model can be computed as

$$w_k = \frac{\exp(-\frac{1}{2}\Delta_k)}{\sum_{l=1}^{K} \exp(-\frac{1}{2}\Delta_l)}. \tag{10.4.11}$$

In contrast to the model posterior probability used in BMA, multimodel inference assigns Akaike weights to candidate models. The application of multimodel inference generally involves two steps. In the first step, a set of models are generated and are calibrated using the available data. In the second step, model averaging is performed using Akaike weights. Thus, the multimodel average of a quantity of interest $\mathbf{u}$ is

$$\bar{\hat{\mathbf{u}}} = \sum_{k=1}^{K} w_k \hat{\mathbf{u}}_k, \tag{10.4.12}$$

where $\hat{\mathbf{u}}_k$ is the estimate obtained by the *k*th model.

10.4.2.2 Generalized Likelihood Uncertainty Estimation (GLUE)

GLUE, introduced by Keith Beven and co-workers in a series of papers (Beven 1993; Beven and Binley 1992; Beven and Freer 2001), provides another methodology for model averaging. The fundamental philosophy underlying GLUE is that no single optimal model or parameter set exists because of structural and parameter uncertainties. Beven et al. coined the term "equifinality" to refer such a notion of nonuniqueness. The goal of GLUE is to find a set of models that satisfy some conditions of acceptability. Thus, for a given set of candidate models, GLUE aims to find *behavioral models* that properly reflect the uncertainties arising from the modeling process and that reproduce the observations (Beven and Binley 1992). In their original paper, Beven and Binley (1992) did not specify a specific likelihood measure for weighing model performance. Thus, not only Bayesian type of likelihood functions, but also other methods for ranking model performance (Nash-Sutcliff efficiency, possibility) may be used.

Toward the practical use of the GLUE framework, Beven (2012) envisioned a six-step process:

1. Define the range of model structures to be considered.
2. Reject model structures that cannot be justified as physically feasible based on a priori information.
3. Define parameter ranges for each model.
4. Reject parameter combinations that cannot be justified as physically feasible a priori.
5. Compare the predictions of each potential model with the available observed data and reject any models which produce unacceptable predictions, taking account of the estimated error in the observations.
6. Use the remaining behavioral models to perform prediction and model uncertainty analysis.

The above process indicates that the outcome of GLUE analysis is an ensemble of behavioral models, each associated with a likelihood value. Prediction bounds (confidence intervals) can be estimated in GLUE by using the CDF of the likelihoods of ensemble.

The main criticism of GLUE is that a significant level of subjectivity may go into steps 1–4. In the extreme (but not rare) case that none of the models considered are selected as behavioral models, an important and potentially costly decision must be made regarding whether to go back and revise models or data collection plans, or to relax the rejection criteria.

In hydrological applications, the model calibration often hinges on the assumption that past historical data provide prediction of the future. However, models that do not perform well may not necessarily have incorrect structures; rather, the poor performance may be caused by epistemic uncertainty that was not accounted for during calibration. In a recent review of GLUE, Beven and Binley (2013) mentioned that "the debate between GLUE and formal statistical approaches is still on-going… with no sign of real resolution because there is no right answer to the problem of epistemic uncertainties." Thus, all UQ techniques mentioned in previous sections are relevant to the "correct" use of GLUE in practice.

To conclude this chapter, we emphasize that UQ plays an important role in virtually all aspects of EWR modeling. Because of uncertainty involved in most applications, UQ should be regarded as an experimental design tool for guiding information improvement, rather than the ending point of a project (Fig. 10.1).

10.5 Review Questions

1. How important is UQ in the modeling? Use a model in your study area to show what kinds of uncertainty are associated with the model and how these uncertainties are currently quantified.
2. In what situations, the Monte Carlo sampling techniques will become inefficient for UQ?
3. Give the names, definitions, and explanations of $V, V_i, V_{-i}, V_{i,j}, S_i, S_i^T$, respectively, and the relationships between them.
4. Show the algorithm of using GSA for UQ step by step, when the model to be considered has two inputs $\left\{\theta_1, \theta_2\right\}$ and three outputs $\left\{u_1, u_2, u_3\right\}$.
5. Show the algorithm of using the Morris sampling method for UQ step by step. How can we make the method more accurate?
6. Explain how GSA is used to screen input factors for a given model application. Should the screen results be different for different model applications?
7. How do we determine the random dimension of a model?
8. Assume that the initial condition of the model considered in Example 10.3 depends on two germs, i.e., in equation (10.3.17), let $u(x, t, \xi_1, \xi_2)\,|_{t=0} = g(x, \xi_1, \xi_2)$. (a) Find the Hermite orthogonal expansion of $u(x, t; \xi_1, \xi_2)$, and (b) outline how the linear regression PCE can be used to find the UQ for this model.
9. Give a detailed description of SCM for the two-random dimensional case. (a) Use SCM to find the UQ for the same model in the previous question, and (b) describe the differences between PCE and SCM.
10. Summarize how the multimodel methods are used in practice for modeling a complex system without knowing its real structure. Can the existing methods give a reliable UQ?

Chapter 11
Optimal Experimental Design

In environmental and water resource (EWR) engineering, different types of design problems exist, such as operation design, monitoring design, detection design, and remediation design. This chapter is devoted to the subject of experimental design for model calibration and parameter estimation. Experimental design plays a critical role in model construction because the reliability of a model is mainly dependent on the quantity and quality of data used for its calibration. The experimental design thus dictates how data should be collected in field campaigns and how many observations are needed. An optimal experimental design (OED), when it is executed, should provide the maximum amount of information with the minimum cost. Basic concepts, theories, and methods of OED are well established in statistics and have been applied to various scientific and engineering disciplines (Silvey 1980; Pázman 1986; Kennedy and O'Hagan 2001; Melas 2006; Atkinson et al. 2007; Fedorov and Hackl 1997).

In EWR fields, however, OED is still a very challenging problem because of the following difficulties:

- A EWR model is usually nonlinear and governed by partial differential equations (PDEs)
- The unknown parameters are usually distributed
- The model scale is large but the impact range of an experiment is small
- Experimental and sampling costs are high
- Observation data are often correlated

Fortunately, in most real cases, historical observation data have been collected based on some preliminary design, a model has been developed with the existing data, and the reliability of the model has been assessed. In such cases, we do not need a completely new design (i.e., the *prospective design*); instead, we only need a revision of the previous design (i.e., the *retrospective design*). When the model performance is satisfactory, the new design determines how to delete some observation points to save operating costs. On the other hand, when the model needs improvement, the new design determines how to collect more observations to decrease the

N-Z. Sun, A. Sun, *Model Calibration and Parameter Estimation,*
DOI 10.1007/978-1-4939-2323-6_11

model uncertainty. A retrospective design is suboptimal, but is easier to make than a prospective one.

In this chapter, besides introducing traditional methods of OED, special considerations are given to EWR modeling. Section 11.1 introduces basic concepts of experimental design for inversion. OED is then formulated as a multiobjective optimization problem. Designing an experiment for model inversion must be done in light of prior information because less information is needed from the experiment (less costs), if we have more prior information and vice versa. Therefore, it is natural to consider OED in the Bayesian framework. Section 11.2 is an introduction to OED for linear model inversion. Alphabetic optimality criteria are derived for both uniform and Gaussian prior information. Section 11.3 considers the design of experiments for nonlinear model inversion. This is a very challenging problem because the optimal design for parameter estimation becomes dependent on the true parameter to be estimated. Bayesian and max-min robust design approaches are introduced in this section, but they are difficult to be applied to large-scale systems. Section 11.4 gives more attention to EWR systems. Instead of parameter uncertainty, the cost-effectiveness and prediction reliability are considered as the major objectives of experimental design. We will see that the OED for model inversion may not be the same as the OED for model prediction.

Many topics that we have covered in previous chapters, such as single and multiobjective optimization, statistical inversion, model differentiation, parameterization, model reduction, and uncertainty analysis are necessary knowledge for reading this chapter.

11.1 Basic Concepts and Formulation

11.1.1 Decision Variables of an Experimental Design

An experimental design is a plan for obtaining an "excitation-response" relationship from a system to be modeled. It consists of two parts: the excitation part and the observation part. The former makes decisions (D_{exc}) on how to excite the system, such as changing the sink/source term and/or boundary conditions naturally or artificially; the latter makes decisions (D_{obs}) on where and how to measure the system responses, such as specifying the locations and times to observe system states. An experimental design problem is thus a decision-making problem involving a set of *design variables* $D = \left\{D_{exc}, D_{obs}\right\}$, as well as experimental objectives. Observation network (sensor) design is a special case of experimental design when the excitation part is predetermined. Note that design variables can vary continuously or can take only discrete values. For example, in the design of aquifer pumping tests, continuous variables are injection/pumping rates and discrete variables are the set of observation wells to be selected from all existing wells or to be drilled from discrete spatial locations. A variable V that depends on an experimental design D will be denoted either by V_D as in previous chapters or by $V(D)$ when D is used to represent the design variables.

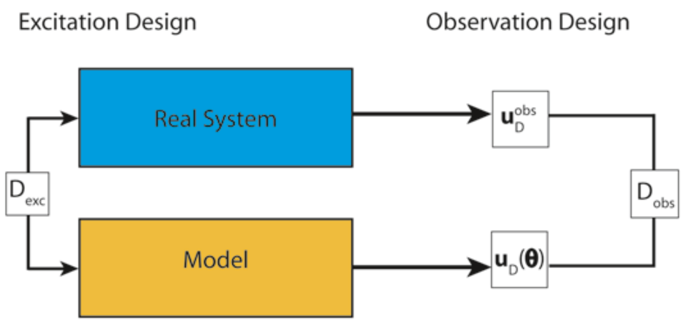


Fig. 11.1 Experimental designs for inversion

As shown in Fig. 11.1, once an experimental design D is executed either in the field or in the lab, we can use the constructed model to simulate the experimental process, namely, using D_{exc} as model inputs and using the model outputs prescribed by D_{obs} to compare with the observed system responses. The inversion approaches covered in previous chapters can then be used to calibrate the model structure and estimate model parameters.

Example 11.1 *Experimental design for identifying mass transport parameters*
In Example 2.2, the identification of mass transport parameters $\boldsymbol{\theta} = \{\alpha_L, R\}$ based on concentration measurements in a 1D flow field is considered. Here, α_L is the longitudinal dispersivity and R is the reactive rate coefficient. The 1D mass transport model has an analytical solution $C(C_{in}, v, \alpha_L, R, x, t)$, which is shown in Eq. (1.2.2). For this problem, the design variables include:

- Concentration of the inflow water, C_{in}
- Flow velocity, v
- Locations and times of measurements, $(x_1, t_1), (x_2, t_2), \cdots, (x_n, t_n)$

The measured concentration values and their corresponding model outputs can be represented, respectively, by

$$\mathbf{C}_D^{obs} = \{C(C_{in}, v, x_i, t_i) \mid i = 1, 2, \cdots, n\}, \tag{11.1.1}$$

and

$$\mathbf{C}_D(\boldsymbol{\theta}) = \{C(C_{in}, v, \alpha_L, R, x_i, t_i) \mid i = 1, 2, \cdots, n\}. \tag{11.1.2}$$

The inverse solution can be obtained by minimizing (see Sect. 3.2.2)

$$S_D(\boldsymbol{\theta}) = \left\|\mathbf{C}_D(\boldsymbol{\theta}) - \mathbf{C}_D^{obs}\right\|_2^2 + \lambda\left\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\right\|_2^2, \tag{11.1.3}$$

where $\boldsymbol{\theta}_0$ is prior guess and λ is the weighting coefficient.

Equation (11.1.3) implies that the inverse solution is dependent on design variables. An interesting problem is thus presented: How do we make an experimental design such that it minimizes the uncertainty of the identified parameters?

Example 11.2 *Observation design for contaminant source identification*
The release history of one or more point sources in an aquifer can be recovered with the concentration measurements taken from downstream of the source (Gorelick et al. 1983; Sun et al. 2006). In Example 2.2, it was shown that the unknown *m*-dimensional source strength parameter vector $\boldsymbol{\theta}$ is the solution of a linear model

$$\mathbf{A}_D\boldsymbol{\theta} = \mathbf{d}_D, \tag{11.1.4}$$

where $\mathbf{A}_D$ is an $n \times m$ matrix with elements $A_{D,ij} = C_j^0(x_i, t_i)$ and $\mathbf{d}_D$ is the *n*-dimensional observation vector with components $d_{D,i} = C_D^{obs}(x_i, t_i)$.

When the locations and times of measurement, $\left\{(x_i, t_i) \mid i = 1, 2, \cdots, n\right\}$, are considered as the design variables, an observation network design problem is thus presented: selecting locations and times of observations such that the uncertainty of the identified release history is minimized.

Example 11.3 *Design of a double-well injection–extraction tracer test*
This experiment uses two wells, an injection well and an extraction well, to collect data for inverse solution. After a steady-state flow field is established between the two wells, a tracer is added to the injection well, and the concentration of the plume is measured in the extraction well. The so-obtained concentration time series are then used for estimating the hydraulic conductivity *K*, longitudinal dispersivity α_L, and transverse dispersivity α_T. Decision variables of this experimental design include:

- Distance between the two wells
- Depths of the two wells
- Pumping and injection rates
- Tracer concentration of injected water
- Duration of the experiment
- Time to start sampling
- Sampling frequency

This problem is more complicated than the previous two examples because the model is governed by nonlinear PDEs and subject to model structure uncertainty. It needs to determine whether a 2D model or a 3D model should be used for design. In this case, a sensitivity-based sequential design process is needed. For a detailed discussion on this experiment, readers may refer to Mercer and Faust (1981), Sun (1996), and Dietrich and Leven (2006).

Example 11.4 *Design of environmental monitoring networks*
Automated environmental monitoring networks play an increasingly important role nowadays. A generic goal of experimental design for this case is to place sensors

in such a way that anomalies can be detected in a timely manner while operating costs are minimized. The sensor placement problem bears similarity to the classic warehouse siting or p-median problem in operations research (Hakimi 1964). The design variables include sensor locations and observation frequencies.

The common objectives of the design may include: Contaminant releases should be detected as early as possible, the spatial extent of an existing contaminant plume should be delineated as accurately as possible, and the cost of network construction and operation should be minimized. If a model has already been constructed, we should consider how the network can gather useful data for data assimilation to further improve the model. When a remediation alternative is being conducted, we should consider how the network can gather useful evidence for adjusting the alternative in time. An additional desirable performance attribute of monitoring networks is that it should be robust against the worst-case scenario. Because multiple design criteria are involved, there is not a single optimal design, but a set of Pareto optimal designs instead.

11.1.2 Formulation of Optimal Experimental Design

When the term OED is used, we must have one or more objectives or criteria to compare different designs and find the optimal one. Commonly used objectives in EWR modeling include, but are not limited to:

- Minimize the detection time
- Maximize spatial coverage of monitoring networks
- Minimize the model uncertainty
- Minimize the time period of model construction
- Minimize the risk of management decisions
- Minimize the cost of experimental expenses

Some objectives depend directly on the design variables D, such as the cost-effectiveness, but most objectives depend indirectly on the design variables through the estimated model parameters $(\boldsymbol{\theta})$ and the predicted system states $(\mathbf{u})$, such as the model reliability. In general, we can use $f(D)$ to represent an objective of design. Different designs may be obtained when different design objectives are considered. Effects of different objectives may be complementary or contradictory. For example, increasing the number of observations can decrease the uncertainty of the identified parameters and probably increase the reliability of model predictions, but at the expense of increased experimental costs. Therefore, it is natural to consider OED in the framework of multiobjective decision making (e.g., Hsu and Yeh 1989; Knopman and Voss 1989; Meyer et al. 1994; Reed and Minsker 2004; Sun et al. 2013; Sun 1994; Wagner 1995). In this framework, OED is defined as a solution of the following multiobjective optimization (MOO) problem:

$$\begin{aligned}&\text{Minimize}\left\{f_1(D), f_2(D), \cdots, f_K(D)\right\}\\&\text{subject to } g_l(D) \le 0\,(l = 1, 2, \cdots, L),\end{aligned} \tag{11.1.5}$$

where $\{f_i(D) \mid i = 1, 2, \cdots, K\}$ are K objectives and $\{g_l(D) \mid l = 1, 2, \cdots, L\}$ are L constraints of the design, respectively. In practice, problem (11.1.5) may involve linear or nonlinear objectives and continuous or discrete design decision variables with high dimensionality. As a result, its solution is very challenging. Multiobjective evolutionary algorithms (MOEA) learned in Sect. 3.6 and their modifications are recently used to solve the multiobjective design problems (Mayer et al. 2002). In recent years, several MOEA algorithms have been developed for solving multiobjective network design problems related to groundwater quality monitoring (e.g., Kollat and Reed 2006; Reed et al. 2013)

This chapter introduces the basic theory and methods of OED for model construction, especially for distributed parameter systems. We will start from considering only a single objective (e.g., minimizing the uncertainty of inverse solutions), and then adding more objectives (e.g., cost-effectiveness, accuracy of model prediction, and the reliability of decision making).

11.2 Experimental Design for Linear Model Inversion

11.2.1 Design with Uniform Prior Distribution

Linear model inversion (or linear regression) was considered in the Bayesian framework in Chap. 4. The observation equation of a linear model, $\mathbf{u} = \mathbf{A}\boldsymbol{\theta}$, is given by

$$\mathbf{u}_D^{obs} = \mathbf{A}_D\boldsymbol{\theta} + \mathbf{e}_D, \tag{11.2.1}$$

where $\mathbf{u}_D^{obs}$ is the observation vector corresponding to a design D, $\mathbf{A}_D$ is the design matrix depending on design variables, and $\mathbf{e}_D$ is the associated observation error vector. In Sect. 4.2.2, we showed that if the probability distribution of $\mathbf{e}_D$ is Gaussian and the prior probability distribution of $\boldsymbol{\theta}$ is uniform in a given region, then the best unbiased estimate of $\boldsymbol{\theta}$ is given by

$$\hat{\boldsymbol{\theta}}_D = [(\mathbf{A}_D^T\mathbf{C}_D^{-1}\mathbf{A}_D)^{-1}\mathbf{A}_D^T\mathbf{C}_D^{-1}]\mathbf{u}_D^{obs}, \tag{11.2.2}$$

and the covariance matrix of estimation is

$$\mathrm{Cov}(\hat{\boldsymbol{\theta}}_D) = (\mathbf{A}_D^T\mathbf{C}_D^{-1}\mathbf{A}_D)^{-1}, \tag{11.2.3}$$

where $\mathbf{C}_D$ is the covariance matrix of the observation error vector. Further, if the observation errors are i. i. d., the covariance becomes $\mathbf{C}_D = \sigma^2\mathbf{I}$ and (11.2.3) becomes

$$\mathrm{Cov}(\hat{\boldsymbol{\theta}}_D) = \sigma^2(\mathbf{A}_D^T\mathbf{A}_D)^{-1}. \tag{11.2.4}$$

The covariance $\mathrm{Cov}(\hat{\boldsymbol{\theta}}_D)$ on the right-hand side of (11.2.3) or (11.2.4) measures the uncertainty of estimation and, thus, can be used as a measure of performance of a design. Equation. (11.2.2) shows that $\hat{\boldsymbol{\theta}}_D$ can be obtained only after a design is conducted, data are collected, and the inverse problem is solved. But (11.2.3) or (11.2.4) shows that $\mathrm{Cov}(\hat{\boldsymbol{\theta}}_D)$ is independent of observation data and can be calculated for each candidate design during the design stage. As a result, the performance of a design can be assessed before it is actually conducted. That is why the problem of OED makes sense. Let us discuss the OED problem for linear regression in detail below.

11.2.1.1 Information Matrix

In information theory, when the observation error is normally distributed, the inverse of the covariance matrix $\mathrm{Cov}(\hat{\boldsymbol{\theta}}_D)$ is called the *Fisher Information Matrix* (FIM) and denoted by

$$\mathbf{M}(D) = \mathbf{A}_D^T \mathbf{C}_D^{-1} \mathbf{A}_D. \tag{11.2.5}$$

When $\mathbf{C}_D = \sigma^2 \mathbf{I}$, we have $\mathbf{M}(D) = \sigma^{-2} \mathbf{A}_D^T \mathbf{A}_D$. With FIM, the OED of minimizing $\mathrm{Cov}(\hat{\boldsymbol{\theta}}_D)$ becomes a problem of maximizing $\mathbf{M}(D)$. In fact, FIM measures the information content carried by observations of a design *D* for the estimation of unknown parameters. The "size" of a matrix is measured by a monotonically increased scalar function $\phi(\cdot)$ for defining the norm of a matrix (see below). A design *D* having a larger value of $\phi[\mathbf{M}(D)]$ means that the designed observations carry more information content for parameter estimation and, thus, produce more reliable estimation results.

11.2.1.2 Optimality Criteria

For linear model inversion, a general form of OED is given by

$$D^* = \arg\max_D \phi[\mathbf{M}(D)], D \in \{D_{ad}\}, \tag{11.2.6}$$

where $\{D_{ad}\}$ is a set of admissible experiments. Different optimality criteria are derived by selecting different definitions of ϕ. Some of them are given below.

D-optimality seeks to maximize the log-determinant of $\mathbf{M}(D)$ by defining ϕ in (11.2.6) as

$$\phi[\mathbf{M}(D)] = \log\det[\mathbf{M}(D)]. \tag{11.2.7}$$

Or equivalently, we can minimize $\log\det[\mathbf{M}^{-1}(D)]$. In Sect. 4.2.1, we learned how to measure the uncertainty of a normally distributed parameter vector. According to Eq. (4.1.10), D-optimality maximizes the information content for parameter estimation or minimizes the uncertainty of the estimated parameters. Therefore, D-optimality is the most popular criterion of OED. A geometric interpretation of the D-optimality criterion is that it seeks to minimize the volume of the confidence ellipsoid in the design space.

A-optimality seeks to minimize the trace of $\mathbf{M}^{-1}(D)$. Thus, ϕ is defined by

$$\phi[\mathbf{M}(D)] = -\text{trace}[\mathbf{M}^{-1}(D)]. \tag{11.2.8}$$

The trace of a matrix is the sum of all elements on its diagonal. This means that A-optimality minimizes the average variance of the estimated parameters. In geometric terms, A-optimality seeks to minimize the average axis length of the confidence ellipsoid. Unlike D-optimality, A-optimality ignores the correlations among the identified parameters.

E-optimality seeks to maximize the minimum eigenvalues of $\mathbf{M}(D)$. Thus in E-optimality ϕ is defined by

$$\phi[\mathbf{M}(D)] = \lambda_{\min}[\mathbf{M}(D)]. \tag{11.2.9}$$

We see that E-optimality is equivalent to minimizing the largest variance of the estimated parameters. In geometric terms, it seeks to minimize the largest axis length of the confidence ellipsoid. There are effective algorithms for E-optimality that do not require the differentiability of the objective function.

T-optimality seeks to maximize the trace of $\mathbf{M}(D)$ and the corresponding ϕ is

$$\phi[\mathbf{M}(D)] = \text{trace}[\mathbf{M}(D)]. \tag{11.2.10}$$

T-optimality is an alternative to A-optimality, but easier to compute.

C-optimality seeks to minimize the variance of a linear combination $g = \mathbf{c}^T \boldsymbol{\theta}$, where **c** is an *m*-dimensional vector. Because

$$\min Var(\mathbf{c}^T \hat{\boldsymbol{\theta}}_D) = \min \mathbf{c}^T [\mathbf{M}^{-1}(D)]\mathbf{c},$$

C-optimality thus seeks to maximize

$$\phi[\mathbf{M}(D)] = -\mathbf{c}^T [\mathbf{M}^{-1}(D)]\mathbf{c}. \tag{11.2.11}$$

For detailed discussions on these and more alphabetic optimality criteria, readers may refer to Silvey (1980), Pázman (1986), Atkinson et al. (2007), Pronzato (2008), and Franceschini and Macchietto (2008).

11.2.1.3 Algorithms of OED for Linear Model Inversion

The OED problem (11.2.6) requires one to search for the best design among a very large number of possible designs. For instance, in an observation network design problem, design variables D may consist of the number, locations, and/or times of observations. The number of candidates grows exponentially as the total number of sensors/samples (i.e., rows in $\mathbf{A}$) increases. This subset selection problem belongs to a type of combinatorial optimization problems, which are NP-hard and not amendable for exact solutions. For practical purposes, we are more interested in near-optimal solutions that can be obtained in meaningful computational time, but have guaranteed performance with respect to the chosen objective function. Classic approaches to the OED problem can be classified in three categories: convex optimization, greedy search methods, and heuristic methods. For instance, a solution of the OED problem (11.2.6) may be obtained by a local optimization algorithm (Sect. 2.3) or by a global optimization/search algorithm (Sect. 3.6), depending on whether the underlying objective function $F(D) = \phi[\mathbf{M}(D)]$ is convex. In recent years, there is a renewed interest in using greedy methods to obtain near-optimal solutions (Krause and Guestrin 2007; Shamaiah et al. 2010; Krause et al. 2008). We will provide a brief survey of methods at the end of this subsection. Generally speaking, an OED algorithm consists of the following steps:

1. Generate an initial design with K observations, for which the locations and/or times of observation points ($\mathbf{x}_K$) are predetermined (e.g., to comply with regulations). Alternatively, the initial set may be empty.
2. Optimize $\mathbf{x}_K$ according to the selected optimality criterion. For example, when D-optimality is used, the optimal $\mathbf{x}_K^*$ are determined by

$$\mathbf{x}_K^* = \arg\max_{\mathbf{x}_K} \log[\det\mathbf{M}(\mathbf{x}_K)]. \tag{11.2.12}$$

3. Check if $\log[\det\mathbf{M}(\mathbf{x}_K^*)]$ is large enough such that the uncertainty of the estimation is less than a predetermined value or such that the number of samples needed is met. If yes, stop.
4. Otherwise, increase the number of observations K and return to step 2.

In order to reduce the computational effort required for solving (11.2.12), $\mathbf{x}_K$ can be regarded as K movable points in the given spatial/temporal region. In each iteration, all or a part of them are moved to new locations to increase the value of $\log[\det\mathbf{M}(\mathbf{x}_K)]$ as much as possible based on sensitivity analysis. This process is very similar to the adaptive model structure optimization algorithm described in Sect. 7.2, where the objective function (fitting residual) is minimized gradually by adding new Voronoi points and moving all or a part of them to their optimum locations. Network structure optimization is a type of structure optimization problem and is, thus, combinatorial in nature.

Similar design problems have been studied in environmental sensing. Krause and Guestrin (2007) showed that by gradually adding sensors that increase the sensing quality of the network the most, a near-optimal solution can be obtained and, thus, more efficient greedy search algorithm like the one we delineated in the above can be used; the authors referred to problems amendable to such treatment as *submodular* problems.

Let $\mathcal{S}$ be a finite set. For any two subsets $A \subseteq B \subset \mathcal{S}$ and an element $i \in \mathcal{S} \setminus B$, a function F is submodular if it satisfies

$$F(A+i) - F(A) \geq F(B+i) - F(B). \tag{11.2.13}$$

An example of submodular functions is the mutual information that is defined as the difference between two entropy measures (e.g., Shannon entropy)

$$F(A) = \mathcal{H}(D_U) - \mathcal{H}(D_U \middle| D_A),$$

where D_A and D_U are two sets of decision variables (e.g., potential monitoring well locations) and $F(A)$ measures the decrease in uncertainty in set U (e.g., unobserved locations) because of the observations of set A (e.g., observed locations). The submodular function is attractive to be used as a cost function for greedy search. It can be shown that the greedy maximization of nondecreasing submodular functions is near-optimal and it guarantees a constant-factor approximation $(1-1/e)$ of the optimal solution in polynomial time (Nemhauser et al. 1978). Substituting the value $e = 2.718$, we see that the greedy search gives an approximate solution that is at least 63 % of the optimal solution. Examples of submodular functions for environmental sensing problems are given in Krause et al. (2008).

OED algorithms have been studied for many years. Variations and modifications of the above algorithm, as well as codes can be found, for example, in Atkinson et al. (2007).

Example 11.5 *Optimal design for the source identification problem*
Let us revisit the source identification problem considered in Example 2.3, in which a linear regression problem was formulated to identify the unknown source strengths in seven periods. As already mentioned in Example 11.2, the goal of our OED problem is to determine the best sampling locations and times. The design variables consist of the ranges of potential observation well locations $\{1, 2, \ldots, 9\}$ [L] and observation times $t = 1$ to 16 [T] in 0.5 [T] intervals. The total number of possible samples is thus 279. We first generate a 279×7 design matrix **G** using the unit-pulse method described in Example 2.3. Assuming that we only have a budget for collecting a small number of samples, we want to maximize the D-optimality criterion to find a subset of rows from **G.** It has been shown that the objective function (11.17) is monotone submodular (Shamaiah et al. 2010) and, thus, we can apply a greedy search algorithm. Starting from an empty set, we gradually select rows from **G** (without replacement) to maximize the D-optimality of the subset, until the number of desired samples is reached.

Figures 11.2a and b show the optimal designs for 10- and 30-point sampling plans, respectively. For reference, we also show the actual concentration

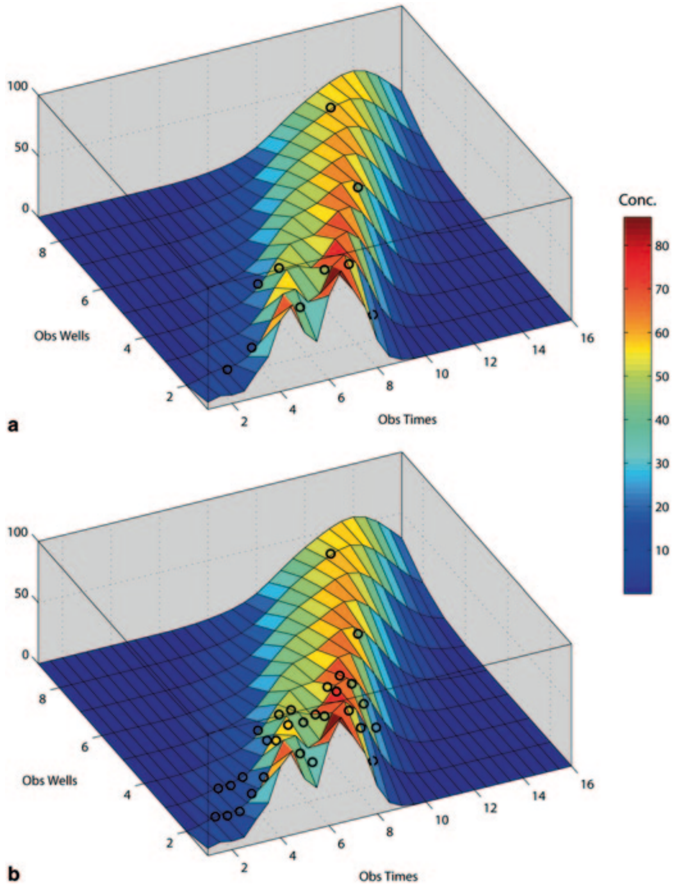


Fig. 11.2 Contour of the concentration field as a function of observation well locations and times. The *open circles* are selected sample points for **a** 10- and **b** 30-point sampling design

contour generated using the true source vector, which exhibits a bimodal shape. In Fig. 11.2a, it is interesting to notice that the first few sampling points selected are late-time samples collected from the far field, although subsequently selected points are closer to the source. This is partly because the far-field late-time samples provide more information about the entire plume, but the near-field samples contain more information about the shape of the plume (therefore, source characteristics) before the plume is smoothed out. The matrix condition numbers of the selected design matrix are 2.56 and 2.66, respectively, for the two cases.

The example also shows that it is useful to have some knowledge of the topography of the design space. When the dimension of the design space is low, the easiest way is to visualize it through pairwise contour map as we have done in Fig. 11.2.

11.2.1.4 Choosing a Criterion

Note that OED is nonunique. Different optimality criteria generally lead to different designs, although their results may be equivalent under certain conditions (Atkinson et al. 2007). Each criterion has advantages but also disadvantages, and the best choice is often problem dependent (Franceschini and Macchietto 2008). For example, Nordqvist (2000) shows that D-optimality is the best one for identifying mass transport parameters, E-optimality can give similar results with less computational effort, but T-optimality produces a significantly different design. Telen et al. (2012) noted that the widely used D-optimality is not always the best tool for different case studies; different criteria may even be in conflict. Ranieri et al. (2014) showed that the optimality criteria maximizing one objective (e.g., maximal entropy) may not yield the best design for other objectives (e.g., mean squared error, MSE). Since there is no single optimal criterion, a new approach is thus presented in the framework of multiobjective optimization, in which several criteria are combined and the final design is found by a trade-off process.

11.2.2 Design with Gaussian Prior Distribution

We showed in Sect. 4.2.3 that when the prior probability distribution of $\boldsymbol{\theta}$ is Gaussian

$$p_0(\boldsymbol{\theta}) \sim N(\boldsymbol{\theta}_0, \mathbf{C}_P),$$

the parameter estimated from (11.2.1) is

$$\hat{\boldsymbol{\theta}}_D = (\mathbf{A}_D^T \mathbf{C}_D^{-1} \mathbf{A}_D + \mathbf{C}_P^{-1})^{-1} (\mathbf{A}_D^T \mathbf{C}_D^{-1} \mathbf{u}_D^{obs} + \mathbf{C}_P^{-1} \boldsymbol{\theta}_0), \tag{11.2.14}$$

and the covariance matrix of estimation is given by

$$\mathrm{Cov}(\hat{\boldsymbol{\theta}}_D) = (\mathbf{A}_D^T \mathbf{C}_D^{-1} \mathbf{A}_D + \mathbf{C}_P^{-1})^{-1} = [\mathbf{M}(D) + \mathbf{C}_P^{-1}]^{-1}, \tag{11.2.15}$$

where the information matrix $\mathbf{M}(D) = \mathbf{A}_D^T \mathbf{C}_D^{-1} \mathbf{A}_D$ (same as in (11.2.5)) measures the information content provided by data and the covariance $\mathbf{C}_P$ measures the uncertainty of prior information. To decrease the uncertainty of the estimated parameter, we can either increase the norm of $\mathbf{M}(D)$ or decrease the norm of $\mathbf{C}_P$. The OED definition (11.2.6) now is modified to

$$D_B^* = \arg\max_D \phi[\mathbf{M}(D) + \mathbf{C}_P^{-1}], D \in \{D_{ad}\}. \tag{11.2.16}$$

This is often called the *Bayesian OED*, in which prior information is incorporated explicitly into the objective function. In particular, the *Bayesian D-optimality* and *Bayesian A-optimality criteria* are given, respectively, by

$$D_{BD}^{*} = \arg\max_{D} \log\det[\mathbf{M}(D) + \mathbf{C}_P^{-1}], D \in \left\{D_{ad}\right\}, \tag{11.2.17}$$

and

$$D_{BA}^{*} = \arg\min_{D} \text{trace}[\mathbf{M}(D) + \mathbf{C}_P^{-1}]^{-1}, D \in \left\{D_{ad}\right\}. \tag{11.2.18}$$

In practice, the prior covariance matrix $\mathbf{C}_P$ cannot be known accurately. Bayesian OED is often used in such a situation that an inversion process is already completed with the existing data and the covariance matrix of the posterior distribution is obtained, but the uncertainty of estimation is unsatisfied; and a new design is thus required to increase the reliability of the estimated parameters. In this case, the posterior distribution of the previous estimation (if it is close to normal) can be used as the prior distribution for the new design. From the point of view of regularization, $\mathbf{C}_P^{-1}$ in (11.2.16) may be replaced by a general term $\alpha\mathbf{R}$, where α is a weighting coefficient and matrix $\mathbf{R}$ describes the reliability of the available prior information.

11.3 Experimental Design for Nonlinear Model Inversion

11.3.1 Linearization

Although the theory and methods of OED for linear model inversion have been well established, OED for nonlinear model inversion is still a very challenging problem. As shown below, the OED criteria for nonlinear model inversion depend on the parameter vector, which is unknown during the design stage. As a result, the uncertainty of estimation cannot be known during the design stage.

11.3.1.1 Linearization

The observation equation based on a design D for inversion of a nonlinear model, $u = u(\boldsymbol{\theta})$, is given by

$$\mathbf{u}_D^{obs} = \mathbf{u}_D(\boldsymbol{\theta}) + \mathbf{e}_D. \tag{11.3.1}$$

After the model is linearized around the unknown parameter vector $\boldsymbol{\theta}$, this equation can be replaced approximately by (Sect. 2.3)

$$\mathbf{u}_D^{obs} = \mathbf{J}_D\boldsymbol{\theta} + \mathbf{r}_D, \tag{11.3.2}$$

where $\mathbf{J}_D(\boldsymbol{\theta}) = [\partial\mathbf{u}_D / \partial\boldsymbol{\theta}]$ is the sensitivity matrix and $\mathbf{r}_D = \mathbf{u}_D(\boldsymbol{\theta}) - \mathbf{J}_D\boldsymbol{\theta} + \mathbf{e}_D$. The above equation has the same form as (11.2.1) if $\mathbf{A}_D$ is replaced by $\mathbf{J}_D$ and $\mathbf{e}_D$ is replaced by $\mathbf{r}_D$. When the linearization error can be ignored, we have $\mathbf{r}_D = \mathbf{e}_D$.

The covariance of estimation in (11.2.3) now becomes

$$\mathrm{Cov}(\hat{\boldsymbol{\theta}}_D) = [\mathbf{J}_D^T(\boldsymbol{\theta})\mathbf{C}_D^{-1}\mathbf{J}_D(\boldsymbol{\theta})]^{-1}. \tag{11.3.3}$$

Its inverse matrix defines the FIM of the nonlinear model

$$\mathbf{M}(D,\boldsymbol{\theta}) = \mathbf{J}_D^{\mathbf{T}}(\boldsymbol{\theta})\mathbf{C}_D^{-1}\mathbf{J}_D(\boldsymbol{\theta}), \tag{11.3.4}$$

which is dependent on the unknown parameter $\boldsymbol{\theta}$. When $\mathbf{C}_D = \sigma^2\mathbf{I}$, we have

$$\mathbf{M}(D,\boldsymbol{\theta}) = \sigma^{-2}\mathbf{J}_D^T(\boldsymbol{\theta})\mathbf{J}_D(\boldsymbol{\theta}). \tag{11.3.5}$$

Minimizing the uncertainty of the estimated parameter vector is equivalent to maximizing a norm of FIM. The OED defined in (11.2.6) can now be rewritten as

$$D^*(\boldsymbol{\theta}) = \arg\max_D \phi[\mathbf{M}(D,\boldsymbol{\theta})], D \in \{D_{ad}\}. \tag{11.3.6}$$

After linearization, we can apply the same optimality criteria for linear model inversion to nonlinear model inversion after replacing $\mathbf{A}_D$ by $\mathbf{J}_D(\boldsymbol{\theta})$. For example, the D-optimality is defined by $\phi[\mathbf{M}(D,\boldsymbol{\theta})] = \log\det[\mathbf{M}(D,\boldsymbol{\theta})]$ and the T-optimality is defined by $\phi[\mathbf{M}(D,\boldsymbol{\theta})] = \mathrm{trace}[\mathbf{M}(D,\boldsymbol{\theta})]$. T-optimality is also called the *sensitivity criterion* of design because the trace of FIM is the sum of squared sensitivities of observations to the identified parameters at the observation points. In fact, from (11.3.5), we have

$$\mathrm{trace}[\mathbf{M}(D,\boldsymbol{\theta})] = \frac{1}{\sigma^2}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\frac{\partial u_{D,i}}{\partial \theta_j}\right)^2, \tag{11.3.7}$$

where $\{u_{D,i} \mid i = 1,2,\cdots,n\}$ are components of $\mathbf{u}_D(\boldsymbol{\theta})$ and $\{\theta_j \mid j = 1,2,\cdots,m\}$ are components of $\boldsymbol{\theta}$, respectively. This criterion tells us that the observation points should be placed at observation locations that are the most sensitive to the identified parameters.

11.3.1.2 Local Optimal Design

When an initial guess $\boldsymbol{\theta}_0$ is available from prior information or previous estimation, we can use it to replace the unknown parameter in (11.3.6) to make a design. Such a design is called a local OED because it is optimal only when the true parameter is close to $\boldsymbol{\theta}_0$. Toward this end, we can sample the admissible region randomly or based on a prior distribution to find how sensitive a local optimal design is with respect to the parameter variability (the Monte Carlo method). Surveys of local optimal design approaches are provided in Ford et al. (1989); Chaloner and Verdinelli (1995), and Papalambros and Wilde (2000).

11.3.1.3 Sequential Design

When $\boldsymbol{\theta}_0$ is not close to $\boldsymbol{\theta}$ and the FIM is sensitive to $\boldsymbol{\theta}_0$, a local OED may be far from the optimal one (Ford et al. 1989). In this case, a sequential design process can be used if it is feasible (Atkinson et al. 2007; Silvey 1980). This process consists of the following steps:

1. Use an initial guess $\boldsymbol{\theta}_0$ to find a local OED.
2. Conduct the design and collect observation data accordingly.
3. Solve the inverse problem to obtain an updated parameter vector $\boldsymbol{\theta}_1$.
4. If $\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_0\|$ is less than a predetermined tolerance, stop; otherwise, replace $\boldsymbol{\theta}_0$ by $\boldsymbol{\theta}_1$ and return to step 1.

The sequential design process requires conducting a sequence of experiments in the field that may not be feasible because of the high cost and long time period. In EWR fields, sequential design is used as a retrospective design process to modify the observation system and update the simulation model alternatively. The concept is similar to that behind the Kalman filter (Chap. 9) and active learning theory. In the latter, a learning algorithm is allowed to choose data or make queries from which it learns such that the algorithm will perform better with less training (Settles 2010). Sequential design is a "work assumption" used in the study of network design and remediation design.

When making a prospective design, however, we prefer to make the design more robust or less dependent on the unknown parameter vector. As shown in the following subsections, robust design approaches are developed in both statistical and deterministic frameworks.

11.3.2 Bayesian Experimental Design

11.3.2.1 Expectation-Based Experimental Design

During the design stage, all $\boldsymbol{\theta}$ in the admissible region Θ_{ad} (determined by prior information) have certain probability to be the true parameter. Thus, for a nonlinear model, we have a set of objective functions $\{\phi[\mathbf{M}(D,\boldsymbol{\theta})] \mid \boldsymbol{\theta} \in \Theta_{ad}\}$. If the probability density function (PDF) $p(\boldsymbol{\theta})$ of $\boldsymbol{\theta}$ is available, we can use it as a weight function to find the expectation (probability average) of the function set,

$$E_{\boldsymbol{\theta}}\left\{\phi[\mathbf{M}(D,\boldsymbol{\theta})]\right\} = \int_{\Theta_{ad}} \phi[\mathbf{M}(D,\boldsymbol{\theta})]\mathbf{p}(\boldsymbol{\theta})d\boldsymbol{\theta}. \qquad (11.3.8)$$

We have mentioned that, for nonlinear model inversion, the uncertainty of the estimation cannot be known before the inverse problem is solved. But with (11.3.8), we can find the expected value of uncertainty during the design stage. In other words, the dependence on the unknown parameter is marginalized out in (11.3.8). The optimal design under probability average is defined by

$$D_E^* = \arg\max_D E_{\boldsymbol{\theta}} \left\{ \phi[\mathbf{M}(D,\boldsymbol{\theta})] \right\}, D \in \left\{ D_{ad} \right\}. \tag{11.3.9}$$

In particular, the D-optimality under probability average is

$$D_{ED}^* = \arg\max_D E_{\boldsymbol{\theta}} \left\{ \log\det[(D,\boldsymbol{\theta})] \right\}, D \in \left\{ D_{ad} \right\}. \tag{11.3.10}$$

Other optimality criteria under expectation can be derived similarly by choosing different $\phi(\cdot)$. Their performances, however, are compared only for simple nonlinear models with a few unknown scalar parameters (Walter and Pronzato 1997; Foo et al. 2012; Uciński 2005).

Note that design (11.3.9) is independent of any individual $\boldsymbol{\theta}$ and, thus, is a robust design. But, on the other hand, it is not the optimal one for all $\boldsymbol{\theta}$. Because the true probability distribution $p(\boldsymbol{\theta})$ is unknown during the design stage, we have to use a prior distribution to replace it in the calculation of (11.3.8). If the prior distribution is not concentrated to the true parameter vector, design (11.3.9) may be far from the optimal one. In practice, the Bayesian experimental design is often used as part of a closed-loop or sequential design process: After an inverse solution and its posterior distribution are obtained from the existing data, we want to design an experiment to collect new data to decrease the uncertainty of the estimated parameters through data assimilation (Chap. 9). In this case, the posterior distribution can be used as $p(\boldsymbol{\theta})$ in (11.3.8) for the design of the next step. This means that the expectation-based design method is more suitable to be used in a retrospective manner to make the new design robust.

Major steps of solving problem (11.3.9) include:

1. Select an optimality criterion $\phi(\cdot)$.
2. Determine a probability distribution $p(\boldsymbol{\theta})$ as described above.
3. Solve the optimization problem (11.3.9). Assume that the design currently searched by the optimization method is D_0.
4. In order to find the value of objective function at D_0 (i.e., $E_{\boldsymbol{\theta}} \left\{ \phi[\mathbf{M}(D_0,\boldsymbol{\theta})] \right\}$), a Markov Chain Monte Carlo (MCMC) numerical integration method (Sect. 4.3) is used to calculate (11.3.8) for design D_0. This step includes sampling L parameter vectors $\left\{ \boldsymbol{\theta}_l \mid l = 1,2,\cdots,L \right\}$ and performing the following substeps:

   For $l = 1,2,\cdots,L,$ do

   – Use a model differentiation method (Chap. 5) to calculate the sensitivity matrix $\mathbf{J}_{D_0}(\boldsymbol{\theta})$ evaluated at $\boldsymbol{\theta}_l$, which is denoted by $\mathbf{J}(D_0,\boldsymbol{\theta}_l)$. The forward problem must be solved many times during this process to generate the model outputs corresponding to design D_0.
   – Calculate $\mathbf{M}(D_0,\boldsymbol{\theta}_l) = \mathbf{J}^T(D_0,\boldsymbol{\theta}_l)\mathbf{C}_D^{-1}\mathbf{J}_D(D_0,\boldsymbol{\theta}_l)$.
   – Calculate the value $\mathbf{M}(D_0,\boldsymbol{\theta}_l)p(\boldsymbol{\theta}_l)$.

   The expectation $E_{\boldsymbol{\theta}} \left\{ \phi[\mathbf{M}(D_0,\boldsymbol{\theta})] \right\}$ is obtained from (11.3.8).

5. Determine the convergence of the iteration process for solving (11.3.9). If yes, stop and use D_0 as D_E^*. Otherwise, find the next candidate design D_1 according to the optimization algorithm, replace D_0 by D_1, and return to step 4.

In this process, an m-dimensional integration in (11.3.8) must be calculated repeatedly that requires the calculation of sensitivity matrices for each candidate design and multiple samples of the unknown parameter vector. Although the MCMC sampling methods introduced in Sect. 4.3 can be used here as a useful tool for numerical integration, the total computational effort becomes infeasible when the number of the unknown parameters is large or the number of the design variables is large.

11.3.2.2 General Bayesian Experimental Design

In the general form of the Bayesian experimental design, the usefulness of an experiment for a given objective is described by a utility function $U(D, \boldsymbol{\theta}, \mathbf{y})$, where D is the design variable vector, $\boldsymbol{\theta}$ is the unknown parameter vector, and $\mathbf{y}$ is the vector of observation data. *Utility*, a concept originated from economics, provides a quantitative measure of the satisfaction of a customer after he/she receives certain goods or services. In the current context, the utility function quantifies the utility of a design from the perspective of the designer. Objectives of the design could be parameter estimation, state prediction, system management, and model discrimination. We see that the design for a linear model inversion with Gaussian prior distribution (Sect. 11.2.2) and the abovementioned expectation-based design are special cases of this general framework. According to Lindley (1972), a good way to design experiments is to select a design that maximizes the *expected utility*, which has the following general form

$$E[U(D)] = \int_Y \int_{\Theta_{ad}} U(D, \boldsymbol{\theta}, \mathbf{y}) \mathbf{p}(\boldsymbol{\theta}, \mathbf{y} \mid D) d\boldsymbol{\theta} d\mathbf{y}. \tag{11.3.11}$$

The optimal design seeks to maximize the expected utility

$$D_B^* = \arg\max_D E[U(D)], D \in \{D_{ad}\}. \tag{11.3.12}$$

When $U(D, \boldsymbol{\theta}, \mathbf{y}) = \phi[\mathbf{M}(D, \boldsymbol{\theta})]$, (11.3.11) reduces to (11.3.8), and the Bayesian design (11.3.12) becomes D_E^* in (11.3.9). When $U(D, \boldsymbol{\theta}, \mathbf{y}) = \log[p(\boldsymbol{\theta} \mid D)]$, (11.3.11) becomes the information content defined in Sect. 4.1.2. Furthermore, as shown in Eq. (4.10), when $p(\boldsymbol{\theta} \mid D)$ is Gaussian, we have

$$E[U(D)] = -\frac{m}{2}(1 + \log 2\pi) - \frac{1}{2} \log \det[\mathrm{Cov}(\hat{\boldsymbol{\theta}}_D)]. \tag{11.3.13}$$

According to (11.2.15), in this case, design (11.3.12) becomes the Bayesian D-optimal design D_{BD}^* in (11.2.27).

When $U(D,\boldsymbol{\theta},\mathbf{y}) = -(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_D)^T \mathbf{A}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_D)$, here $\mathbf{A}$ is an asymmetric non-negative definite matrix, and $p(\boldsymbol{\theta} \mid D)$ is Gaussian, integration of (11.3.11) gives $E[U(D)] = -\text{trace}[\mathbf{A}\text{Cov}(\hat{\boldsymbol{\theta}}_D)]$. In this case, when $\mathbf{A} = \mathbf{I}$, design (11.3.12) becomes the Bayesian A-optimal design D_{BA}^* in (11.2.18).

When $\mathbf{A} = \mathbf{cc}^T$, we have $E[U(D)] = -\mathbf{c}^T \text{Cov}(\hat{\boldsymbol{\theta}}_\mathbf{D})\mathbf{c}$ and design (11.3.12) becomes the Bayesian C-optimal design

$$D_{BC}^* = \arg\max_D \left\{ -\mathbf{c}^T [\mathbf{M}(D) + \mathbf{C}_P^{-1}]^{-1} \mathbf{c} \right\},\ D \in \left\{ D_{ad} \right\}. \tag{11.3.14}$$

For a uniform prior distribution, (11.3.14) becomes the *C*-optimal design in (11.2.11).

In summary, all criteria of OED, for either linear model inversion or nonlinear model inversion, can be derived in the framework of the Bayesian experimental design (Chaloner and Verdinelli 1995). The major steps used for solving problems (11.3.9) and (11.3.12) are the same, but the latter may become more complicated. A major challenge associated with Bayesian experimental design is the lack of effective algorithms. Reviews of recent developments in this area, including metamodeling-based design techniques for reducing computational demand, can be found in Wang and Shan (2007), Shan and Wang (2010), Huan and Marzouk (2013), and Brochu et al. (2010). Another problem associated with the Bayesian experimental design is its reliability. When prior information assigns a small probability to the true parameter, an inferior design will be generated and the data collected according to such a design will make the inverse solution unreliable. For applications having low tolerance to such risks, the designer may turn to the following max-min robust design approach.

11.3.3 Robust Experimental Design

Robust experimental design for nonlinear model inversion is an acceptable design for the identification of all parameters in a given region. It is based on the philosophy of optimizing against the worst-case scenario. When there are uncertainties in model structure and model parameters, the concept of robustness is often used for experimental design and system control (Steinberg and Hunter 1984; Pronzato and Walter 1988; Rojas et al. 2007; Dror and Steinberg 2006). To date, studies and applications of robust experimental design are still very limited in the EWR fields (Uciński 2005; Sun and Yeh 2007).

The max-min robust experimental design is defined by

$$D_R^* = \arg\max_D \min_{\boldsymbol{\theta}} \phi[\mathbf{M}(D,\boldsymbol{\theta})], D \in \{D_d\} \text{ and } \boldsymbol{\theta} \in \{\Theta_{ad}\}, \tag{11.3.15}$$

where $\{\Theta_{ad}\}$ is an admissible region of θ determined by prior information. The above max-min problem can be rewritten as

$$D_R^* = \arg\max_D \phi[\mathbf{M}(D,\tilde{\boldsymbol{\theta}}_D)], D \in \{D_{ad}\} \tag{11.3.16}$$

and

$$\tilde{\boldsymbol{\theta}}_D = \arg\min_{\boldsymbol{\theta}} \phi[\mathbf{M}(D,\boldsymbol{\theta})],\ \boldsymbol{\theta} \in \{\Theta_{ad}\}, \tag{11.3.17}$$

where $\tilde{\boldsymbol{\theta}}_D$ is called the *worst-case parameter* (WCP) associated with design *D*. From (11.3.17), $\tilde{\boldsymbol{\theta}}_D$ has the maximum uncertainty compared with other parameters in the admissible region; in other words, it is the most difficult one to be identified for design *D*. From (11.3.16), robust design is the optimal design for the WCP. The concept of robust experimental design is now clear: The exact uncertainty of estimation is unknown during the design stage, but we can make a conservative design to minimize its upper bound.

Although the concept of robust experimental design is relatively straightforward, the solution of the resulting max-min problem is not easy, unless the WCP is design-independent (in that case, the robust design reduces to a local design). In the general case, when a global optimization algorithm is used to solve the maximization problem (11.3.16), for each candidate design D the minimization problem (11.3.17) must be solved to find the value of objective function $F(D) = \phi[\mathbf{M}(D,\tilde{\boldsymbol{\theta}}_D)]$. Like the min-min problem discussed in Chap. 7, to make this iterative process computationally feasible, $\{\Theta_{ad}\}$ and $\{D_{ad}\}$ must be represented by finite sets, and the computation effort of solving the inner minimization problem (11.3.17) must be significantly reduced. Prozato and Walter (1988) suggested the following relaxation process:

1. Choose an initial parameter vector $\boldsymbol{\theta}_1$, define the first set of representative values $Z_1 = \{\boldsymbol{\theta}_1\}$, and set $k = 1$.
2. Solving the current max-min problem

$$D_k = \arg\max_D \min_{\boldsymbol{\theta}} \phi[\mathbf{M}(D,\boldsymbol{\theta})], D \in \{D_d\} \text{ and } \boldsymbol{\theta} \in \mathbf{Z}_k. \tag{11.3.18}$$

Note that the inner minimization problem is now easy to solve.

3. Solve the minimization problem

$$\boldsymbol{\theta}_{k+1} = \arg\min_{\boldsymbol{\theta}} \phi[\mathbf{M}(D,\boldsymbol{\theta})], \boldsymbol{\theta} \in \{\Theta_{ad}\}. \tag{11.3.19}$$

4. If $\phi[\mathbf{M}(D_k, \boldsymbol{\theta}_{k+1})] \geq \min_{\boldsymbol{\theta} \in Z_k} \phi[\mathbf{M}(D_k, \boldsymbol{\theta})] - \varepsilon$, where ε is a small predetermined constant, then use D_k as the robust experimental design; otherwise, add $\boldsymbol{\theta}_{k+1}$ to Z_k, increase *k* by 1, and return to step 2.

Under certain assumptions, this procedure terminates after a finite number of iterations (Shimizu and Aiyoshi 1980). A numerical example of using the above algorithm for sensor location design of a distributed parameter system can be found in Uciński (2005). This algorithm will be inefficient when the number of unknown parameters or the number of design variables becomes large.

The algorithm presented in Körkel et al. (2004) is based on the assumption that the probability distribution of the unknown parameter vector is Gaussian with a known mean and covariance matrix, namely, $p(\boldsymbol{\theta}) = \mathbf{N}(\boldsymbol{\theta}_0, \mathbf{C}_p)$. This can be considered as a prior distribution obtained by inversion using the existing data, as in the case of the Bayesian experimental design. The purpose is to make a robust experimental design over the prior confidence ellipsoid, and (11.3.15) becomes

$$\max_{D} \min_{\boldsymbol{\theta}} \phi[\mathbf{M}(D, \boldsymbol{\theta})], D \in \left\{D_d\right\} \text{and } \boldsymbol{\theta} \in \left\{\left\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\right\|_{\mathbf{C}_p} < \gamma\right\}. \tag{11.3.20}$$

Here, $\left\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\right\|_{\mathbf{C}_P} = [(\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T \mathbf{C}_P^{-1} (\boldsymbol{\theta} - \boldsymbol{\theta}_0)]^{1/2}$ is the generalized L_2-norm and γ is a positive number. In particular, when the objective function $\phi[\mathbf{M}(D, \theta)]$ is replaced by its first-order approximation in the prior confidence ellipsoid, the inner minimization problem can be solved explicitly, namely, we have

$$\begin{aligned}\min_{\|\boldsymbol{\theta}-\boldsymbol{\theta}_0\|_{\mathbf{C}_P}<\gamma} \phi[\mathbf{M}(D, \boldsymbol{\theta})] &\approx \min_{\|\boldsymbol{\theta}-\boldsymbol{\theta}_0\|_{\mathbf{C}_P}<\gamma} \left\{\phi[\mathbf{M}(D, \boldsymbol{\theta}_0)] + \frac{\partial \phi}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0} (\boldsymbol{\theta} - \boldsymbol{\theta}_0)\right\} \\ &= \phi[\mathbf{M}(D, \boldsymbol{\theta}_0)] + \gamma \left\|\frac{\partial}{\partial \boldsymbol{\theta}} \phi[\mathbf{M}(D, \boldsymbol{\theta}_0)]\right\|_{\mathbf{C}_P}.\end{aligned} \tag{11.3.21}$$

The max-min robust design (11.3.20) is thus reduced to

$$D_R^* = \arg\max_{D} \left\{\phi[\mathbf{M}(D, \boldsymbol{\theta}_0)] + \gamma \left\|\frac{\partial}{\partial \boldsymbol{\theta}} \phi[\mathbf{M}(D, \boldsymbol{\theta}_0)]\right\|_{\mathbf{C}_P}\right\}, D \in \left\{D_d\right\}. \tag{11.3.22}$$

The right-hand side of the above equation consists of two terms: The first term is a local optimal design for $\boldsymbol{\theta}_0$ called "the nonrobust part" of the design; the second term is called the "robust part" of the design which can be seen as a penalty to the parameter variability. The radius γ of the admissible region now becomes a weight coefficient between the two parts. This algorithm, of course, is not completely robust because of the use of first-order approximation.

In this section, major approaches of designing an experiment for nonlinear model inversion are reviewed. We should say that none of them is very satisfactory.

To date, finding effective algorithms for Bayesian experimental design and robust experimental design is still a challenging topic (Atkinson et al. 2007; Franceschini and Macchietto 2008; Uciński 2005; Alaña and Theodoropoulos 2012), especially for distributed parameter systems and large-scale systems.

11.3.4 Interval Identifiability and Experimental Design

The ill-posed nature of the inverse problem must be considered when designing an experiment. In Sect. 2.2.3, we learned that without the support of sufficient data, the inverse solution will be nonunique and unstable. If the data collected from a designed experiment are insufficient, the design is unsuccessful. But, how sufficient is sufficient? And, can we answer this question before an experiment is actually conducted? The classical identifiability requires that the mapping between the inverse solution and observed data is an injection (i.e., one-to-one function). This requirement, of course, can never be satisfied in practice because of the existence of observation error. The quasi-identifiability defined in Chap. 2 allows the existence of observation error but requires the quasi-solution be unique and continuously dependent on data. Finding an experiment to satisfy such a requirement may be very expensive, if not impossible. The interval identifiability (INI) introduced in Sun and Yeh (1990) does not require the uniqueness of inverse solution. Instead, it only requires that the identified parameter fall into an assigned region around the true parameter.

In this section, the concept of INI is used for finding a robust design with significantly decreased computational effort. Moreover, if the model error can be neglected, the data sufficiency problem can be addressed before the experiment is really conducted.

11.3.4.1 Interval Identifiability

A parameter vector $\boldsymbol{\theta} \in \mathbf{P}_{ad}$ is said to be *interval identifiable* for a given accuracy requirement $\varepsilon > 0$ if there is a design D that satisfies the condition below

$$\left\| \mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta}) \right\| < \delta \text{ implies } \left\| \boldsymbol{\theta}' - \boldsymbol{\theta} \right\| < \varepsilon, \tag{11.3.23}$$

where $\boldsymbol{\theta}' \in \mathbf{P}_{ad}$, and $\delta > 0$ is a number to be assigned later. Equation (11.3.23) means that once the distance between $\mathbf{u}_D(\boldsymbol{\theta}')$ and $\mathbf{u}_D(\boldsymbol{\theta})$ is less than δ in the observation space, the distance between $\boldsymbol{\theta}'$ and $\boldsymbol{\theta}$ must be as close as required in the parameter space. In the study of INI, the weighted L_2-norm with $\sum w_i^2 = 1$ (see Sect. 2.4.1) will be used for all spaces. For example, we can assign a small weight to a component of $\boldsymbol{\theta}$ if the required accuracy to that component is low, and assign the

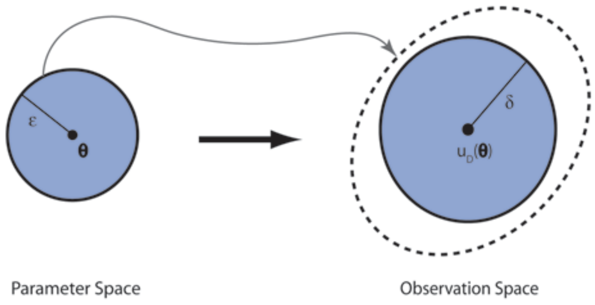


Fig. 11.3 Illustration of the interval identifiability concept

small weight to a component of $\mathbf{u}_D(\boldsymbol{\theta})$ if the observation error associated with that component is large. Note that condition (11.3.23) is different from the continuity of $\mathbf{u} = \mathbf{u}(\boldsymbol{\theta})$ because the value of δ is assigned and the value of $\left\|\mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta})\right\|$ is dependent on a design.

Condition (11.3.23) is equivalent to the following statement:

$$\left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\| \geq \varepsilon \text{ implies } \left\|\mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta})\right\| \geq \delta. \tag{11.3.24}$$

But, this condition can be tested by solving the following minimization problem:

$$S_D(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}' \in \mathbf{P}_{ad}} \left\|\mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_\mathbf{D}(\boldsymbol{\theta})\right\|, \text{ s.t.} \left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\| \geq \varepsilon. \tag{11.3.25}$$

If $S_D(\boldsymbol{\theta}) \geq \delta$, then $\boldsymbol{\theta}$ is interval identifiable. Problem (11.3.25) can be solved by the Gauss–Newton method with a penalty term given by

$$\begin{aligned} &S_D(\boldsymbol{\theta},\boldsymbol{\theta}') = \left\|\mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta})\right\|_2^2 + \lambda \mathbf{g}^2(\boldsymbol{\theta},\boldsymbol{\theta}'), \text{where} \\ &\mathbf{g}^2(\boldsymbol{\theta},\boldsymbol{\theta}') = \max(\varepsilon^2 - \left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\|_2^2, 0). \end{aligned} \tag{11.3.26}$$

The concept of INI can be explained intuitively with Fig. 11.3. Region $\left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\| \leq \varepsilon$ is an *m*-dimensional ellipsoid in the parameter space with center $\boldsymbol{\theta}$ and $\left\|\mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta})\right\| \leq \delta$ is an *n*-dimensional ellipsoid in the state space with center $\mathbf{u}_D(\boldsymbol{\theta})$. Condition (11.3.23) means that the image of the region $\left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\| < \varepsilon$ in the observation space (shown by the dotted ellipse) must contain the region $\left\|\mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta})\right\| < \delta$. The size of region $\left\|\mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta})\right\| < \delta$ depends on design *D*. It becomes smaller when the designed data provide more information and vice versa. After ε and δ are assigned, we can always modify the design to make the region $\left\|\mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta})\right\| < \delta$ just inside the image region corresponding to $\left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\| < \varepsilon$ in the observation space.

With this geometric explanation, the INI condition for a design can be checked directly, for example, by finding out if the condition $\left\|\mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta})\right\| \geq \delta$ is satisfied by all $\boldsymbol{\theta}'$ located at the ends of each axis of the ellipsoid $\left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\| \leq \varepsilon$.

INI is able to deal with the observation error. After a designed experiment is actually conducted, the observed data $\tilde{\mathbf{u}}_D(\theta)$ may contain the observation error $\mathbf{e}_D$,

$$\tilde{\mathbf{u}}_D(\boldsymbol{\theta}) = \mathbf{u}_D(\boldsymbol{\theta}) + \mathbf{e}_D. \tag{11.3.27}$$

Let η be the upper bound of $\left\|\mathbf{e}_D\right\|$. If the model error can be ignored, we expect to find such a design *D* that satisfies

$$\left\|\mathbf{u}_D(\boldsymbol{\theta}') - \tilde{\mathbf{u}}_D(\boldsymbol{\theta})\right\| < \eta \text{ implies } \left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\| < \varepsilon. \tag{11.3.28}$$

A design satisfying (11.3.28) is called an *INI design* for identifying $\boldsymbol{\theta}$. It is a sufficient design because the required reliability of INI can be assured before the designed observation data are really collected. Let us show how an INI design can be found when there is an observation error. From the definition of INI, if there is a design *D* that can make $\boldsymbol{\theta}$ be interval identifiable for $\delta = 2\eta$, namely,

$$\left\|\mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta})\right\| < 2\eta \text{ implies } \left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\| < \varepsilon, \tag{11.3.29}$$

then, for this design, once $\left\|\mathbf{u}_D(\boldsymbol{\theta}') - \tilde{\mathbf{u}}_D(\boldsymbol{\theta})\right\| < \eta$ we must have

$$\begin{aligned}\left\|\mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta})\right\| &\leq \left\|\mathbf{u}_D(\boldsymbol{\theta}') - \tilde{\mathbf{u}}_D(\boldsymbol{\theta})\right\| + \left\|\mathbf{u}_D(\boldsymbol{\theta}) - \tilde{\mathbf{u}}_D(\boldsymbol{\theta})\right\| \\ &< \eta + \eta = \delta .\end{aligned} \tag{11.3.30}$$

From (11.3.29), $\left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\| < \varepsilon$ is obtained (i.e., *D* is an INI design for identifying $\boldsymbol{\theta}$). The optimal design then can be chosen from all sufficient designs based on other criteria, for instance, the cost-effectiveness. Note that INI requires that the observation data be sensitive enough to all components of the estimated parameter vector.

11.3.4.2 Robust INI Design

For nonlinear model inversion, we need a robust INI design that is sufficient for the identification of all $\boldsymbol{\theta} \in P_{ad}$. As in the last subsection, we can define the worst-case parameter (WCP) as the most difficult one to be identified in the admissible region. If a design is an INI design for identifying WCP, it must be an INI design for identifying all $\boldsymbol{\theta} \in P_{ad}$ and, thus, a *robust INI design*. From (11.3.25), WCP can be calculated by

$$\tilde{\boldsymbol{\theta}}_D = \arg\min_{\boldsymbol{\theta}} S_D(\boldsymbol{\theta}), \boldsymbol{\theta} \in P_{ad}. \tag{11.3.31}$$

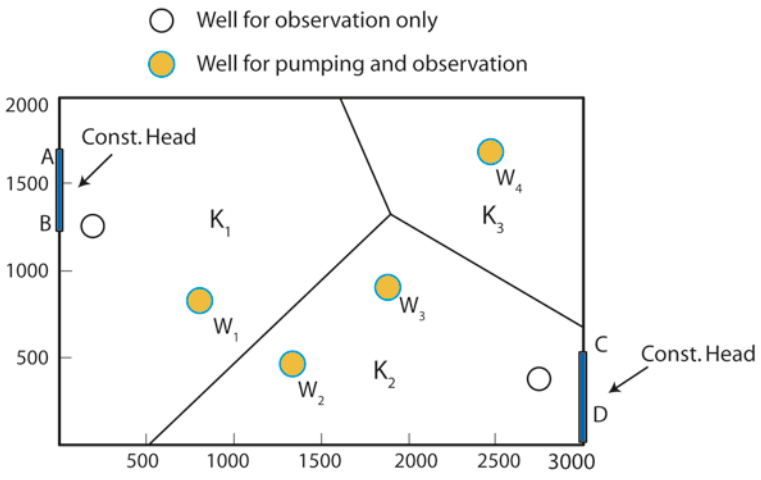


Fig. 11.4 Problem set up for Example 11.6

In fact, we do not need to solve (11.3.31). Because the WCP for an INI design is the most insensitive parameter vector to the designed observations, it must be at an extreme point of the boundary of P_{ad} (Hendrix and Boglárka 2010). Especially, when P_{ad} is an *m*-dimensional hyperbox, WCP must be located at a vertex of the hyperbox. In this case, if the dimension *m* is not high, we can simply compute and compare the values of $S_D(\boldsymbol{\theta})$ at all its vertices. The vertex associated with the smallest value is WCP, and, if this value is not less than 2η, we can conclude that design *D* is a robust INI design.

Example 11.6 *Finding an INI robust design for hydraulic conductivity estimation*
Let us consider the hypothetical aquifer described in Example 5.14. Assume that the hydraulic conductivity of the aquifer is parameterized into a three-zone structure by the Voronoi zonation method as shown in Fig. 11.4. Our purpose is to find an INI robust design for estimating the values of hydraulic conductivity, K_1, K_2 and K_3, associated with the three zones. From prior information, the admissible region is defined as $[20 \le K_1 \le 40; 30 \le K_2 \le 50; 10 \le K_3 \le 30]$ in [m/day], and the specific storage coefficient $S_s = 0.0001\ \text{m}^{-1}$ is assumed to be given.

Our initial design *ID* is (i) pumping at 500, 2000, 1000 m^3/day from W_1, W_2, and W_3, respectively, with a total pumping rate of 3500 m^3/day; (ii) the head values are taken from observation wells located in zone 1 and zone 2 only; and (iii) head observations are taken at 0.05, 0.1, 0.5, 1.0, and 3.0 days, respectively. The observation wells are placed in those areas where the heads are most sensitive with respect to the hydraulic conductivity (see Fig. 5.5).

In the definition of INI, we set $\varepsilon = 3.0$ m/day as the root-mean-square error (RMSE) of parameter estimation and $\delta = 2\eta = 0.1$ m. To test if a design *D* is an INI robust design, we need to find the WCP associated with the design first and then check if $S_D(WCP) \ge 0.1$. Since the WCP must be located at a vertex of the admis-

Table 11.1 The eight vertices of the admissible region

Vertex	Number	Vertex	Number
(40,50,30)	1	(20,50,30)	5
(40,50,10)	2	(20,50,10)	6
(40,30,30)	3	(20,30,30)	7
(40,30,10)	4	(20,30,10)	8

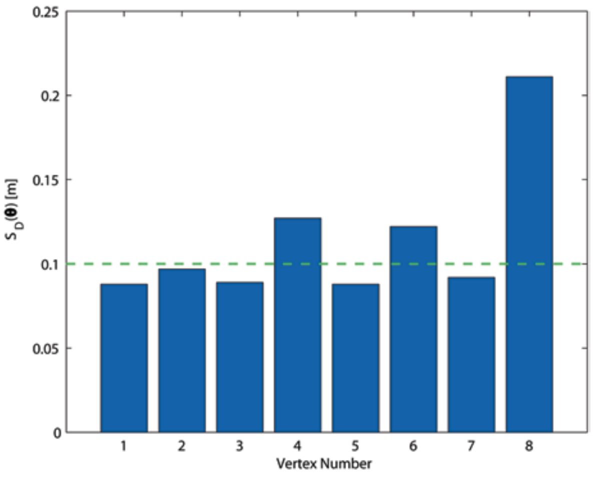


Fig. 11.5 Value of $S_D(\boldsymbol{\theta})$ at the vertices of the admissible region in the modified design, where the dashed line corresponds to the fitting residual requirement, $\delta = 0.1$ m

sible region, we only need to test the eight vertices of the 3D admissible parameter region, as numbered in Table 11.1.

The calculated values of $S_D(\boldsymbol{\theta})$ at the eight vertices of the initial design range from the minimum value 0.026 m at vertex #1 to the maximum value 0.057 m at vertex #8. This means that vertex #1 is the WCP and the initial design is not a robust design because $S_D(WCP) = 0.026 < 0.1$ m. From the calculation, we found that the smallness of $S_D(\boldsymbol{\theta})$ is caused by the insensitivity of observations with respect to K_3. Then we tried to add observation wells to the zone associated with K_3, but that only caused $S_D(WCP)$ to increase from 0.026 to 0.029 m. We also tried to increase the pumping rate at Well W_2 from 2000 to 8000 m^3/day, but that only made $S_D(WCP)$ to increase from 0.026 to 0.033 m.

For this case, in fact, adding a pumping well in zone 3 is the only effective method to increase the sensitivity of observations with respect to K_3. Let us consider a modified design that is the same as the initial design, but a new pumping well W_4 is added in zone 3 to replace pumping well W_3 in zone 2 without changing the pumping rate (see Fig. 11.4). Now, there is a pumping well within each zone. The values of $S_D(\boldsymbol{\theta})$ at the eight vertices for the modified design are shown in Fig. 11.5.

From Fig. 11.5, we can see that (i) the values of $S_D(\boldsymbol{\theta})$ are significantly increased without increasing the pumping rates, (ii) Vertex #1 is the WCP, and (iii)

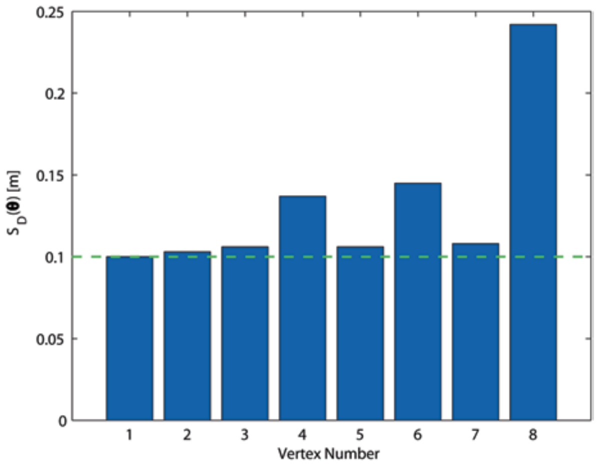


Fig. 11.6 $S_D(\boldsymbol{\theta})$ value at the vertices of the admissible region in the robust design

the modified design is still not a robust design because $S_D(WCP) = 0.088$ m is less than $\delta = 0.1$ m. According to the conceptual diagram of INI design in Fig. 11.3a, cost-effective robust INI design can be found by systematically varying the pumping rates of the three wells to make $S_D(WCP)$ just larger than 0.1 m, but no more. The minimum pumping rates under this condition are 400, 2200, and 1150 m^3/day, respectively, from W_1, W_2, and W_4. For this robust design, the calculated values of $S_D(\boldsymbol{\theta})$ at the eight vertices are shown in Fig. 11.6. It shows that Vertex #1 is still the WCP, where all K's reach their upper bounds, (40, 50, 30) m/day, and $S_D(WCP) = 0.1$ m.

To test the sufficiency and robustness of the design, we assume that the true parameter vector is (38, 48, 28) m/day, which is very close to the WCP. Using the forward solution to simulate the designed experiment, we obtained a set of designed "observation data." Applying these data and the initial guess (30, 30, 30) m/day to solve the inverse problem, we terminated the iteration process when the fitting residual reached $\delta = 0.1$ m. At that time, the identified parameter vector is (36.76, 43.97, 29.02) m/day, and the RMSE of $\boldsymbol{\theta}$ is $\varepsilon = 2.50$ m/day, which is indeed within the required interval (0, 3) m/day.

Through this example we learned that:

- Making a robust design for a distributed parameter model inversion may be difficult, computationally expensive, and even infeasible because it requires the designed observations be sufficiently sensitive to all components of the unknown parameter vector in the admissible region.
- Prior information that determines the size of the admissible region plays an important role and can be incorporated into the INI robust design directly.
- In this example, we set the norm of the observation error to $\eta = 0.05$ m, but in practice, the model structure error cannot be avoided in the estimation of a distributed parameter due to parameterization. We cannot simply combine the structure error into the observation error by increasing the value of η. In this

example, when η is increased from 0.05 to 0.1 m, the total pumping rate has to be increased from 3750 to 8800 m^3/day in order to satisfy the INI requirement. The "sufficiency" of an INI design is under the no model error assumption. For the identification of a distributed parameter, we have to assume that the model structure error can be neglected, as in this example. The problem of designing an experiment for model structure identification will be discussed in Sect. 11.4 and also in the next chapter.

- In INI, the value of ε is an accuracy requirement of the estimated parameters. A smaller value of ε requires the observation data to provide more information for inversion. In this example, we set $\varepsilon = 3.0$ m/day for identifying the hydraulic conductivity. But how is this value determined? In EWR modeling, this question cannot be answered directly because the identification of model parameters is usually not the ultimate objective of model development. In this case, the accuracy requirement of model parameters is determined indirectly by the reliability requirement of model application. ■

INI design has several advantages. First, for a complex system, finding an INI design is easier than finding an optimal design because INI design is only an acceptable design according to a preset accuracy requirement without considering the uniqueness and optimality. Second, using the data of an INI design, the iteration process of inversion can be terminated once the fitting residual is less than a preset value δ. Third, a robust INI design can be found effectively. And finally, the INI design process is independent of linearization. After the WCP is found, however, a linearization-based optimal design approach can be used to find the optimal design.

Using different methods and criteria, different experimental designs are obtained for the same problem. The accuracy of inverse solution is not the only criterion to assess the goodness of a design or to compare two different designs. In the next section, we will see that experimental deigns should be tailored based on the objectives of model application.

11.4 Other Objectives of Optimal Experimental Design

11.4.1 Design for Cost-Effectiveness

11.4.1.1 Minimum Cost Design

So far only a single objective, maximizing the information content of data or minimizing the uncertainty of estimation, is used to define the optimal design. It implies that the cost of the experiment is not a problem. In practice, especially in EWR modeling, the budget of the experiment and the reliability of parameter estimation should be considered simultaneously by the designer. As shown in Fig. 11.7, in such a case we have multiple Pareto optimal solutions and a tradeoff must be

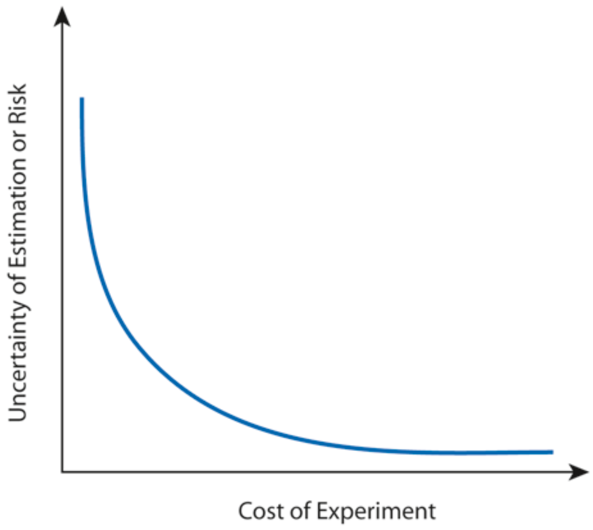


Fig. 11.7 Design as a multiobjective optimization problem

made between the two conflicting objectives of a design: minimizing the cost of the experiment and minimizing the uncertainty of estimation.

Let $Q(D)$ be the cost of a design D and $F(D)$ be a measure of information content of data. For linear model inversion, let $F(D) = \phi[\mathbf{M}(D)]$; for nonlinear model inversion, let $F(D) = E_{\boldsymbol{\theta}}\left\{\phi[\mathbf{M}(D,\boldsymbol{\theta})]\right\}$ when the Bayesian optimal design method is used, and $F(D) = \phi[\mathbf{M}(D,\tilde{\boldsymbol{\theta}}_D)]$ when the max-min robust design method is used, where $\tilde{\boldsymbol{\theta}}_D$ is the WCP. The cost-effective optimal design for inversion can be formulated as

$$\min_D \left\{Q(D), -F(D)\right\}, D \in \left\{D_{ad}\right\}. \tag{11.4.1}$$

We can use the multiobjective optimization algorithms discussed in Chap. 3 to solve problem (11.4.1) (Schöneberger et al. 2010). One way to find the set of Pareto optimal solutions is to solve a single-objective optimal design problem, either $\min Q(D)$ or $\max F(D)$, repeatedly; in contrast, MOEA methods evolve an approximation to the Pareto optimal set simultaneously (Coello Coello 2007; Marler and Arora 2004; Reed et al. 2013). In practice, cost-effective design is often formulated into constraint optimization problems, either

$$\min_D Q(D), D \in \left\{D_{ad}\right\}, \text{s. t. } \phi[\text{Cov}(\tilde{\boldsymbol{\theta}}_D)^{-1}] < \varepsilon, \tag{11.4.2}$$

where ε is a tolerance on the uncertainty of estimation, or

$$\min_D \ \phi[\text{Cov}(\hat{\boldsymbol{\theta}}_D)^{-1}], D \in \left\{D_{ad}\right\}, \text{s. t. } Q(D) < B \tag{11.4.3}$$

where B is the limit of the budget. These problems are still hard to solve when the number of design variables and/or the number of the unknown parameters are large. For example, Hadka and Reed (2012) demonstrated that most modern MOEAs can fail in terms of both convergence and reliability on test problems with as few as four objectives. Moreover, for nonlinear models, the constraint in (11.4.2) is unknown during the design stage. In order to further decrease the computational effort, in many EWR case studies, such as the design of pumping tests or tracer tests, it is often assumed that (i) the number of the unknown parameters is low after reparameterization; (ii) the optimal observation locations can be chosen only from a small number of candidate sites (the existing wells, for example); (iii) a suboptimal design obtained by a heuristic process is acceptable; and (iv) a local optimal design (not robust) is acceptable or a sequential design process would be conducted. Examples of using one or more of these assumptions to find a cost-effective design can be found in Alaña and Theodoropoulos (2012), Catania and Paladino (2009), Leven and Dietrich (2006), Altmann-Dieses et al. (2002), Knopman et al. (1991), Knopman and Voss (1989), Loaiciga et al. (1992), Nishikawa and Yeh (1989), Sun et al. (2013), Reed et al. (2000), and Meyer et al. (1994).

11.4.1.2 Cost-Effective INI Design

A cost-effective INI design is such a design that has the lowest cost among all robust designs for the given requirement of INI. It can be formulated into the following constrained optimization problem:

$$\min_{D} Q(D), D \in \{D_{ad}\}, \text{s. t.}\, S_D(\tilde{\boldsymbol{\theta}}_D) \geq 2\eta, \tag{11.4.4}$$

where $\tilde{\boldsymbol{\theta}}_D$ is the WCP defined in (11.3.31). For a large-scale problem, we may use a heuristic process to find a suboptimal cost-effective design. Start with an initial design, if the design is not a robust INI design, modify the design parameters gradually to obtain a new design based on sensitivity analysis, including (i) increasing the excitations to the system, (ii) relocating the existing observations, and (iii) adding new observations to increase the contribution of observations to parameter identification as large as possible, with a slight increase of the cost. Then test if the new design is a robust INI design. This process is repeated until a robust INI design is obtained.

Example 11.7 *Find a cost-effective robust design for Example 11.6*
Let us consider how the experimental cost of the INI robust design in Example 11.6 changes with different accuracy requirements of inversion. For simplification, the total pumping rate is used to measure the cost of the experiment. Let the pumping rates of the three wells W_1, W_2 and W_4 be the design variables and keep all other design settings the same as those given in Example 11.6. After systematically changing the value of ε and finding the robust INI designs accordingly (i.e., find-

Table 11.2 Robust INI design pumping rates for different values of ε

ε	1.0	2.0	3.0	4.0	5.0	6.0
W1	1900	800	400	300	250	200
W2	6100	3300	2200	1600	1350	950
W4	3900	2100	1200	800	600	450
Total	11,900	6200	3800	2700	2000	1600

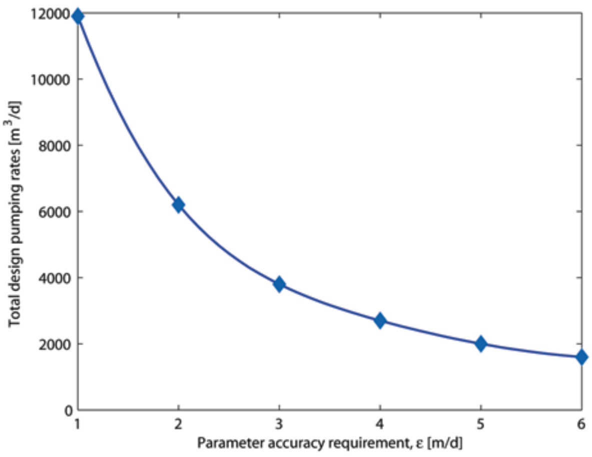


Fig. 11.8 Total pumping rate (a proxy of experimental cost) vs. the parameter accuracy requirement of INI

ing the minimum pumping rates to make $S_D(WCP) > 0.10$ m), we obtained a set of solutions as shown in Table 11.2. By translating the total pumping rate to cost, an L-curve related to the cost of the experiment and the accuracy requirement of INI is generated in Fig. 11.8. It shows that the cost of the experiment increases very fast with the increase of the accuracy requirement of INI or the decrease of ε. The problem is whether we really need to increase the cost of the experiment significantly for finding a more accurate inverse solution. To address this question, we need to consider other objectives of experimental design besides the cost-effectiveness.

11.4.2 Design for Model Prediction

In most real case studies, the ultimate objective of developing a model is to predict the states of a system when control variables and/or boundary conditions are changed. In this case, the major objective of designing an experiment for model calibration is the reliability of model prediction, rather than the accuracy of model parameters. Intuitively, a calibrated model that can minimize the uncertainty of model parameters should also minimize the uncertainty of model prediction. The problem is that the data provided by a design can only help us find approximate values of the unknown parameters, instead of their true values. Although the objectives of making a design for parameter estimation and making a design for model prediction are not in conflict with each other, the optimal design for them would not be the same, especially when the cost-effectiveness is considered. For example, a design based on E-optimality seeks to decrease the uncertainty of a parameter component that has the maximum uncertainty, but that parameter component may be insensitive to the required model prediction. In this case, a design that seeks to decrease the uncertainty of the most sensitive parameter component to the required model prediction should be more effective. Moreover, because different model predictions are sensitive to different parameter components, the optimal design is different for different model predictions.

Let $\mathbf{u} = \mathbf{u}(\boldsymbol{\theta}, \mathbf{q})$ be a model of a system, where parameters $\boldsymbol{\theta}$ need to be estimated and $\mathbf{q}$ is the control vector (sink/source and/or boundary conditions). Designing an experiment *D* for model prediction needs: (i) Use control variables $\mathbf{q}_D$ that describe the conditions of the designed experiment as model inputs to run the simulation model, from which model outputs $\mathbf{u}_D(\boldsymbol{\theta}) = \mathbf{u}(\boldsymbol{\theta}, \mathbf{q}_D)$ corresponding to the design are obtained; (ii) use $\tilde{\mathbf{u}}_D(\boldsymbol{\theta}) = \mathbf{u}_D(\boldsymbol{\theta}) + \mathbf{e}_D$ as the "observed data" of the experiment, where $\mathbf{e}_D$ simulates observation errors, to find an inverse solution $\hat{\boldsymbol{\theta}}_D$; (iii) use control variables $\mathbf{q}_E$ that describe the conditions of prediction and $\hat{\boldsymbol{\theta}}_D$ as inputs to run the simulation model, from which the required model prediction $\hat{\mathbf{u}}_E = \mathbf{u}(\hat{\boldsymbol{\theta}}_D, \mathbf{q}_E)$ is obtained. After knowing how to find $\hat{\boldsymbol{\theta}}_D$ and $\hat{\mathbf{u}}_E$ for a given design *D*, we now consider how to find the optimal design for model prediction.

11.4.2.1 Optimality Criteria for Model Prediction

First, assume the model is linear with respect to $\boldsymbol{\theta}$ and the required model predictions are represented by a *k*-dimensional vector $\mathbf{u}_E = \mathbf{A}_E\boldsymbol{\theta}$, where $\mathbf{A}_E$ is a $k \times m$ matrix and *m* is the dimension of $\boldsymbol{\theta}$. Because $\boldsymbol{\theta}$ is unknown, model prediction $\mathbf{u}_E$ can only be estimated by $\hat{\mathbf{u}}_E = \mathbf{A}_E\hat{\boldsymbol{\theta}}_D$, where $\hat{\boldsymbol{\theta}}_D$ is an estimation of $\boldsymbol{\theta}$ satisfying the observation equation $\mathbf{u}_D^{obs} = \mathbf{A}_D\hat{\boldsymbol{\theta}}_D + \mathbf{e}_D$. In this case, the optimal design *D* should minimize a norm of $\mathrm{Cov}(\hat{\mathbf{u}}_E)$. Using (11.2.3), we have

$$\begin{aligned}\mathrm{Cov}(\hat{\mathbf{u}}_E) &= \mathrm{Cov}(\mathbf{A}_E\hat{\boldsymbol{\theta}}_D) = \mathbf{A}_E\mathrm{Cov}(\hat{\boldsymbol{\theta}}_D)\mathbf{A}_E^T \\ &= \mathbf{A}_E(\mathbf{A}_D^T\mathbf{C}_D^{-1}\mathbf{A}_D)^{-1}\mathbf{A}_E^T\end{aligned} \tag{11.4.5}$$

in which $\mathbf{C}_D$ is the covariance matrix of the observation error vector (see Eq. (11.2.3)). The *G-optimality* criterion for model prediction seeks to minimize the maximum variance of the matrix $\mathrm{Cov}(\hat{\mathbf{u}}_E)$. Let $d_1, d_2, \cdots, d_k$ all be diagonal elements of $\mathrm{Cov}(\hat{\mathbf{u}}_E)$, the G-optimal design is the solution of the following optimization problem:

$$D_G^* = \arg\min_D \max_i \left\{d_i\right\}, 1 \le i \le k, D \in \left\{D_{ad}\right\}. \tag{11.4.6}$$

Another design criterion for model prediction is the *I-optimality* that seeks to minimize the average prediction variance,

$$D_I^* = \arg\min_D \text{trace}[\text{Cov}(\hat{\mathbf{u}}_E)], D \in \{D_{ad}\} \tag{11.4.7}$$

which is similar to the A-optimality for inversion that minimizes the trace of $\text{Cov}(\hat{\boldsymbol{\theta}}_D)$ (see Sect. 11.2).

When the model used for prediction is nonlinear, the optimality criteria will depend on the unknown parameter vector $\boldsymbol{\theta}$. Using the linearization technique described in Sect. 11.3.1, we have $\hat{\mathbf{u}}_E \approx \mathbf{J}_E\hat{\boldsymbol{\theta}}_D$ and $\mathbf{u}_D^{obs} \approx \mathbf{J}_D\hat{\boldsymbol{\theta}}_D + \mathbf{e}_D$, where $\mathbf{J}_E = [\partial\mathbf{u}_E / \partial\boldsymbol{\theta}]$ and $\mathbf{J}_D = [\partial\mathbf{u}_D / \partial\boldsymbol{\theta}]$ are Jacobian matrices evaluated at $\boldsymbol{\theta}$. Replacing $\mathbf{A}_D$ by $\mathbf{J}_D(\boldsymbol{\theta})$ and $\mathbf{A}_E$ by $\mathbf{J}_E(\boldsymbol{\theta})$ in (11.4.5), we obtain

$$\text{Cov}[\hat{\mathbf{u}}_E(\boldsymbol{\theta})] = \mathbf{J}_E(\boldsymbol{\theta})[\mathbf{J}_D^T(\boldsymbol{\theta})\mathbf{C}_D^{-1}\mathbf{J}_D(\boldsymbol{\theta})]^{-1}\mathbf{J}_E^T(\boldsymbol{\theta}). \tag{11.4.8}$$

The G-optimal design for nonlinear model prediction is then given by

$$D_G^*(\boldsymbol{\theta}) = \arg\min_D \max_i \{d_i(\boldsymbol{\theta})\}, 1 \le i \le k, D \in \{D_{ad}\}, \tag{11.4.9}$$

where $d_1(\boldsymbol{\theta}), d_2(\boldsymbol{\theta}), \cdots, d_k(\boldsymbol{\theta})$ are diagonal elements of $\text{Cov}[\hat{\mathbf{u}}_E(\boldsymbol{\theta})]$. Equation (11.4.8) clearly shows that the optimal design for inversion and the optimal design for prediction are different because of the effect of the sensitivity matrix on prediction.

To deal with the difficulty of nonlinearity, we can (i) use the approaches described in Sect. 11.3.1 to find a local G-optimal design with a guessed $\boldsymbol{\theta}_0$, (ii) conduct a sequential experiment process, or (iii) find a Bayesian or a max-min robust G-optimal design. For a given $\boldsymbol{\theta}$, the optimization problem (11.4.9) can be solved by a global optimization method, and the Jacobian matrices $\mathbf{J}_D(\boldsymbol{\theta})$ and $\mathbf{J}_E(\boldsymbol{\theta})$ in (11.4.8) can be calculated by a model differentiation method given in Chap. 5.

11.4.2.2 Cost-Effective Design for Model Prediction

In EWR modeling, two objectives, the reliability of model prediction and the cost-effectiveness of data collection, are often combined when we design an experiment for model parameter estimation. For example, we want to predict the development of a contaminant plume with a model, but the model parameters need to be estimated first by performing site characterization. The problem is how we can minimize the cost of observation and at the same time minimize the uncertainty of model prediction. Toward this end, Wagner (1995) considered the problem of predicting the development of a contaminant plume in groundwater, in which the bi-objective optimal design problem is solved by constrained optimization,

$$\min_D \text{trace}\{\mathbf{J}_E(\boldsymbol{\theta}_0)[\mathbf{J}_D^T(\boldsymbol{\theta}_0)\mathbf{C}_D^{-1}\mathbf{J}_D(\boldsymbol{\theta}_0)]^{-1}\mathbf{J}_E^T(\boldsymbol{\theta}_0)\}, D \in \{D_{ad}\} \tag{11.4.10}$$

subject to

$$\sum_{l} c_l z_l \le B, \tag{11.4.11}$$

where D is a network design that prescribes where and when to measure the contaminant plume. Equation (11.4.10) means that the I-optimality criterion is used to find a local optimal design at a given $\boldsymbol{\theta} = \boldsymbol{\theta}_0$. In (11.4.11), c_l is the cost of obtaining the l-th measurement; z_l is a binary variable associated with l-th measurement, for which $z_l = 1$ when the measurement is taken and $z_l = 0$ otherwise; and B is the network design budget. By changing the value of B, we can obtain a set of Pareto optimal solutions.

In this problem, the components of $\boldsymbol{\theta}$ consist of hydraulic conductivity, porosity, and dispersion coefficients used in the coupled groundwater flow and mass transport model (See Example 5.9 in Sect. 5.3.3). The Jacobian $\mathbf{J}_D = [\partial \mathbf{u}_D / \partial \boldsymbol{\theta}]$ can be calculated by a model differentiation method, where $\mathbf{u}_D(\boldsymbol{\theta})$ is the model outputs assigned by the design (i.e., the model calculated concentration values at a set of assigned locations/times). The Jacobian $\mathbf{J}_E = [\partial \mathbf{u}_E / \partial \boldsymbol{\theta}]$ is obtained by the same method, but $\mathbf{u}_E(\boldsymbol{\theta})$ is the model outputs required by the prediction problem (the model predicted concentration values at a set of locations/ times). For searching the local optimal design at $\boldsymbol{\theta} = \boldsymbol{\theta}_0$, both $\mathbf{J}_D$ and $\mathbf{J}_E$ in (11.4.10) are evaluated at $\boldsymbol{\theta}_0$, and the constrained optimization problem can be solved by genetic algorithm (GA) or other global optimization methods. A detailed description and a numerical example for this problem can be found in Wagner (1995).

The methodology of experimental design for prediction was applied to the Death Valley regional groundwater flow model by Tiedeman et al. (2003) for decreasing the uncertainty of predicted head distribution. Janssen et al. (2008) used travel time as a state variable to minimize the uncertainty of the predicted contaminant breakthrough, in which the probability distribution of the unknown parameter (hydraulic conductivity) is assumed to be Gaussian and known.

11.4.2.3 INI Design for Model Prediction

The INI-based design approach in Sect. 11.3.4 can be extended from parameter estimation to model prediction (Sun and Yeh 1990). It has two advantages: The data sufficiency can be assessed during the design stage, and a robust design can be found effectively. Let us introduce the following definition:

A parameter vector $\boldsymbol{\theta} \in \mathbf{P}_{ad}$ is said to be *interval identifiable* for a given prediction vector $\mathbf{u}_E(\boldsymbol{\theta})$ and an accuracy requirement $\gamma > 0$, if there is a design D such that

$$\left\| \mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta}) \right\| < \delta \text{ implies } \left\| \mathbf{u}_E(\boldsymbol{\theta}') - \mathbf{u}_E(\boldsymbol{\theta}) \right\| < \gamma, \tag{11.4.12}$$

where $\boldsymbol{\theta}' \in \mathbf{P}_{ad}$, $\delta > 0$ is a number to be assigned.

Equation (11.4.12) means the distance between $\mathbf{u}_E(\boldsymbol{\theta}')$ and $\mathbf{u}_E(\boldsymbol{\theta})$ in the prediction space must be less than a required value (γ) provided that the distance between $\mathbf{u}_D(\boldsymbol{\theta}')$ and $\mathbf{u}_D(\boldsymbol{\theta})$ in the state space is less than a assigned number (δ). All results in Sect. 11.3.4 now can be derived in parallel for INI prediction. The following is a brief description.

If there is an observation error but no model error, we can assign $\delta = 2\eta$ in (11.4.12) to find a design D and show that

$$\left\|\mathbf{u}_D(\boldsymbol{\theta}') - \tilde{\mathbf{u}}_D(\boldsymbol{\theta})\right\| < \eta \text{ implies } \left\|\mathbf{u}_E(\boldsymbol{\theta}') - \mathbf{u}_E(\boldsymbol{\theta})\right\| < \gamma, \tag{11.4.13}$$

where $\tilde{\mathbf{u}}_D(\boldsymbol{\theta})$ is defined in (11.3.27). We call such a design an INI design for prediction. It is a sufficient design because once it is conducted, the designed data are collected, and the fitting residual is less than η, the inverse solution must give reliable prediction results. To determine whether a design D is an INI design for prediction, we can solve the following optimization problem

$$S_D(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta} \in P_{ad}} \left\|\mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta})\right\|, \text{s.t.}\, S_E(\boldsymbol{\theta}',\boldsymbol{\theta}) = \left\|\mathbf{u}_E(\boldsymbol{\theta}') - \mathbf{u}_E(\boldsymbol{\theta})\right\| \geq \gamma. \tag{11.4.14}$$

If $S_D(\boldsymbol{\theta}) \geq 2\eta$, we can conclude that D is an INI design for prediction associated with parameter vector $\boldsymbol{\theta}$. This optimization problem can be solved by the same approach used for solving the problem (11.3.26) but changing $\mathbf{g}^2(\boldsymbol{\theta},\boldsymbol{\theta}')$ to

$$\mathbf{g}^2(\boldsymbol{\theta},\boldsymbol{\theta}') = \max(\gamma^2 - S_E^2(\boldsymbol{\theta}',\boldsymbol{\theta}), 0). \tag{11.4.15}$$

An INI design problem for prediction can be transformed into an INI design problem for inversion considered in Sect. 11.3.4. For each given $\boldsymbol{\theta}$ in the admissible region, after linearization, the region $\left\|\mathbf{u}_E(\boldsymbol{\theta}') - \mathbf{u}_E(\boldsymbol{\theta})\right\| \geq \gamma$ in the prediction space is mapped onto a region $\left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\| \geq \varepsilon(\boldsymbol{\theta})$ (the surface of an ellipsoid) in the parameter space (Sun and Yeh 1990). The problem (11.4.14) is thus converted into problem (11.3.25), but with accuracy requirement $\varepsilon(\boldsymbol{\theta})$.

Figure 11.9 shows the relationship between prediction, observation, and parameter spaces, as well as the process of finding an INI design for prediction. A simplified algorithm consists of two steps: First, finding an ellipsoid $\left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\| \leq \varepsilon(\boldsymbol{\theta})$ in the parameter space such that its image in the prediction space (bounded by the dotted curve) is just within the ellipsoid $\left\|\mathbf{u}_E(\boldsymbol{\theta}') - \mathbf{u}_E(\boldsymbol{\theta})\right\| \leq \gamma$. Note that in this step we need only run the prediction model because the relation between $\varepsilon(\boldsymbol{\theta})$ and γ is independent of observation data. Second, finding the image of ellipsoid $\left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\| \leq \varepsilon(\boldsymbol{\theta})$ in the observation space based on an initial design. If the image does not completely contain the ellipsoid $\left\|\mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta})\right\| \leq \delta$, modify the initial design to make the observation data more sensitive to $\boldsymbol{\theta}$. As a result, the size of the ellipsoid will be decreased but the cost of the experiment will be increased. Otherwise, modify the initial design to make the size of the ellipsoid larger to decrease

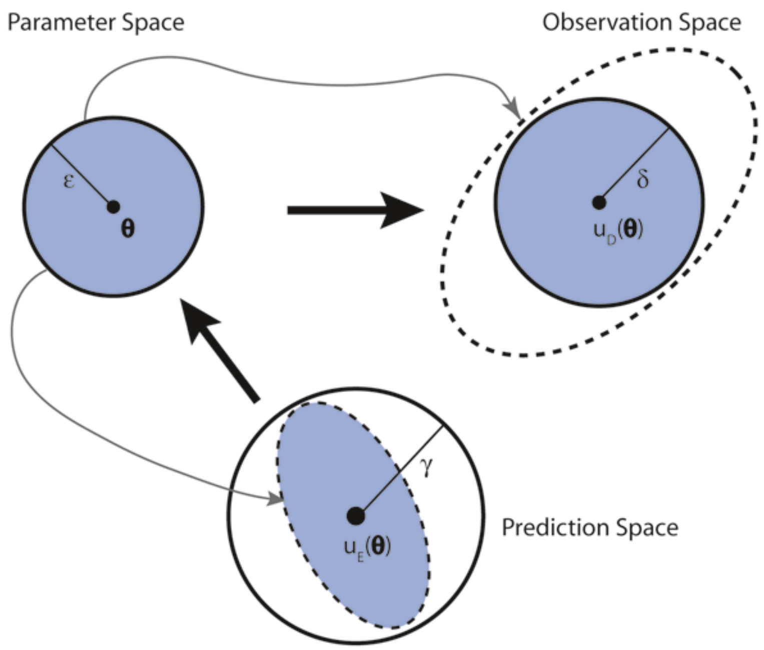


Fig. 11.9 Process of finding an INI design for prediction

the cost of the experiment. Ideally, the ellipsoid should be just within the image of $\left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\| \leq \varepsilon(\boldsymbol{\theta})$ in the observation space as shown in Fig. 11.9.

Finding a robust design for prediction is challenging when the model is nonlinear and governed by PDEs, because of the complexity and high-computational demand involved. In contrast, finding an INI robust design for prediction is relatively easy. As explained in Sect. 11.3.4, when the admissible region of $\boldsymbol{\theta}$ is an m-dimensional hyperbox, WCP must be located at a vertex of the hyperbox. After WCP is found, the robust INI design problem reduces to a local INI design problem. Moreover, a cost-effective robust design for prediction has the same formulation as (11.4.4) when $S_D(\boldsymbol{\theta})$ is defined by (11.4.14).

Example 11.8 *Find a robust INI design for the reliability of model prediction*
The hypothetical design problem in Example 11.6 is used here to show how a robust design for the prediction is made. Assume that the objective of parameter estimation is to predict the drawdowns in the three pumping wells $W_1, W_2,$ and W_4 at the steady state when their pumping rates reach the planned maximum values of 2000, 10,000, and 3000 m^3/day, respectively. It is required that the RMSE of prediction not to exceed $\gamma = 1.0$ m. A robust INI design for this problem should satisfy $S_D(WCP) \geq 2\eta = 0.1$ m.

Using the aforementioned algorithm, we find that the WCP for prediction is vertex #8 of the admissible region where all K's reach their lower bounds. As shown in Fig. 11.10, the value of $\varepsilon(\boldsymbol{\theta})$ at vertex #8 is 2.2 m/day which is less than that of

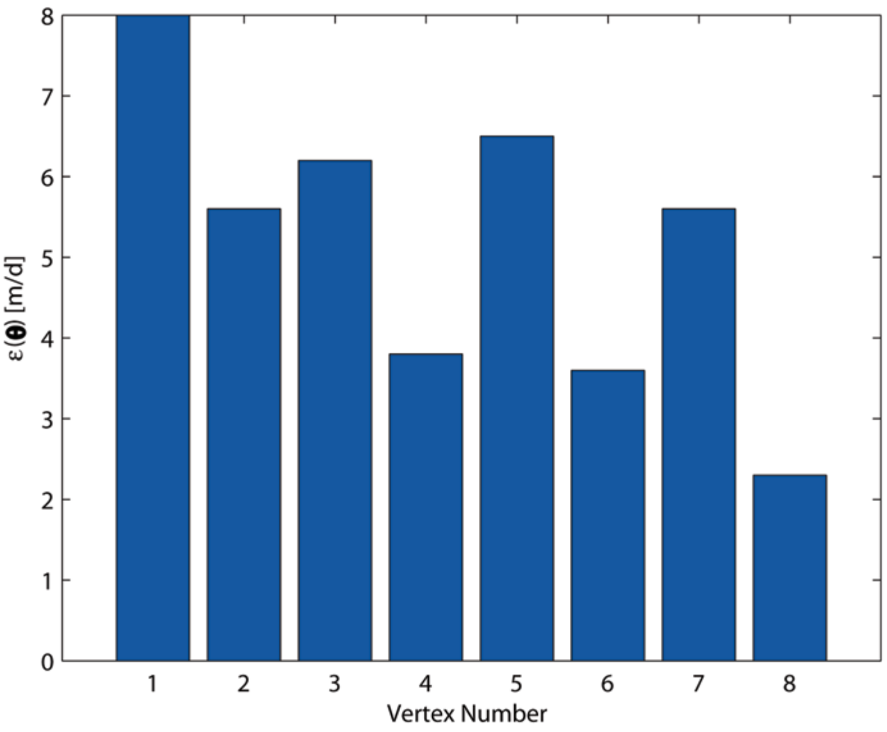


Fig. 11.10 Values of $\varepsilon(\boldsymbol{\theta})$ at the eight vertices of the admissible region for prediction accuracy requirement $\gamma = 1.0$ m

Table 11.3 Values of $\varepsilon(\boldsymbol{\theta})$ for different prediction accuracy requirement

γ (m)	Vertex #							
	1	2	3	4	5	6	7	8
0.5	4.9	3.4	3.3	2.1	4.0	1.9	3.2	1.2
1.0	8.0	5.6	6.2	3.8	6.5	3.6	5.6	2.2
1.5	11.2	8.0	8.7	5.7	8.7	5.5	7.7	3.6
2.0	13.3	10.4	11.1	7.7	11.2	7.5	9.4	5.2

other vertices. Comparing to Fig. 11.5, the WCP for inversion is vertex #1 where all *K*'s reach their upper bounds, and the parameter vector at vertex #8 is the easiest one to be identified.

We can also find how the experimental cost depends on the accuracy requirement of prediction. Table 11.3 shows the values of $\varepsilon(\boldsymbol{\theta})$ for different accuracy requirements of model prediction.

After $\varepsilon(\boldsymbol{\theta})$ is found, we can move to the second step to find a robust design for prediction (i.e., find a local INI design for $\varepsilon(WCP)$). Table 11.4 shows the neces-

Table 11.4 Pumping rates for different accuracy requirements of prediction

γ	0.5	1.0	1.5	2.0
$\varepsilon(WCP)$	1.2	2.2	3.6	5.2
W_1	1000	450	300	200
W_2	4050	2350	1500	1100
W_4	850	500	350	300
Total	5900	3300	2150	1600

sary pumping rates from the three pumping wells in order to satisfy the condition $S_D(WCP) > 0.1$ m for different accuracy requirements of prediction. In particular, for $\gamma = 1.0$ m, the robust design is to pump 450, 2350, and 500 m^3/day, respectively, from wells W_1, W_2, and W_4

As a test, assume that the true parameter vector is $\boldsymbol{\theta} = (22, 32, 12)$ m/day, which is close to WCP=(20, 30, 10) m/day and the accuracy requirement is $\gamma = 1.2$ m. Using interpolation between $\gamma = 1.0$ m and $\gamma = 1.5$ m in Table 11.4, we obtain the pumping rates of the robust design are 390, 2010, and 440 m^3/day, respectively, from wells W_1,W_2, and W_4. After running the simulation model with the true parameter vector and the designed pumping rates as model inputs and finding the designed "observation data" from the model output, we can solve the inverse problem. During the inverse solution process, we find the identified parameter vector $\boldsymbol{\theta}' = (25.3, 31.4, 12.4)$ m/day when the fitting residual $S_D(\boldsymbol{\theta}', \boldsymbol{\theta}) = 0.11 > \delta$. Then we use $\boldsymbol{\theta}'$ as the input parameter vector to run the prediction model to find the steady-state head values at wells W_1,W_2, and W_4 when the pumping rates of them reach their planned maximum values 2000, 10000, and 3000 m^3/day, respectively. We find that the prediction error $S_E(\boldsymbol{\theta}', \boldsymbol{\theta}) = 1.13$ m, which is indeed less than the required error $\gamma = 1.2$ m.

The relation between the cost of the experiment (the total pumping rate) and the accuracy of prediction is plotted in Fig. 11.11, and the relation between ε and γ (i.e., the last column in Table 11.3) is shown in Fig. 11.12.

From this example, we see that the most effective designs for prediction and the most effective design for inversion are generally different. The WCPs of them are different too because the most difficult parameter to be estimated may not be the most sensitive parameter for model prediction. Moreover, the most effective designs are different for different objectives and different accuracy requirements of model prediction. Therefore, when accurate estimation of model parameters is not the ultimate objective of an experimental design, the objective of model prediction or, more generally, the objective of model application, should be taken into account during the data collection stage. INI design provides an effective tool for this purpose.

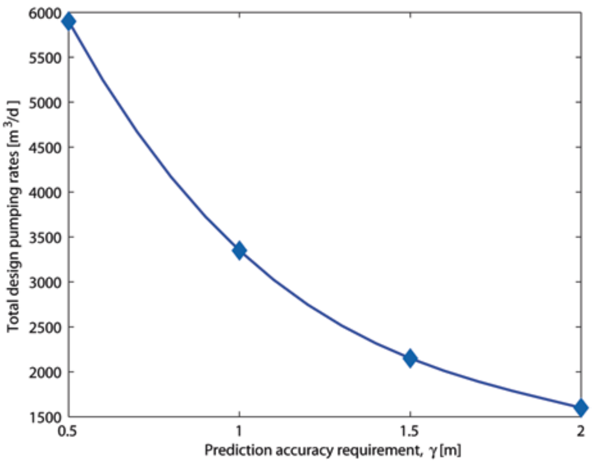


Fig. 11.11 Cost of experiment vs. accuracy requirements of prediction

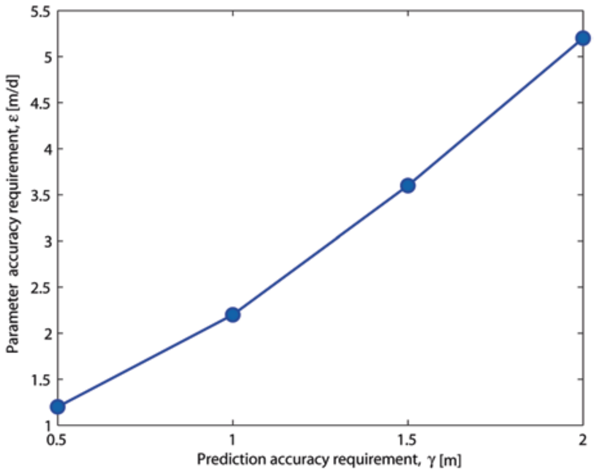


Fig. 11.12 Relation between parameter accuracy requirement ε and prediction accuracy requirement γ

11.4.2.4 Retrospective Design for Prediction

In EWR it is often the case that a previously designed observation system already exists, data provided by the system have been used to calibrate the model for prediction, and the results of model prediction have been validated by real observations. At this point, if the results are satisfied, we may ask how to decrease observation locations and times to save the operation cost. Otherwise, we may ask how to increase observation locations and times to improve the reliability of model prediction. The problem of retrospective design for prediction is thus presented. It is easier to solve than a prospective design problem because (i) the objectives of design are specified, (ii) the number of design variables is relatively small; (iii) a local optimal design would be sufficient, and (iv) the work on model modification becomes less.

There are different criteria to formulate a retrospective design problem. When data are sufficient, we may use

$$\min_{D} Q(D), D \in \left\{D_{ad}\right\}, \text{s. t.} \left\|\mathbf{u}_E(\hat{\boldsymbol{\theta}}_D) - \mathbf{u}_E^{obs}\right\| < \varepsilon, \tag{11.4.16}$$

where $Q(D)$ is the cost of the experiment, $\mathbf{u}_E(\hat{\boldsymbol{\theta}}_D)$ are the required model predictions after the model is calibrated by the data of a new design D, $\mathbf{u}_E^{obs}$ are the corresponding real observations, and $\left\{D_{ad}\right\}$ contains all possible changes to the original observation system. In order to make the solution of (11.4.16) easier, the observation locations and/or times are deleted, gradually, one or more at a time. The design obtained from (11.4.16) is only a suboptimal one. When the original observation data are insufficient, we may use the following formulation

$$\min_{D} \left\|\mathbf{u}_E(\hat{\boldsymbol{\theta}}_D) - \mathbf{u}_E^{obs}\right\|, D \in \left\{D_{ad}\right\}, \text{s. t. } Q(D) < B. \tag{11.4.17}$$

The observation locations and/or times are added, gradually, one or more at a time. Note that the prediction vector may consist of several state variables. For example, in a groundwater remediation project, we need to predict the total mass of contaminants, the high concentration areas, and arrival times to assigned locations. In this case, (11.4.17) is a multiobjective optimization problem (Zhang et al. 2005; Mayer et al. 2002; Kollat et al. 2011; Reed and Minsker 2004).

11.4.3 Design for Decision Making

11.4.3.1 Optimality Criteria of Experimental Design For Model Application

When the ultimate goal of constructing a model is to solve an application problem rather than the identification of model parameters, we can design an experiment to increase the reliability of a model application while reducing the cost of the experiment. Water resources management and contaminated aquifer remediation are typical examples of model applications related to decision making. Model prediction considered in the last subsection can be seen as a simple case of model application. Let the k-dimensional vector $\mathbf{g}_E(\boldsymbol{\theta})$ represent k objectives of a given model application, where $\boldsymbol{\theta}$ is the model parameter vector. After an experimental design D is conducted, the designed observation data $\tilde{\mathbf{u}}_D(\boldsymbol{\theta})$ are collected, and the inverse solution $\hat{\boldsymbol{\theta}}_D$ is obtained, we have the model-calculated values of model application $\hat{\mathbf{g}}_E = \mathbf{g}_E(\hat{\boldsymbol{\theta}}_D)$. The problem is how to minimize the uncertainty of $\hat{\mathbf{g}}_E$.

Comparing the problem considered in the last subsection and the problem considered here, we see that the only difference between them is that the model prediction vector $\mathbf{u}_E(\boldsymbol{\theta})$ is replaced by a general model application vector $\mathbf{g}_E(\boldsymbol{\theta})$. Therefore, all results derived for the former can be derived in parallel for the latter. For example, according to (11.4.8), we have

$$\mathrm{Cov}[\hat{\mathbf{g}}_E] = \mathbf{J}_E[\mathbf{J}_D^T\mathbf{C}_D^{-1}\mathbf{J}_D]^{-1}\mathbf{J}_E^T, \tag{11.4.18}$$

where $\mathbf{J}_E = [\partial\mathbf{g}_E / \partial\boldsymbol{\theta}]$. The G-optimality and I-optimality criteria for model application can then be obtained.

11.4.3.2 Remediation Design

Model-based remediation design is one of the major areas of model application in the EWR and has been studied for more than three decades. Because of the complexity and difficulty of aquifer cleanups, remediation design is a critical component. There are many different remediation techniques, such as hydraulic capture, pump-and-treat, bioremediation, soil vapor extraction, in situ chemical treatment, and natural attenuation. Because the applicability of a specific remediation technique is site dependent, a model-based feasibility study is needed. Let us use the pump-and-treat (PnT) technique as an example to show how the optimal remediation design is formulated, especially, how OED for model calibration and parameter estimation is used in a remediation design.

Decision variables
Besides extraction wells, injection wells may also be needed when a hydraulic wall is needed to prevent the contaminated plume from migrating to the downstream areas. Thus, decision variables of a PnT design consist of the pumping rate of each potential well, $q_i(t)(i = 1, 2, \cdots, I)$. The rate is negative for extraction wells, positive for injection wells, and zero for inactive wells. For simplification, we assume that the total remediation period is divided into J intervals $[t_{j-1}, t_j](j = 1, 2, \cdots, J)$ and all $q_i(t)$'s are constant in each interval $(q_i(t) = q_{i,j}$, when $t_{j-1} \le t < t_j)$, where $t_0 = 0$ and $t_J = t_f$ are the initial and final times of the remediation period, respectively. The decision variable vector thus becomes

$$\mathbf{q}_E = \left\{q_{i,j} \mid i = 1, 2, \cdots, I; j = 1, 2, \cdots J\right\}. \tag{11.4.19}$$

Simulation model
The forward model is a coupled groundwater flow and mass transport model that simulates the movement and change of the contaminant plume caused by the

designed remediation activities. After incorporating the design variables, the model in Example 5.9 becomes

$$\begin{cases} S_y \dfrac{\partial h}{\partial t} - \nabla \cdot [K(h-b)\nabla h] + Q_N - Q_P = 0 \\ \dfrac{\partial \phi C}{\partial t} - \nabla \cdot [\phi \mathbf{D} \nabla C] + \nabla \cdot (\phi \mathbf{V} C) - S(C) = 0 \\ \qquad 0 \le t \le t_f, \mathbf{x} \in (\Omega) \\ \text{subject to initial and boundary conditions} \\ h|_{t=0} = f_0, h|_{(\Gamma_1)} = f_1 \text{ and } K(h-b)\nabla h \cdot \mathbf{n}|_{\Gamma_2} = f_2 \\ C|_{t=0} = g_0, C|_{(\Gamma_1)} = g_1 \text{ and } \phi(\mathbf{D}\nabla C - \mathbf{V}C)\cdot \mathbf{n}|_{\Gamma_2} = g_2 \ , \end{cases} \tag{11.4.20}$$

where ϕ is the porosity, Q_N is a natural sink/source term, and Q_P are the designed pumping rates defined by

$$Q_P(\mathbf{x}, t) = \sum_{i=1}^{I} q_{i,j} \delta(\mathbf{x} - \mathbf{x}_i), \text{when } t_{j-1} \le t < t_j, j = 1, 2, \cdots J. \tag{11.4.21}$$

All other notations in (11.4.20) have been explained in Example 5.9. Inputs of the model include PnT decisions, model parameters, and initial and boundary conditions. Outputs of the model are head and concentration distributions.

Model parameters

The components of the parameter vector $\boldsymbol{\theta}$ of the model may consist of parameterized hydraulic conductivity, porosity, and dispersivity. The geostatistical parameterization method is often used for heterogeneous aquifers. When initial and boundary conditions are known, the model-simulated concentration distribution can be represented by $C(\mathbf{q}_E, \boldsymbol{\theta}; \mathbf{x}, t)$. The uncertainty in $\boldsymbol{\theta}$ will lead to uncertainty in the concentration distribution and ultimately affect the remediation design. For example, a deviation in hydraulic conductivity may cause the model predicted high concentrations to appear at incorrect locations and/or times. As a result, the designed pumping wells for remediation will not be as effective as expected. In this case, the main purpose of data collection for inversion becomes minimizing the uncertainty of the optimal remediation design.

Objectives and constraints

Cost-effectiveness and remediation-effectiveness are usually used as two objectives for the PnT design. They are given, respectively, by

$$f_{\text{cost}}(\mathbf{q}_E) = c_{io} + \sum_{i=1}^{I} \sum_{j=1}^{J} c_i \left| q_{i,j} \right| (t_j - t_{j-1}) \tag{11.4.22}$$

and

$$f_{\text{reme}}(\mathbf{q}_E, \boldsymbol{\theta}) = \rho \sum_{m=1}^{M} \phi_m V_m C_m(\mathbf{q}_E, \boldsymbol{\theta}; t_f). \tag{11.4.23}$$

In (11.4.22), $f_{\text{cost}}(\mathbf{q}_E)$ is the total cost of remediation, c_{io} is the capital cost of ith well, and c_i is the cost of the pumping unit volume of water from the well. In (11.4.23), $f_{\text{reme}}(\mathbf{q}_E)$ is the total remaining contaminant mass in the aquifer at the ending time of the remediation, ρ is the density of the contaminant, M is the total number of elements, V_m and ϕ_m are the volume and porosity of mth element, respectively, and $C_m(\mathbf{q}_E, \boldsymbol{\theta}; t_f)$ is the model-calculated concentration of the element at the ending time. The optimal PnT design is a Pareto optimal solution of the following MOO problem

$$\mathbf{q}_E^o(\boldsymbol{\theta}) = \arg\min_{\mathbf{q}_E} \left\{ f_{\text{cost}}(\mathbf{q}_E), f_{\text{reme}}(\mathbf{q}_E, \boldsymbol{\theta}) \right\}. \tag{11.4.24}$$

According to the actual conditions of the case under study, various constraints can be added to the MOO problem. For example, the upper and lower bounds of the groundwater level in each pumping well, the maximum pumping capacity of each pumping well, and the maximum concentration at the ending time. For detailed discussions on PnT design, including decision variables, objectives and constraints, solution methods, and case studies, readers may refer to Ahlfeld et al. (1988), Huang and Mayer (1997), Zheng and Wang (1999), Mayer et al. (2002), Bear and Sun (1998), Chang et al. (2007), Singh and Chakrabarty (2010), and He et al. (2008).

Dealing with parameter uncertainty

Equation (11.4.24) clearly shows that the optimal remediation decision $\mathbf{q}_E^o$ depends on the model parameter vector $\boldsymbol{\theta}$. The uncertainty on $\boldsymbol{\theta}$ will propagate to $\mathbf{q}_E^o$ and further through $\mathbf{q}_E^o$ to the optimal remediation objectives

$$\begin{aligned} g_{E,1}(\boldsymbol{\theta}) &= f_{\text{cost}}[\mathbf{q}_E^{\mathbf{o}}(\boldsymbol{\theta})], \\ g_{E,2}(\boldsymbol{\theta}) &= f_{\text{reme}}[\mathbf{q}_E^{\mathbf{o}}(\boldsymbol{\theta})]. \end{aligned} \tag{11.4.25}$$

A model-based remediation design without considering the uncertainty of model parameters is unreliable. An example of combining parameter estimation and optimal PnT design is given in Sun et al. (1998). There are several approaches to dealing with this problem:

- If the model parameters are not estimated by inverse solution, we can use the method of this section to design an experiment aimed at minimizing the uncertainty of the remediation design. Let $\hat{\boldsymbol{\theta}}_D$ be the estimated parameter vector after the experiment is conducted and the inverse problem is solved. For PnT design, the objective vector of model application is defined by (11.4.25), i.e., $\mathbf{g}_E(\boldsymbol{\theta}_D) = \left\{ g_{E,1}(\boldsymbol{\theta}_D), g_{E,2}(\boldsymbol{\theta}_D) \right\}$. Let $\hat{\mathbf{g}}_E = \mathbf{g}_E(\hat{\boldsymbol{\theta}}_D)$, the optimal experimental design D should minimize a norm of $\text{Cov}[\hat{\mathbf{g}}_E]$ in (11.4.18), where the sensitivity matrix $\mathbf{J}_E = [\partial \mathbf{g}_E / \partial \boldsymbol{\theta}]$ can be calculated by a model differentiation method. Because of the nonlinearity of $\mathbf{g}_E(\boldsymbol{\theta})$, a robust optimal design is required.

- If the model parameters are estimated previously, we can use the estimated mean, $\hat{\boldsymbol{\theta}}$, as $\boldsymbol{\theta}$ to solve the PnT design problem (11.4.24) and use the covariance matrix $\mathrm{Cov}[\hat{\boldsymbol{\theta}}]$ of estimation to complete a posterior sensitivity analysis for assessing the reliability of the resulting PnT design (i.e., using (11.4.18) to calculate $\mathrm{Cov}[\mathbf{g}_E(\hat{\boldsymbol{\theta}})]$). If the confidence interval is not satisfied, we can design a new experiment for inversion to decrease the uncertainty of those parameter components that are most sensitive to the objectives of remediation. In practice, because PnT is a long-term activity, instead of designing a new experiment, we can use the data collected from a stress period of the remediation process to calibrate the simulation model and then modify the remediation design for the next stress period accordingly. For example, in the PnT design problem considered by Baú and Mayer (2008), the measurements of cumulative contaminant mass pumped out from each extraction well are used as new data to continuously condition the simulation model and modify future pumping strategies.
- Incorporate the model uncertainty into the MOO remediation design problem. In this type of methods, multiple realizations of the unknown parameter are generated and used to obtain the statistics of the objective functions for each candidate design $\mathbf{q}_E$. In the MOO problem (11.4.24), the value of $f_{\mathrm{reme}}(\mathbf{q}_E, \boldsymbol{\theta})$ is considered as a random variable with known mean and variance, and the Pareto optimal solution is in the probabilistic sense. Singh and Minsker (2008) present a so-called probabilistic multiobjective genetic algorithm for PnT design and applies their method to a case study.

11.4.3.3 INI Design for Decision Making

INI design for model prediction can be extended to INI design for general model applications. A parameter vector $\boldsymbol{\theta} \in P_{ad}$ is said to be interval identifiable for a given model application vector $\mathbf{g}(\boldsymbol{\theta})$ and an accuracy requirement $\gamma > 0$ if there is a design D such that

$$\left\| \mathbf{u}_D(\boldsymbol{\theta}') - \mathbf{u}_D(\boldsymbol{\theta}) \right\| < \delta \text{ implies } \left\| \mathbf{g}(\boldsymbol{\theta}') - \mathbf{g}(\boldsymbol{\theta}) \right\|_{\mathbb{G}} < \gamma, \tag{11.4.26}$$

where $\boldsymbol{\theta}' \in P_{ad}$, $\delta > 0$ is a number to be assigned, and $\mathbb{G}$ is the prediction space. When there is no model error, all discussions on INI design for model prediction can be derived in parallel for general model applications, including finding a sufficient INI design and a robust INI design. Figure 11.9 can still be used but changing the model prediction space to a model application space. Besides using weighted or generalized least squares norms, the accuracy requirement of a model application may be stated in L_∞-norm (Sect. 2.4.1) or in the form of the relative error. An example of using INI design for groundwater remediation design can be found in (Sun and Yeh 1990). In McPhee and Yeh (2004), INI design is used to solve a multiobjective groundwater resources management problem. We will return to INI design in the next chapter for model structure identification.

11.4.4 Design for Geostatistical Inversion

In geostatistical inversion method (Sect. 7.4.3), the unknown parameter $\theta(\mathbf{x})$ is regarded as a realization of a parameterized random field. After discretization, the random field becomes an N-dimensional random vector $\boldsymbol{\theta}$, where N is the number of nodes. Usually, it is assumed that $\boldsymbol{\theta}$ has a multi-Gaussian distribution, its mean is parameterized by trend parameters $\boldsymbol{\beta}$ and its covariance matrix is characterized by statistical parameters $\boldsymbol{\psi}$. After $\boldsymbol{\beta}$ and $\boldsymbol{\psi}$ are estimated by MLE with the data set $\mathbf{z}_D^{obs}$, the unknown parameter vector $\boldsymbol{\theta}$ is then estimated by co-kriging with the same data set. These estimations thus depend on how the data set is collected (i.e., an experimental design D). The OED for geostatistical inversion should minimize the uncertainty of these estimations. To date, however, studies on this topic are still very limited.

11.4.4.1 For trend Estimation

Assume the model is linear, i.e., $\mathbf{u}(\boldsymbol{\theta}) = \mathbf{A}\boldsymbol{\theta}$, and $\overline{\boldsymbol{\theta}} = \mathbf{X}\boldsymbol{\beta}$ then $\mathbf{u}(\overline{\boldsymbol{\theta}}) = \mathbf{A}\mathbf{X}\boldsymbol{\beta}$. In this case, the observation equation is $\mathbf{z}_D^{obs} = \mathbf{A}_D\mathbf{X}\boldsymbol{\beta} + \mathbf{e}_D$, where $\mathbf{z}_D^{obs}$ is an l-dimensional measurement vector, $\mathbf{A}_D$ is an $l \times N$ matrix, $\mathbf{X}$ is an $N \times k$ matrix, $\boldsymbol{\beta}$ is a k-dimensional trend vector, and $\mathbf{e}_D$ is an l-dimensional error vector with zero mean and covariance matrix $\mathbf{C}_D$. When $\boldsymbol{\beta}$ is a normally distributed random vector with prior distribution $p_0(\boldsymbol{\beta}) \sim N(\boldsymbol{\beta}_0, \mathbf{C}_{\beta\beta})$, according to Sect. 11.2.2, the estimated $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}}_D = (\mathbf{X}^T\mathbf{A}_D^T\mathbf{C}_D^{-1}\mathbf{A}_D\mathbf{X} + \mathbf{C}_{\beta\beta}^{-1})^{-1}(\mathbf{X}^T\mathbf{A}_D^T\mathbf{C}_D^{-1}\mathbf{z}_D^{obs} + \mathbf{C}_{\beta\beta}^{-1}\boldsymbol{\beta}_0) \qquad (11.4.27)$$

and the covariance of estimation is

$$\mathrm{Cov}(\hat{\boldsymbol{\beta}}_D) = (\mathbf{X}^T\mathbf{A}_D^T\mathbf{C}_D^{-1}\mathbf{A}_D\mathbf{X} + \mathbf{C}_{\beta\beta}^{-1})^{-1}. \qquad (11.4.28)$$

The OED seeks to minimize a norm of $\mathrm{Cov}(\hat{\boldsymbol{\beta}}_D)$. By defining information matrix $\mathbf{M}_\beta(D) = \mathbf{X}^T\mathbf{A}_D^T\mathbf{C}_D^{-1}\mathbf{A}_D\mathbf{X}$, all Bayesian optimality criteria in Sect. 11.2.2 can be applied to the estimation of $\boldsymbol{\beta}$. For example, the Bayesian D-optimality and A-optimality are given, respectively, by

$$D_{BD}^* = \arg\max_D \log\det[\mathbf{M}_\beta(D) + \mathbf{C}_{\beta\beta}^{-1}], D \in \{D_{ad}\} \qquad (11.4.29)$$

and

$$D_{BA}^* = \arg\min_D \mathrm{trace}[\mathbf{M}_\beta(D) + \mathbf{C}_{\beta\beta}^{-1}]^{-1},\ D \in \{D_{ad}\}. \qquad (11.4.30)$$

When the dimension k of $\boldsymbol{\beta}$ is not high, a cost-effective OED can be found with affordable computational effort.

11.4.4.2 For Co-Kriging Estimation

The problem of designing an experiment for estimating statistical parameters $\boldsymbol{\psi}$ through inversing measurements $\mathbf{z}_D^{obs}$ is very difficult because the relationship between them is highly nonlinear and nonunique. Usually, $\boldsymbol{\psi}$ is estimated by fitting a variogram model with some measurements of the estimated $\theta(\mathbf{x})$ directly (Sect. 6.2.3) other than by model inversion. Detailed discussion on optimal sampling design for variogram parameter estimation can be found in Müller (2007).

11.4.4.3 For Geostatistical Inversion

In geostatistical inversion, the collected data $(\mathbf{z}_D^{obs})$ are used twice, first to determine a random field $(\boldsymbol{\beta}$ and $\boldsymbol{\psi})$ and then to determine a realization of the field. A design that is optimal for estimating the random field may not be the optimal one for estimating a realization.

In principle, the optimality criteria used for $\boldsymbol{\beta}$ estimation can also be used for $\boldsymbol{\theta}$ estimation. But, there are some important differences: The dimensions of $\boldsymbol{\theta}$ are very high that may make the computational effort unaffordable and, because β is estimated from data, its uncertainty will propagate to $\boldsymbol{\theta}$ and thus increase the uncertainty of $\boldsymbol{\theta}$. In this case, the prior covariance matrix of $\boldsymbol{\theta}$, $\mathbf{C}_\mathrm{P}$, should be replaced by the following generalized covariance matrix (Kitanidis 1993):

$$\mathbf{G}_{\theta\theta} = \mathbf{C}_P + \mathbf{X}\mathbf{C}_{\beta\beta}\mathbf{X}^T. \tag{11.4.31}$$

Using $\mathbf{G}_{\theta\theta}$ to replace $\mathbf{C}_{\theta\theta}$ in (11.2.15), we will have

$$\mathrm{Cov}(\hat{\boldsymbol{\theta}}_D) = [\mathbf{A}_D^T\mathbf{C}_D^{-1}\mathbf{A}_D + (\mathbf{C}_P + \mathbf{X}\mathbf{C}_{\beta\beta}\mathbf{X}^T)^{-1}]^{-1}. \tag{11.4.32}$$

The optimality criteria of design for estimating $\hat{\boldsymbol{\theta}}_D$ then can be obtained. For example, the D-optimality and A-optimality can be obtained from (11.2.17) and (11.2.18), respectively. In Nowak et al. (2010), these and other optimality criteria for geostatistical inversion are reviewed and compared. The authors commented that D-optimality is unacceptable computationally because of the large size of matrix $\mathbf{G}_{\theta\theta}$, while the A-optimality is relatively efficient and valid for non-Gaussian and nonlinear problems.

There are effective methods for finding a suboptimal retrospective design for geostatistical inversion. On the one hand, we can find terms in (7.4.57) that make very small contributions to the summation. Then we can find which observation points (location/time) are associated with these terms. Deleting those observation points from the design will cause only insignificant effect to the variance of estimation. For example, if the data provided by two observation points are strongly correlated, then one of them can be deleted. On the other hand, when new observation points are needed, they should be selected such that the absolute value of the summation in (7.4.57) will be increased as large as possible after their contribution is added.

11.4.4.4 Observation Design for the Pilot Point Method

OED can be used to the pilot point method (Sect. 7.4.4) to solve the following two design problems:

- Assuming that the number and locations of pilot points are given, find the OED for state observation.
- Assuming that the data of state observation are given, find the optimal locations for a certain number of pilot points.

The first problem is an experimental design problem for inverse solution, the second one is needed in the pilot point algorithms, and both of them can be solved by the classical OED approach. For the first problem, the Jacobian matrix used to calculate the FIM in (11.3.4) is given by

$$\mathbf{J}_D(\boldsymbol{\theta}) = [\partial \mathbf{u}_D / \partial \boldsymbol{\theta}], \tag{11.4.33}$$

where D is a design that determines how n state observations $\mathbf{u}_D(\boldsymbol{\theta})$ will be generated, and $\boldsymbol{\theta}$ is the unknown parameter vector associated with the m given pilot points, which is independent of the design.

For the second problem, the Jacobian matrix used in the FIM becomes

$$\mathbf{J}_D(\boldsymbol{\theta}) = [\partial \mathbf{u} / \partial \boldsymbol{\theta}_D], \tag{11.4.34}$$

where D is a design that determines how m pilot points are located, $\boldsymbol{\theta}_D$ is the unknown parameter vector associated with these pilot points, and $\mathbf{u}(\boldsymbol{\theta}_D)$ is the model outputs corresponding to the n given state observations, which are independent of the design.

Once the FIM is calculated and an optimality criterion is selected, a local optimal design can then be obtained by solving an optimization problem. For example, in Jung et al. (2011), the D-optimality criterion is used to solve the second problem mentioned above a numerical example is given.

In practice, however, both the number of data (n) and the number of pilot points (m) are unknown and should be considered as a part of design. A more general design problem for the pilot point method is to design an observation network without knowing m and n. The number m depends on the complexity of the model structure and the number n, in turn, depends on m because identifying a more complex model needs more data. We will consider such a design problem related to the determination of model complexity in the next chapter.

11.4.5 Design for Model Structure Identification

So far in this chapter, we have assumed that there is no model error or the model error can be neglected. But this assumption is generally impractical for modeling a distributed parameter system or a large-scale system encountered in EWR modeling. After parameterization, model calibration requires the identification of not only physical parameters, but also model structure parameters and/or hyperparameters. The ODE problem for extended inverse problem (EIP) is thus presented. In Chap. 7, the solution of EIP for identifying an entire model is given by

$$(\hat{\mathbf{v}}, \hat{\boldsymbol{\theta}}) = \arg\min_{(\mathbf{v},\boldsymbol{\theta})}\left\{\left\|\mathbf{u}_D(\nu, \boldsymbol{\theta}) - \mathbf{u}_D^{obs}\right\| + \alpha R(\mathbf{v}, \boldsymbol{\theta})\right\}, \quad (\mathbf{v}, \boldsymbol{\theta}) \in \mathfrak{M}_{ad}, \tag{11.4.35}$$

where $\mathbf{v}$ is a k-dimensional shape vector characterizing the model structure, $\boldsymbol{\theta}$ is an m-dimensional parameter vector associated with the structure, and $\alpha R(\mathbf{v}, \boldsymbol{\theta})$ is a regularization term including prior information. The observation equation for inversion can be written as

$$\mathbf{u}_D^{obs} = \mathbf{u}_D(\mathbf{v}, \boldsymbol{\theta}) + \mathbf{e}_D. \tag{11.4.36}$$

After linearization and using $(\mathbf{v}, \boldsymbol{\theta})$ to replace $\boldsymbol{\theta}$ in (11.3.3), we obtain

$$\mathrm{Cov}(\hat{\mathbf{v}}_D, \hat{\boldsymbol{\theta}}_D) = [\mathbf{J}_D^T(\mathbf{v}, \boldsymbol{\theta})\mathbf{C}_D^{-1}\mathbf{J}_D(\mathbf{v}, \boldsymbol{\theta})]^{-1}, \tag{11.4.37}$$

where $\mathbf{J}_D(\mathbf{v}, \boldsymbol{\theta}) = \left[\frac{\partial \mathbf{u}_D}{\partial \mathbf{v}} \; \frac{\partial \mathbf{u}_D}{\partial \boldsymbol{\theta}}\right]$ is an $n \times (k + m)$ matrix. According to Eqs. (7.2.9) and (7.2.10), all elements of the matrix can be calculated by

$$\frac{\partial u_{D,l}}{\partial \nu_j} = \sum_{i=1}^{N} \frac{\partial u_{D,l}}{\partial \theta_{N,i}} \frac{\partial \theta_{N,i}}{\partial \nu_j}, \; l = 1, 2, \cdots, n, \; j = 1, 2, \cdots, k \tag{11.4.38}$$

and

$$\frac{\partial u_{D,l}}{\partial \theta_j} = \sum_{i=1}^{N} \frac{\partial u_{D,l}}{\partial \theta_{N,i}} \frac{\partial \theta_{N,i}}{\partial \theta_j}, \; l = 1, 2, \cdots, n, \; j = 1, 2, \cdots, m, \tag{11.4.39}$$

where N is the node number of the numerical model, $\theta_{N,i}$ is the parameter value at node i, and all sensitivity coefficients $(\partial u_{D,l} / \partial \theta_{N,i})$ can be obtained effectively by the adjoint-state method of model differentiation. After obtaining (11.4.37), at least in principle, we can define the FIM and then define the alphabetic optimality criteria as what we have done in Sect. 11.3.1. But, such an extension is not very useful because of the following inherent difficulties: (i) Model output $\mathbf{u}_D(\mathbf{v}, \boldsymbol{\theta})$ could be highly nonlinear with respect to shape parameters, a linearization-based

OED may become useless; (ii) a small change in shape parameters may cause a significant change to the OED, a robust design may not exist; and (iii) because shape parameters may have no physical meaning, we do not know how to make them close their "true" values. Therefore, during the design stage, we should avoid using shape parameters unless they have certain physical meanings, for example, when a structure is parameterized by zonation, the boundaries between zones that we want to locate can be explained in geology.

In the case that a distributed parameter needs to be estimated and no prior structure information on it is available, a multiscale design process can be used, for which, the objective of the design is to estimate the nodal parameter values over a coarse and homogeneous grid (see Fig. 7.6). Because there is no structural parameter involved, classical methods of OED can be used in this case. After the designed data is collected, of course, any parameterization method in Chap. 7 can be used for solving the EIP. The key of using this method is to determine an appropriate scale of the grid or an appropriate resolution of the identified parameter. The cost of the experiment will significantly increase with the increase of the node number. In other words, we have to pay for the "cost of complexity" (Hjalmarsson 2009; Rojas et al. 2011).

We have shown in Sects. 11.4.2 and 11.4.3, when the ultimate objective of model construction is model prediction or model application rather than parameter estimation, the OED will become goal (or objective)-dependent. In fact, when the structure of the real system is complex and unknown, determining an appropriate model complexity according to the given objective of model application is the most important thing and needs to be considered first in model construction. Developing goal (or objective)-oriented models is the topic of the next chapter, in which goal-oriented experimental design is completed after goal-oriented model complexity is determined.

11.5 Review Questions

1. Try to list as many objectives of OED for model construction as possible.
2. Use geometric terms to explain the meanings of D-optimality, A-optimality, and E-optimality. Explain how these explanations are obtained.
3. Derive Eqs. (11.2.14) and (11.2.15). What assumptions are needed for the derivation?
4. In what cases the sequential design method is feasible and acceptable? And, in what cases, a global OED becomes necessary?
5. What are the limitations of the Bayesian experimental design? Is it a robust design for nonlinear model inversion? What are the perspective ways to make this design method practical?
6. Explain why the design defined in Eq. (11.3.15) is robust. Compare the algorithms of this max-min problem to that of the min-min problem for solving the EIP.

7. Use Fig. 11.3 to explain: (1) whats is INI, (2) how the numbers ε and δ are determined, (3) how INI depends on a design, and (4) why Eq. (11.3.24) is equivalent to Eq. (11.3.23). Moreover, is $S_D(\boldsymbol{\theta}) \geq \delta$ a sufficient condition or a necessary condition? How the observation error is counted in INI? In the concept, what are the differences between the INI design and the optimal design?
8. Compare the computational effort for generating the *L*-curve in Fig. 11.7 by (1) solving the biobjective optimization problem in Eq. (11.4.2) or (11.4.3), and (2) solving the cost-effective INI design problem in Eq. (11.58).
9. Give the details of deriving the G-optimal design in Eq. (11.4.9). Then, formulate the robust G-optimal design.
10. Use Fig. 11.9 to explain how model inversion and prediction reliability are related to the sufficiency of data in the INI design.
11. Give an outline on how to use INI to design a PnT system (or any model-based decision-making problem considered in your study area).
12. Formulate D-optimality criteria, respectively, for solving the two kinds of OED problems associated with the pilot point method.

Chapter 12
Goal-Oriented Modeling

Two criteria are traditionally used for model parameter estimation and model structure identification: (C-1) fitting observation data; and (C-2) honoring prior information. As we have shown in Chaps. 3 and 7, these two criteria cannot determine a model uniquely by solving either classical inverse problem (CIP) or extended inverse problem (EIP) when observation error and model error exist. Models that cannot be rejected by prior information and observed data are called *data-acceptable* models (Sambridge 2001). In EWR modeling, because the real system structure is complex and unknown, there may be infinite combinations of model structures and model parameters that can fit the existing data equally well. Different modelers may construct different models for the same system based on the same data (Refsgaard et al. 2006). As we explained in Chap. 10, this type of model nonuniqueness is called *equifinality* by Beven and coworkers (Beven and Binley 1992).

Ideally, a model of a system should be able to produce the same responses as the real system does for any excitation (see Chap. 1), but a data-acceptable model is tested only by incomplete system responses (observed data) and limited excitations. As a result, there is no guarantee that it would produce the same responses as the real system does for all other excitations unseen during model training or calibration. In other words, a model that cannot be rejected by existing data may not be acceptable for prediction and management purposes. In Chap. 8, the structural risk minimization principle was used in support vector machine (SVM) algorithms to bound the predictive error, whereas in Chap. 10, we attempted to find all data-acceptable models for uncertainty analysis. We may collect more data to decrease the number of data-acceptable models and thus to decrease the model uncertainty. But in EWR modeling, we cannot find all data-acceptable models for uncertainty analysis because the number of acceptable model structures is infinite and, more importantly, the real structure of the modeled system is usually not accessible because of its complexity and multiscale variability. As a consequence, an appropriate model structure cannot be determined before a model is constructed, and a posterior uncertainty analysis cannot guarantee the model's reliability. From the discussion herein, we can see that constructing a universally reliable EWR model is a very difficult or even infeasible task, although new data collection techniques and computational capacity will keep enhancing our capability to create more realistic models in the future.

"Goal-oriented modeling" or "objective-oriented modeling" is a paradigm that attempts to find a representative model (or a metamodel) for the system under study such that the model is only useful for some preset goals or objectives of model application. A goal-oriented model is not a model in the original sense because it gives the same "responses" as the real system does only for some "specified excitations," instead of for "any excitations." But a goal-oriented model is useful in practice and can be constructed with limited data. In this chapter, we will give a general introduction on how to construct goal-oriented models, especially, for large-scale distributed parameter systems.

The *goal-oriented forward problem* (GFP) introduced in Sect. 12.1 assumes that the model parameters are known and considers how to solve the forward problem more effectively and accurately for a preset goal, including adaptive model order reduction, adaptive mesh optimization, and adaptive upscaling or homogenization (Bui-Thanh et al. 2007; Fang et al. 2010; Oden and Vemaganti 2000). In Sect. 12.2, the *goal-oriented inverse problem* (GIP) presented in Sun and Sun (2002), Sun (2005), and Sun and Yeh (2007a) is formulated by three criteria: In addition to (C-1) and (C-2), it requires also (C-3) (i.e., the constructed model should be reliable for the preset goals of model application). Section 12.3 gives theory, methods, and algorithms for the solution of GIP, including the discussions on identifiability, model complexity, and data sufficiency. Finally, Sect. 12.4 is contributed to the problem of goal-oriented experimental design, in which the interval identifiability (INI) design method is extended to include the determination of the model structure.

12.1 Goal-Oriented Forward Problem

12.1.1 Goal-Oriented Model Reduction

Before a model is constructed, the modeler should know why the model is needed and how it will be used. When a model is used to solve a goal-oriented engineering problem (Van Lamsweerde 2001), goals (or objectives) and their accuracy requirements provide important information to guide the construction of a customized model. The following goals are commonly seen in EWR engineering:

- *Prediction*. To predict the values of state variables at specified locations and/or times when planned changes to the system are taken place. For example, to estimate the peak concentration in a well when the concentration of inflow water from a boundary is changed to a specified value.
- *Management*. To control a system to reach a predetermined state. For example, to decrease the contaminant level of a site to a specified value.

- *Decision making*. To compare different scenarios for controlling a system according to certain criteria. For example, to assess the feasibility of remediation alternatives for cleaning-up a contaminated site.
- *Design*. To determine the optimal design variables for engineering purposes. For example, to design a monitoring network or a remediation plan.

Presently, a numerical model used for modeling a complex and large-scale EWR system may have $10^4 \sim 10^6$ or even more nodes. When such a high-order model is used for optimal design or decision making, the computational time and cost may not be affordable. GFP attempts to make the forward solution more effective and sufficiently accurate for a preset goal (or goals) of model application. A preset goal of model application may be represented in general by a function $g(u, \theta, q)$, where *u* is the system state, θ is the system parameter, and *q* is the control or decision variable. For a distributed system, $u = u(\theta, q, \mathbf{x}, t)$ is the forward solution of a model $L(u, \theta, q, \mathbf{x}, t) = 0$. In most practical cases, θ and q do not appear in the goal expression explicitly, but they impact the goal implicitly through *u* (i.e., $g = g[u(\theta, q)]$).

As shown in Sect. 1.2.2, when a numerical method (finite element method (FEM), for example) is used for solving partial differential equations (PDEs), after spatial discretization, the forward problem becomes the solution of a set of ordinary differential equations (ODEs). The entire goal-oriented model consists of a dynamic system model with a preset goal, viz.

$$\begin{aligned} &\mathbf{E}\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, \boldsymbol{\theta}, \mathbf{q}), \qquad \mathbf{u}(0) = \mathbf{u}_0 \\ &\mathbf{g} = \mathbf{g}(\mathbf{u}, \boldsymbol{\theta}, \mathbf{q}) \end{aligned} \tag{12.1.1}$$

where $\mathbf{u} \in \mathbb{R}^N$ is the unknown state vector associated with *N* nodes and $\mathbf{u}_0$ is its initial values, $\boldsymbol{\theta} \in \mathbb{R}^M$ is the parameter vector associated with *M* elements, and $\mathbf{q} \in \mathbb{R}^r$ is the control vector; $\boldsymbol{\theta}$ and $\mathbf{q}$ are model inputs, and **g** is the model outputs. The coefficient matrix $\mathbf{E}$ may depend on parameter vector $\boldsymbol{\theta}$ and boundary conditions. For nonlinear models, $\mathbf{E}$ depends also on the unknown state vector. The grid scale of the numerical model (12.1.1) is assumed to be fine enough (or the node number *N* is large enough) such that $\mathbf{g}(\mathbf{u}, \boldsymbol{\theta}, \mathbf{q}) \in \mathbb{R}^p$ can be used accurately as an approximation of $g(u, \theta, q)$. Now, we want to find a reduced-order model to replace the original high-order model (12.1.1) for the same goal. Let

$$\begin{aligned} &\hat{\mathbf{E}}\frac{d\hat{\mathbf{u}}}{dt} = \hat{\mathbf{f}}(\hat{\mathbf{u}}, \hat{\boldsymbol{\theta}}, \mathbf{q}), \qquad \hat{\mathbf{u}}(0) = \hat{\mathbf{u}}_0 \\ &\hat{\mathbf{g}} = \hat{\mathbf{g}}(\hat{\mathbf{u}}, \hat{\boldsymbol{\theta}}, \mathbf{q}) \end{aligned} \tag{12.1.2}$$

where $\hat{\mathbf{u}} \in \mathbb{R}^k (k << N)$ and $\hat{\boldsymbol{\theta}} \in \mathbb{R}^m (m << M)$. Model (12.1.2) is called an acceptable model for goal $\mathbf{g}$ if the following reliability requirement is satisfied

$$\left\| \mathbf{g}(\mathbf{u}, \boldsymbol{\theta}, \mathbf{q}) - \hat{\mathbf{g}}(\hat{\mathbf{u}}, \hat{\boldsymbol{\theta}}, \mathbf{q}) \right\| < \varepsilon_g, \quad \text{for } \boldsymbol{\theta} \in \Theta, \ \mathbf{q} \in Q \tag{12.1.3}$$

Here, $\|\cdot\|$ is a norm furnished in the goal space $\mathbb{G}$, ε is a preset tolerance of error in goal, $\Theta \, \text{and} \, Q$ are given ranges of $\boldsymbol{\theta}$ and $\mathbf{q}$, respectively. In the study of a goal-oriented optimal control or decision-making problem, $\boldsymbol{\theta}$ is fixed and the reduced-order model is required to be acceptable for all $\mathbf{q} \in Q$; while in the study of a goal-oriented parameter uncertainty problem, $\mathbf{q}$ is fixed and the reduced-order model is required to be acceptable for all $\boldsymbol{\theta} \in \Theta$. Obviously, acceptable models for a goal are nonunique, and, for the same system, a model that is acceptable for one goal may not be acceptable for other goals. Several adaptive methods have been developed that allow us to find the lowest-order acceptable model for a given goal.

Example 12.1 *A goal-oriented linear dynamic system control problem*
The following model can be seen in an optimal control problem when the dynamic system model (12.1.1) and the goal $\mathbf{g}(\mathbf{u}, \mathbf{q})$ are both linear

$$\begin{aligned} &\mathbf{E}\frac{d\mathbf{u}}{dt} = \mathbf{A}\mathbf{u} + \mathbf{B}\mathbf{q}, \ \mathbf{u}(0) = \mathbf{0} \\ &\mathbf{g} = \mathbf{C}\mathbf{u} + \mathbf{D}\mathbf{q} \end{aligned} \tag{12.1.4}$$

where $\mathbf{E}, \mathbf{A}, \mathbf{C}$ and $\mathbf{D}$ are $N \times N, N \times N, p \times N, \text{and} \ p \times r$ matrices, respectively. The reduced-order system (12.1.2) and the corresponding goal $\hat{\mathbf{g}}(\hat{\mathbf{u}}, \mathbf{q})$ are given by

$$\begin{aligned} &\hat{\mathbf{E}}\frac{d\hat{\mathbf{u}}}{dt} = \hat{\mathbf{A}}\hat{\mathbf{u}} + \hat{\mathbf{B}}\mathbf{q}, \hat{\mathbf{u}}(0) = \mathbf{0} \\ &\hat{\mathbf{g}} = \hat{\mathbf{C}}\hat{\mathbf{u}} + \hat{\mathbf{D}}\mathbf{q} \end{aligned} \tag{12.1.5}$$

where $\hat{\mathbf{E}}, \hat{\mathbf{A}}, \hat{\mathbf{C}}$ and $\hat{\mathbf{D}}$ are $k \times k, k \times k, p \times k, \text{and} \, p \times r$ matrices, respectively. Using Laplace transformation $[\mathbf{u}(t) \rightarrow \mathbf{u}(s), \text{d}\mathbf{u}(t) \, / \, dt \rightarrow s\mathbf{u}(s)]$ to the linear system in (12.1.4), we have $s\mathbf{E}\mathbf{u}(s) = \mathbf{A}\mathbf{u}(s) + \mathbf{B}\mathbf{q}(s)$. Thus, $\mathbf{u}(s) = (s\mathbf{E} - \mathbf{A})^{-1}\mathbf{B}\mathbf{q}(s)$, and $\mathbf{g}(s) = \mathbf{C}\mathbf{u}(s) + \mathbf{D}\mathbf{q}(s) = \mathbf{G}\mathbf{q}(s)$, where

$$\mathbf{G} = \mathbf{C}(s\mathbf{E} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$$

is called the transfer function. Applying the same process to the reduced-order linear model (12.1.5), we obtain

$$\|\mathbf{g} - \hat{\mathbf{g}}\| = \|\mathbf{G}\mathbf{q} - \hat{\mathbf{G}}\mathbf{q}\| \leq \|\mathbf{G} - \hat{\mathbf{G}}\| \|\mathbf{q}\|, \tag{12.1.6}$$

where $\hat{\mathbf{G}} = \hat{\mathbf{C}}(s\hat{\mathbf{E}} - \hat{\mathbf{A}})^{-1}\hat{\mathbf{B}} + \hat{\mathbf{D}}$. This equation provides an upper bound of the model error in goal $\mathbf{g}$ for each reduced-order k.

12.1.2 Goal-Oriented Multiscale Modeling

A large-scale EWR numerical model may need a huge number of nodes in order to describe the details of a distributed parameter at fine scales. In this case, finding a reduced-order model for a preset goal depends on how the heterogeneity of the parameter affects the accuracy of the calculated goal. The adaptive mesh refinement process described in Sect. 7.3 for multiscale inversion can be used here for multiscale goal-oriented modeling. As in Sect. 7.3, the adjoint-state method of sensitivity analysis plays an important role in the selection of appropriate elements for refinement. The sensitivity of goal **g** to a parameter component θ_j in the fine scale is given by

$$\frac{\partial \mathbf{g}}{\partial \theta_j} = \sum_{i=1}^{N} \frac{\partial \mathbf{g}}{\partial u_i} \frac{\partial u_i}{\partial \theta_j}. \qquad (12.1.7)$$

It means that if **g** is insensitive to u_i $(\partial \mathbf{g} / \partial u_i \text{is small})$ and/or u_i is insensitive to $\theta_j(\partial u_i / \partial \theta_j \text{ is small})$ in a region, we can coarsen the mesh in that region to decrease the computational effort, otherwise, we should use a refined mesh to increase the accuracy of **g**. As shown in Fig. 12.1, although $\boldsymbol{\theta}$ and $\hat{\boldsymbol{\theta}}$ are not close in the parameter space and $\mathbf{u}$ and $\hat{\mathbf{u}}$ are not close in the state space, $\mathbf{g}$ and $\hat{\mathbf{g}}$ might still be close in the goal space.

The goal-oriented adaptive mesh refinement process consists of the following major steps:

1. Start with a coarse mesh, estimate the model error in state, $\|\mathbf{u} - \hat{\mathbf{u}}\|$, and then estimate the model error in goal, $\|\mathbf{g} - \hat{\mathbf{g}}\|$, through the adjoint sensitivity analysis.
2. If the model error in goal is less than the given tolerance, stop.
3. Refine the elements where the fine-scale parameter heterogeneity gives the most significant effect to the goal; then, return to step 1 with the refined mesh.

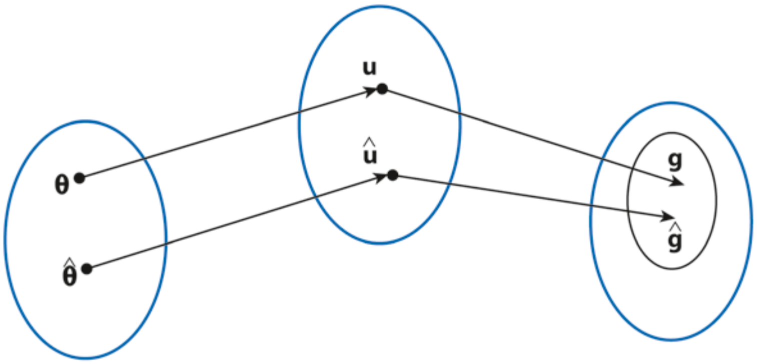


Fig. 12.1 Two parameter vectors are not close in the parameter space, and their images in the state space are not close either, but their images in the goal space are as close as required

Theories, algorithms, and applications exist in the literature on the estimation of error in goal $\left\|\mathbf{g}-\hat{\mathbf{g}}\right\|$ and on the implementation of the adaptive mesh refinement process. For a detailed discussion on this topic, readers may refer to Oden and Zohdi (1997), Oden and Vemaganti (2000), Venditti and Darmofal (2000), among others. Recent studies and reviews can be found in Fang et al. (2010), Prudhomme and Oden (2011), Jhurani and Demkowicz (2012).

12.1.3 Goal-Oriented Principal Component Analysis

The PCA- or proper orthogonal decomposition (POD)-based model order reduction method that was used for effective inversion in Sect. 6.4.3 can also be used here for effective goal-oriented modeling. Depending on the problem being considered, snapshots and accurate $\mathbf{g}(\mathbf{u},\boldsymbol{\theta},\mathbf{q})$ values are obtained by solving the fine-scale model with different $\boldsymbol{\theta}$ and/or $\mathbf{q}$ sampled from their given ranges. For each reduced-order model obtained by truncated singular value decomposition (TSVD), we can calculate $\hat{\mathbf{g}}(\hat{\mathbf{u}},\hat{\boldsymbol{\theta}},\mathbf{q})$ and check the accuracy requirement (12.1.3) to determine if the model order (number of PCs) should be increased or decreased. Detailed discussions and recent studies on this topic can be found in Bui-Thanh et al. (2007), Fang et al. (2010), and Carlberg and Farhat (2011).

In summary, it is critical to be able to control the level of model complexity for distributed parameter models. When we introduced model order reduction techniques in Chap. 6, we did not answer when to stop increasing the model order. When we introduced multiscale modeling in Chap. 7, we did not answer when we should stop refining the grid. In Chap. 8, the model complexity is reduced, but we did not answer how to determine the model dimensions. Now, we find that these questions can be answered by incorporating the goals of model application and their accuracy requirement as the third criterion into the inverse problem formulation, not only for parameter estimation, but also for structure identification.

12.2 Goal-Oriented Inverse Problem

12.2.1 Basic Problems in Model Construction

There are several basic but challenging problems in the construction of a useful model, as shown in the following hypothetical example.

Example 12.2 *A goal-oriented prediction problem (from Sun and Yeh 2007a)*
Consider again the 2-D confined aquifer used in Example 5.14 and Example 11.7. As shown in Fig. 12.2, it is 3000 m long in the *x* direction, 2000 m wide in the *y* direction, and with a constant thickness of 20 m. The head is fixed at 100 m on boundary

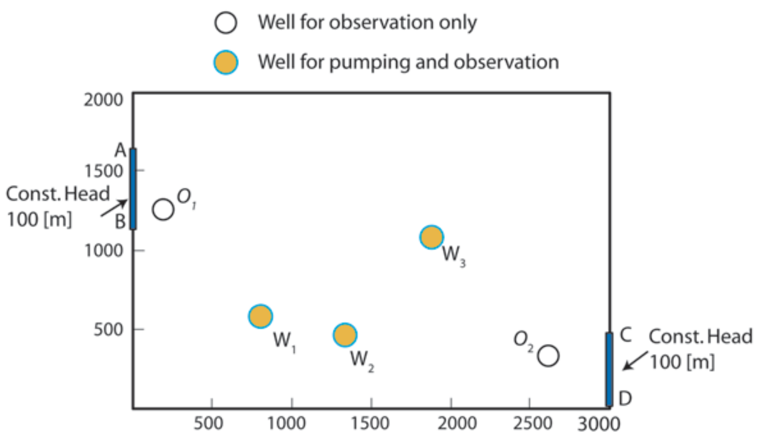


Fig. 12.2 Problem configuration of the numerical example

sections AB and CD and there is no flow across other boundary sections. The initial head is 100 m everywhere.

The goal (**g**) of constructing a groundwater flow model for this aquifer is to predict the steady-state hydraulic heads in three pumping wells W_1, W_2, and W_3 when their pumping rates (the control variable **q**) reach their planned maximum values 2000, 10,000, and 4000m^3/day, respectively. It requires that the difference between the true (unknown) and the model-predicted heads in the three wells be less than 1.0 m (the accuracy tolerance ε_g).

Assume that from prior information, the values of hydraulic conductivity $K(x,y)$ (the parameter θ) in the aquifer may vary from a lower bound of 10 m/day to an upper bound of 50 m/day, but its structure is unknown. The true structure of the aquifer may be a continuously varying structure, a randomly distributed structure with or without a trend, or a discrete structure with or without fractures. The number of possible structures is actually infinite. If we do not have more information to narrow the possible range of $K(x,y)$, the uncertainty of model prediction can be estimated directly by running the prediction model twice using the lower bound $K(x,y)=10$ m/day and the upper bound $K(x,y)=50$ m/day, respectively. The difference in model outputs between the two runs gives the following ranges of model-predicted steady-state heads in the three wells:

$$49.3\text{m} \le h_E(W_1) \le 90.0\text{m},\ 25.3\text{m} \le h_E(W_2) \le 85.2\text{m},$$
$$47.7\text{m} \le h_E(W_3) \le 89.7\text{m}$$

These uncertainty ranges are obviously unacceptable based on the given accuracy requirement ($\varepsilon_g = 1.0\,\text{m}$). In order to decrease the uncertainty of model prediction, a pumping test is conducted that consists of (i) pumping 500, 2000, and 1000 m^3/day, respectively, from wells W_1, W_2, and W_3; (ii) five observation locations at W_1, W_2, W_3, O_1, and O_2; and (iii) heads at these locations are measured at time $t = 0.05$, 0.1, 0.5, 1.0, and 3.0 day, respectively. These data will then be used to estimate the unknown hydraulic conductivity $K(x, y)$.

At this point, modelers may ask the following questions:

- How do we parameterize the structure of $K(x, y)$ without knowing its true structure?
- How can we assure that the identified $K(x, y)$ is reliable for the given goal of model prediction?
- Are the existing data sufficient for constructing a reliable model?
- If the existing data are insufficient, how do we design a cost-efficient test for new data collection?

These questions are generally referred to as the *model complexity problem*, the *model reliability problem*, the *data sufficiency problem*, and the *experimental design problem*, respectively. Of the four basic problems of model construction, answering the first one (i.e., model complexity) is the key. Without knowing an appropriate level of model complexity, the model reliability cannot be assessed because the model error is unknown, the data sufficiency cannot be judged because a more complex model needs more data to calibrate, and, finally, without knowing the data sufficiency, designing a cost-effective experiment would become meaningless.

But, what is "an appropriate level of model complexity" when the true structure of the modeled system is unknown? Model complexity, of course, depends on the complexity of the modeled system and available data. But an appropriate model complexity should depend more on the goal of model application and its accuracy requirement. CIP assumes that the model structure is predetermined (called a conceptual model) and only a number of scalar model parameters are identified by data. The model complexity problem is not considered. In EIP, the model structure is obtained by an optimization process that extracts as much as possible information from data and prior information to avoid over- and under-parameterization. A model structure determined by EIP is "appropriate" for the existing data, but may not be appropriate for prediction or other goals of model application. In other words, a model generated by solving CIP or EIP is a data-acceptable model, but a data-acceptable model may not be a goal-acceptable one. In contrast, GIP uses the goal of model application and its accuracy requirement to determine an appropriate level of complexity in the model structure and then determines the data sufficiency.

12.2.2 Formulation of GIP

Following the notations from Chap. 7, let us use $(\mathbf{S}, \boldsymbol{\theta})$ to denote a distributed parameter model to be identified, where $\mathbf{S}$ represents a model structure and $\boldsymbol{\theta}$ is the model parameter vector associated with the structure. GIP uses the following three kinds of information for model identification:

- Prior information, such as an initial estimation of the model, $(\mathbf{S}^0, \boldsymbol{\theta}^0)$, or an admissible region determined by upper and lower bounds, $\underline{\boldsymbol{\theta}} \leq \boldsymbol{\theta} \leq \overline{\boldsymbol{\theta}}$.
- A set of state observations, $\mathbf{u}_D^{obs}$, taken at different times and/or locations. These data are the observed system responses to an excitation characterized by a control variable $\mathbf{q}_D$ used to set, for example, boundary conditions and/or sources terms.
- A set of goals of model application, $\mathbf{g}(\mathbf{u})$, and their accuracy requirement:

$$\left\| \mathbf{g}[\mathbf{u}(\mathbf{S}^t, \boldsymbol{\theta}^t, \mathbf{q}_E)] - \mathbf{g}[\mathbf{u}(\mathbf{S}, \boldsymbol{\theta}, \mathbf{q}_E)] \right\| < \varepsilon_g. \tag{12.2.1}$$

Here, $(\mathbf{S}^t, \boldsymbol{\theta}^t)$ is the true (accurate) model of the unknown parameter, $\mathbf{q}_E$ is a planned control variable used to generate the required state $\mathbf{u}$ for calculating $\mathbf{g}(\mathbf{u})$, ε_g is a preset accuracy requirement, $\mathbf{g}[\mathbf{u}(\mathbf{S}^t, \boldsymbol{\theta}^t, \mathbf{q}_E)]$ is the true $\mathbf{g}(\mathbf{u})$ but unknown, and $\mathbf{g}[\mathbf{u}(\mathbf{S}, \boldsymbol{\theta}, \mathbf{q}_E)]$ is the model-generated $\mathbf{g}(\mathbf{u})$. It is assumed that $(\mathbf{S}^t, \boldsymbol{\theta}^t)$ and $(\mathbf{S}, \boldsymbol{\theta})$ include all nodes or elements of the numerical model used for solving the distributed state $\mathbf{u}$.

With these three types of information, GIP is formulated by three criteria:

(C-1) In the parameter space $\mathbb{P}$, the model should not be too far from it initial guess, namely, the parameter deviation satisfies the following condition

$$PE(\mathbf{S}, \boldsymbol{\theta}) \equiv \left\| (\mathbf{S}, \boldsymbol{\theta}) - (\mathbf{S}^0, \boldsymbol{\theta}^0) \right\| < \varepsilon_p, \tag{12.2.2}$$

where ε_P is the required closeness. This criterion can be replaced by requiring the model to be in the admissible region.

(C-2) In the observation space $\mathbb{F}$, the model outputs should be close to the observed system state, namely, the fitting residual satisfies

$$RE(\mathbf{S}, \boldsymbol{\theta}) \equiv \left\| \mathbf{u}_D^{obs} - \mathbf{u}_D(\mathbf{S}, \boldsymbol{\theta}) \right\| < \varepsilon_d, \tag{12.2.3}$$

where $\mathbf{u}_D(\mathbf{S}, \boldsymbol{\theta})$ are the values of the model output $\mathbf{u}(\mathbf{S}, \boldsymbol{\theta}, \mathbf{q}_D)$ at observation locations and times, and ε_d is the required upper bound of the fitting residual that should be larger than the norm of observation error.

(C-3) In the goal space $\mathbb{G}$, the model calculated goal $\mathbf{g}_E(\mathbf{S}, \boldsymbol{\theta})$ should be reliable, namely, the accuracy requirement (12.2.1) should be satisfied or the model application error is constrained by

$$AE(\mathbf{S}, \boldsymbol{\theta}) \equiv \left\| \mathbf{g}_E(\mathbf{S}^t, \boldsymbol{\theta}^t) - \mathbf{g}_E(\mathbf{S}, \boldsymbol{\theta}) \right\| < \varepsilon_g. \tag{12.2.4}$$

The solution of GIP thus becomes a multiobjective optimization (MOO) problem with three objectives, namely, minimization of *PE*, *RE,* and *AE*. From Sect. 3.1, when the weighted sum method is used, the GIP solution is given by

$$(\hat{\mathbf{S}},\hat{\boldsymbol{\theta}}) = \arg\min_{(\mathbf{S},\boldsymbol{\theta})} f(\mathbf{S},\boldsymbol{\theta}), \tag{12.2.5}$$

where

$$f(\mathbf{S},\boldsymbol{\theta}) = \mu AE(\mathbf{S},\boldsymbol{\theta}) + \nu RE(\mathbf{S},\boldsymbol{\theta}) + \lambda PE(\mathbf{S},\boldsymbol{\theta}) \tag{12.2.6}$$

and μ, ν and λ are weights. If we set $\mu = 0$ in (12.2.6), GIP reduces to EIP. In this case, the reliability of the model application is not taken into account during the inverse solution. When the ε-constraint method is used and minimization of *AE* is required, the GIP solution is given by

$$\begin{aligned}&(\hat{\mathbf{S}},\hat{\boldsymbol{\theta}}) = \arg\min_{(\mathbf{S},\boldsymbol{\theta})} AE(\mathbf{S},\boldsymbol{\theta}),\\ &\text{subject to } RE(\mathbf{S},\boldsymbol{\theta}) < \varepsilon_d \text{ and } PE(\mathbf{S},\boldsymbol{\theta}) < \varepsilon_p.\end{aligned} \tag{12.2.7}$$

If minimization of the fitting residual is required, the GIP solution is given by

$$\begin{aligned}&(\hat{\mathbf{S}},\hat{\boldsymbol{\theta}}) = \arg\min_{(\mathbf{S},\boldsymbol{\theta})} RE(\mathbf{S},\boldsymbol{\theta}),\\ &\text{subject to } AE(\mathbf{S},\boldsymbol{\theta}) < \varepsilon_g \text{ and } PE(\mathbf{S},\boldsymbol{\theta}) < \varepsilon_p.\end{aligned} \tag{12.2.8}$$

A fundamental challenge in the solution of GIP is that the term $\mathbf{g}(\mathbf{S}^t,\boldsymbol{\theta}^t)$ in the definition of GIP is unknown because $(\mathbf{S}^t,\boldsymbol{\theta}^t)$ is unknown. This makes the model application error $AE(\mathbf{S},\boldsymbol{\theta})$ incomputable and, as a result, all optimization problems in the GIP formulation become undetermined. Theories and methods are given in the next section for dealing with this difficulty.

12.3 Goal-Oriented Inversion

12.3.1 Model Structure Reduction and Model Application Error

The complexity of a model structure is measured by its dimensions (the number of degrees of freedom, DOF). In the worst case, if we do not have any prior information on the model structure, the dimensions of $\mathbf{S}^t$ may be as large as the number of grid blocks, *M*, of the numerical model. Such a complex model, of course, cannot be identified by limited data. In Chap. 7, the true model $(\mathbf{S}^t,\boldsymbol{\theta}^t)$ is replaced by a parameterized model $(\mathbf{S}_m,\boldsymbol{\theta}_m)$, where $m \ll M$, and the latter is then identified by solving EIP. A model structure error is thus introduced. The objectives of EIP and GIP are different: The former is to minimize the fitting residual and the latter is to find a reliable model for the preset goals. However, the adaptive structure identification methods described in Chap. 7 can be used to solve both EIP and GIP.

To solve GIP, first of all, we have to deal with the incomputable problem of *AE* because $(\mathbf{S}^t, \boldsymbol{\theta}^t)$ is unknown. The dependence of *AE* on the true model can be expressed explicitly by

$$AE(\mathbf{S}^t, \boldsymbol{\theta}^t; \mathbf{S}_m, \boldsymbol{\theta}_m) \equiv \left\| \mathbf{g}_E(\mathbf{S}^t, \boldsymbol{\theta}^t) - \mathbf{g}_E(\mathbf{S}_m, \boldsymbol{\theta}_m) \right\|. \tag{12.3.1}$$

If the true model structure $\mathbf{S}^t$ is known, we can use the following method to find the maximum model application error when structure $\mathbf{S}^t$ is reduced to structure $\mathbf{S}_m$. Let the admissible ranges of $\boldsymbol{\theta}^t$ and $\boldsymbol{\theta}_m$ be $\Theta(\mathbf{S}^t)$ and $\Theta(\mathbf{S}_m)$, respectively. Note that $\Theta(\mathbf{S}^t)$ and $\Theta(\mathbf{S}_m)$ have different dimensions. The model application error of replacing $(\mathbf{S}^t, \boldsymbol{\theta}^t)$ by a model with structure $\mathbf{S}_m$ can be minimized by solving the following CIP

$$AE(\mathbf{S}^t, \boldsymbol{\theta}^t; \mathbf{S}_m, \hat{\boldsymbol{\theta}}_m) = \min_{\boldsymbol{\theta}_m} AE(\mathbf{S}^t, \boldsymbol{\theta}^t; \mathbf{S}_m, \boldsymbol{\theta}_m),\ \boldsymbol{\theta}_m \in \Theta(\mathbf{S}_m), \tag{12.3.2}$$

where $(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m)$ is the closest one to the true model in all admissible models having the structure $\mathbf{S}_m$ and is called the projection of $(\mathbf{S}^t, \boldsymbol{\theta}^t)$ onto $\mathbf{S}_m$. Because $\boldsymbol{\theta}^t$ is unknown, what we can do is to find the following maximum model application error resulting from reducing the true structure $\mathbf{S}^t$ to structure $\mathbf{S}_m$, i.e.,

$$\begin{gathered} AE(\mathbf{S}^t, \mathbf{S}_m) \equiv \max_{\boldsymbol{\theta}^t} \min_{\boldsymbol{\theta}_m} AE(\mathbf{S}^t, \boldsymbol{\theta}^t; \mathbf{S}_m, \boldsymbol{\theta}_m) \\ \boldsymbol{\theta}_m \in \Theta(\mathbf{S}_m) \text{ and } \boldsymbol{\theta}^t \in \Theta(\mathbf{S}^t). \end{gathered} \tag{12.3.3}$$

In the worst case, when there is no structure information available, $\mathbf{S}^t$ is considered as $\mathbf{S}_M$, where *M* is the number of grid blocks, i.e., the maximum DOF of the numerical model. Now, we can use the computable term $AE(\mathbf{S}_M, \mathbf{S}_m)$ to replace the incomputable model application error (12.3.1) for GIP solution. But, in this case, solving the max-min problem (12.3.3) directly is computationally prohibitive. Concepts and methods are given below for dealing with this difficulty.

12.3.2 Measure the Difference Between Two Model Structures

Let us consider two models $(\mathbf{S}_A, \boldsymbol{\theta}_A)$ and $(\mathbf{S}_B, \boldsymbol{\theta}_B)$, where structures $\mathbf{S}_A$ and $\mathbf{S}_B$ may have different dimensions and patterns. When L_2-norm is used, the *structure error* resulting from replacing $\mathbf{S}_A$ with $\mathbf{S}_B$ is defined as

$$\begin{gathered} SE(\mathbf{S}_A, \mathbf{S}_B) = \max_{\boldsymbol{\theta}_A} \min_{\boldsymbol{\theta}_B} d(\mathbf{S}_A, \boldsymbol{\theta}_A; \mathbf{S}_B, \boldsymbol{\theta}_B) \\ \text{s.t. } \boldsymbol{\theta}_A \in \Theta(\mathbf{S}_A) \text{ and } \boldsymbol{\theta}_B \in \Theta(\mathbf{S}_B), \end{gathered} \tag{12.3.4}$$

where

$$d(\mathbf{S}_A,\boldsymbol{\theta}_A;\mathbf{S}_B,\boldsymbol{\theta}_B)=\Big\{\mu^2\left\|\mathbf{g}_E(\mathbf{S}_A,\boldsymbol{\theta}_A)-\mathbf{g}_E(\mathbf{S}_B,\boldsymbol{\theta}_B)\right\|_2^2 \\ +\nu^2\left\|\mathbf{u}_D(\mathbf{S}_A,\boldsymbol{\theta}_A)-\mathbf{u}_D(\mathbf{S}_B,\boldsymbol{\theta}_B)\right\|_2^2+\lambda^2\left\|(\mathbf{S}_A,\boldsymbol{\theta}_A)-(\mathbf{S}_B,\boldsymbol{\theta}_B)\right\|_2^2\Big\}^{1/2}. \tag{12.3.5}$$

From the above definition, it is easy to deduce the following:

- In general, $SE(\mathbf{S}_A,\mathbf{S}_B)\neq SE(\mathbf{S}_B,\mathbf{S}_A)$.
- When $\mathbf{S}_B$ is a simplification of $\mathbf{S}_A$, we have $SE(\mathbf{S}_B,\mathbf{S}_A)=0$.
- If both $SE(\mathbf{S}_A,\mathbf{S}_B)$ and $SE(\mathbf{S}_B,\mathbf{S}_A)$ are less than a given tolerance, the two structures are said to be equivalent. When $\mathbf{S}_B$ is a simplification of $\mathbf{S}_A$, we only need to calculate $SE(\mathbf{S}_A,\mathbf{S}_B)$ to determine their equivalence.
- If $\mathbf{S}_C$ is a simplification of $\mathbf{S}_B$, then $SE(\mathbf{S}_A,\mathbf{S}_B)\leq SE(\mathbf{S}_A,\mathbf{S}_C)$ (i.e. $SE(\mathbf{S}_A,\mathbf{S}_B)$ decreases with the increase of the complexity of $\mathbf{S}_B$).
- Because the homogeneous structure, $\mathbf{S}_1$, can be seen as a simplification of any structure, we always have $0\leq SE(\mathbf{S}_A,\mathbf{S}_B)\leq SE(\mathbf{S}_A,\mathbf{S}_1)$. In other words, the homogenization error, $SE(\mathbf{S}_A,\mathbf{S}_1)$, gives the maximum structure error when structure $\mathbf{S}_A$ is replaced by another structure.

The structure errors measured in the goal and observation spaces when $\mathbf{S}_A$ is replaced by $\mathbf{S}_B$, are denoted, respectively, by $AE(\mathbf{S}_A,\mathbf{S}_B)$ and $RE(\mathbf{S}_A,\mathbf{S}_B)$. The former can be obtained from (12.3.4) by setting $\mu=1,\nu=0$ and $\lambda=0$ in (12.3.5), while the latter can be obtained by setting $\mu=0,\nu=1$ and $\lambda=0$. The maximum model application error (12.3.3) is a special case of $AE(\mathbf{S}_A,\mathbf{S}_B)$ when $\mathbf{S}_A$ is the true model structure $\mathbf{S}^t$ (or $\mathbf{S}_M$) and $\mathbf{S}_B$ is the reduced-model structure $\mathbf{S}_m$.

12.3.2.1 The Worst-Case Parameter

The model $(\mathbf{S}_B,\boldsymbol{\theta}_{AB})$ is called a *projection* of the model $(\mathbf{S}_A,\boldsymbol{\theta}_A)$ onto structure $\mathbf{S}_B$, if $\boldsymbol{\theta}_{AB}$ is the solution of the following minimization problem

$$\boldsymbol{\theta}_{AB}=\arg\min_{\boldsymbol{\theta}_B} d(\mathbf{S}_A,\boldsymbol{\theta}_A;\mathbf{S}_B,\boldsymbol{\theta}_B),\ \text{s.t. } \boldsymbol{\theta}_B\in\Theta(\mathbf{S}_B). \tag{12.3.6}$$

Finding $\boldsymbol{\theta}_{AB}$ from (12.3.6) is equivalent to solving a CIP (i.e., searching for a parameter $\boldsymbol{\theta}_B$ with a fixed parameter structure $\mathbf{S}_B$ to minimize the distance between the two models).

Next, we define the *worst-case parameter* (WCP) of structure $\mathbf{S}_A$ as

$$\tilde{\boldsymbol{\theta}}_A=\arg\max_{\boldsymbol{\theta}_A} d(\mathbf{S}_A,\boldsymbol{\theta}_A;\mathbf{S}_B,\boldsymbol{\theta}_{AB}),\ \text{s. t. } \boldsymbol{\theta}_A\in\Theta(\mathbf{S}_A). \tag{12.3.7}$$

Substituting $\tilde{\boldsymbol{\theta}}_A$ obtained in the above equation and its projection $\tilde{\boldsymbol{\theta}}_{AB}$ obtained in (12.3.6) into the definition of the structure error (12.3.4), we have

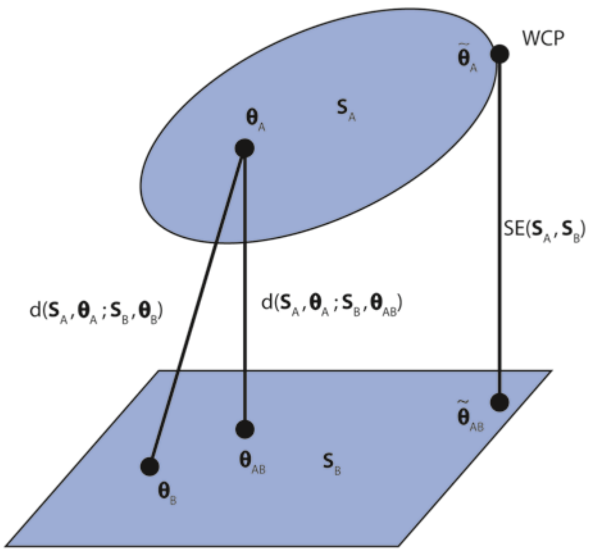

WCP
$\tilde{\boldsymbol{\theta}}_A$
$\boldsymbol{\theta}_A$
$\mathbf{S}_A$
SE($\mathbf{S}_A$, $\mathbf{S}_B$)
d($\mathbf{S}_A$,$\boldsymbol{\theta}_A$;$\mathbf{S}_B$,$\boldsymbol{\theta}_B$)
d($\mathbf{S}_A$,$\boldsymbol{\theta}_A$;$\mathbf{S}_B$,$\boldsymbol{\theta}_{AB}$)
$\tilde{\boldsymbol{\theta}}_{AB}$
$\boldsymbol{\theta}_{AB}$
$\mathbf{S}_B$
$\boldsymbol{\theta}_B$

Fig. 12.3 Projection of model $(\mathbf{S}_A, \boldsymbol{\theta}_A)$ onto structure $\mathbf{S}_B$, where $d(\mathbf{S}_A, \boldsymbol{\theta}_A; \mathbf{S}_B, \boldsymbol{\theta}_B)$ is the distance between two models. The structure error $SE(\mathbf{S}_A, \mathbf{S}_B)$ can be found by using the WCP $\tilde{\boldsymbol{\theta}}_A$ and its projection on $\mathbf{S}_B$, $\tilde{\boldsymbol{\theta}}_{AB}$,

$$SE(\mathbf{S}_A, \mathbf{S}_B) = d(\mathbf{S}_A, \tilde{\boldsymbol{\theta}}_A; \mathbf{S}_B, \tilde{\boldsymbol{\theta}}_{AB}). \tag{12.3.8}$$

Thus, once the WCP $\tilde{\boldsymbol{\theta}}_A$ of $\mathbf{S}_A$ is known, the structure error $SE(\mathbf{S}_A, \mathbf{S}_B)$ can be calculated by solving a single min problem (12.3.6), in lieu of the max-min problem (12.3.4). Figure 12.3 explains the concepts behind (12.3.6) to (12.3.8). But, how do we find the WCP in the first place? The following proposition is shown in Sun (2005) and Sun and Yeh (2007a):

Proposition: *When* $\Theta(\mathbf{S}_A)$ *is a hyperbox determined by the lower and upper bounds of its components, the WCP associated with* $\mathbf{S}_A$ *must be a vertex of the box when structure* $\mathbf{S}_A$ *is replaced by a structure* $\mathbf{S}_B$.

With this proposition, the WCP of structure $\mathbf{S}_A$ can be found by searching only all vertices of the hyperbox, instead of searching the entire box. Letting the dimensions of $\mathbf{S}_A$ be *k* and the set of all 2^k vertices of $\Theta(\mathbf{S}_A)$ be $\hat{\Theta}(\mathbf{S}_A)$, then the WCP can be found by solving the following discrete optimization problem:

$$\tilde{\boldsymbol{\theta}}_A = \arg\max_{\hat{\boldsymbol{\theta}}_A} d(\mathbf{S}_A, \hat{\boldsymbol{\theta}}_A; \mathbf{S}_B, \hat{\boldsymbol{\theta}}_{AB}), \text{ s.t. } \hat{\boldsymbol{\theta}}_A \in \hat{\Theta}(\mathbf{S}_A) \tag{12.3.9}$$

where the hat symbol is used to denote vertices, and $\tilde{\boldsymbol{\theta}}_{AB}$ is the projection of $(\mathbf{S}_A, \hat{\boldsymbol{\theta}}_A)$ onto $\mathbf{S}_B$. When *k* is small, we can directly compare the values of $d(\mathbf{S}_A, \hat{\boldsymbol{\theta}}_A; \mathbf{S}_B, \hat{\boldsymbol{\theta}}_{AB})$ at all 2^k vertices of $\Theta(\mathbf{S}_A)$ to find the WCP. For a medium size *k*, we can use genetic algorithm (GA) to search for the WCP. When *k* is large, the problem becomes hard to solve. For this case, a multiscale method can be used to find an approximation of the WCP as shown in the next subsection.

12.3.2.2 Calculation of Structure Error

To calculate the structure error $SE(\mathbf{S}_A, \mathbf{S}_B)$ of replacing $\mathbf{S}_A$ by $\mathbf{S}_B$ we need a forward solution code, a model application code, a CIP solution code, and a GA optimization (or any other global optimization) code. The main computation effort is to search for WCP by solving the maximization problem (12.3.9) with GA, the structure error is then obtained directly from (12.3.8). When the admissible region $\Theta(\mathbf{S}_A)$ is a hyperbox, GA generates a series of vertices of the region, $k_1, k_2, \cdots, k_r, \cdots$, for finding the WCP, and the following steps are completed for each vertex to be searched:

1. Let the distributed parameter at a vertex be $(\mathbf{S}_A, \hat{\boldsymbol{\theta}}_A)$ and use it as the input to run the forward solution code that simulates the designed experiment to obtain the model simulated observations $\mathbf{u}_D(\mathbf{S}_A, \hat{\boldsymbol{\theta}}_A)$.
2. Use $(\mathbf{S}_A, \hat{\boldsymbol{\theta}}_A)$ again as the input parameter to run the application code to obtain the model-predicted goal values $\mathbf{g}_E(\mathbf{S}_A, \hat{\boldsymbol{\theta}}_A)$.
3. Use the CIP solution code to find the projection of $(\mathbf{S}_A, \hat{\boldsymbol{\theta}}_A)$ onto the structure $\mathbf{S}_B$ (i.e., $(\mathbf{S}_B, \hat{\boldsymbol{\theta}}_{AB})$) defined by (12.3.6).
4. Repeat steps 1 and 2 but change the input parameter to $(\mathbf{S}_B, \hat{\boldsymbol{\theta}}_{AB})$ to run the forward code and the application code to obtain $\mathbf{u}_D(\mathbf{S}_B, \hat{\boldsymbol{\theta}}_{AB})$ and $\mathbf{g}_E(\mathbf{S}_B, \hat{\boldsymbol{\theta}}_{AB})$, respectively.
5. Calculate the value of the objective function $d(\mathbf{S}_A, \hat{\boldsymbol{\theta}}_A; \mathbf{S}_B, \hat{\boldsymbol{\theta}}_{AB})$ in (12.3.9) according to (12.3.5), then return this value to GA for searching the next vertex.

Example12.3 *Structure error of replacing a structure by another structure*
Let us return to Example 12.2 and consider how to assess the difference between two parameter structures of hydraulic conductivity $K(x, y)$. Figure 12.4 shows a three-zone structure $\mathbf{S}_3$ and a six-zone structure $\mathbf{S}_6$. In the definition of the structure error (12.3.5), we set $\mu = 1$, $\nu = 10$, and $\lambda = 0$ according to the accuracy requirement $\varepsilon_g = 1.0$, the norm of the observation error $\varepsilon_d = 0.1$, and the assumption of no prior structure information. In this case, only the parameter range is given, which is $10 \le K(x, y) \le 50$ m/day.

Because the dimensions of the two structures are low, the numbers of WCP candidates of $\mathbf{S}_3$ and $\mathbf{S}_6$ are only 8 and 64, respectively. Using the abovementioned algorithm, it is easy to do an exhaustive search to obtain the following results:

- For the case of replacing $\mathbf{S}_6$ by $\mathbf{S}_3$, the WCP of $\mathbf{S}_6$ and its projection to $\mathbf{S}_3$ are shown in Fig. 12.4a. The structure error is $SE(\mathbf{S}_6, \mathbf{S}_3) = 6.42$ with $RE(\mathbf{S}_6, \mathbf{S}_3) = 0.57$ and $AE(\mathbf{S}_6, \mathbf{S}_3) = 2.98$.
- For the case of replacing $\mathbf{S}_3$ by $\mathbf{S}_6$, the WCP of $\mathbf{S}_3$ and its projection to $\mathbf{S}_6$ are shown in Fig. 12.4b. The structure error is $SE(\mathbf{S}_3, \mathbf{S}_6) = 0.69$ with $RE(\mathbf{S}_3, \mathbf{S}_6) = 0.07$ and $AE(\mathbf{S}_3, \mathbf{S}_6) = 0.18$.

These results clearly show that the two models are not equivalent, and $\mathbf{S}_6$ cannot be replaced by $\mathbf{S}_3$ for the given goal of model application because $AE(\mathbf{S}_6, \mathbf{S}_3) > \varepsilon_g$. But $\mathbf{S}_3$ can be replaced by $\mathbf{S}_6$ because $AE(\mathbf{S}_3, \mathbf{S}_6) < \varepsilon_g$.

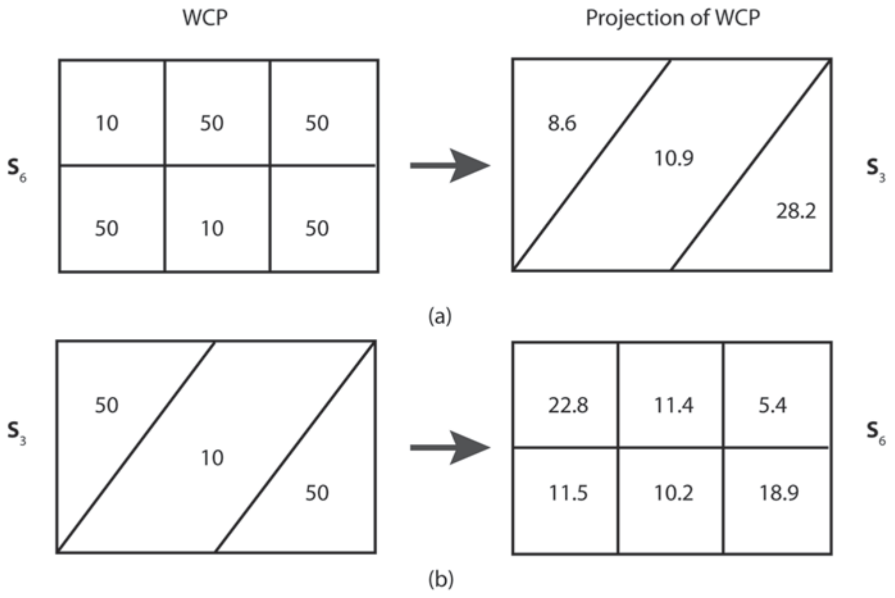


Fig. 12.4 **a** Replacing $\mathbf{S}_6$ by $\mathbf{S}_3$. *Left*: the WCP of $\mathbf{S}_6$; *right*: its projection onto $\mathbf{S}_3$; **b** Replacing $\mathbf{S}_3$ by $\mathbf{S}_6$. *Left*: the WCP of $\mathbf{S}_3$; *right*: its projection onto $\mathbf{S}_6$

The WCP of replacing one model structure by another model structure depends on many factors. Besides the dimensions and patterns of the two structures, it also depends on the methods of parameterization, the ranges of their admissible regions, the goals of model application, as well as the boundary conditions. Readers may refer to Sun and Yeh (2007a) for detailed discussions.

12.3.3 Solution Process

When the ε-constraint method (12.2.8) is used to solve GIP, the solution process is basically the same as that of solving EIP, but with an extra goal-acceptable constraint. During this process, a series of model structures is constructed:

$$\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_m, \mathbf{S}_{m+1}, \cdots. \tag{12.3.10}$$

In this series, the structure complexity is increased gradually and adaptively until a goal-acceptable model structure is found. Let us give some explanations before introducing an algorithm for constructing such a series.

How to Increase the Structure Complexity The adaptive parameterization method used in EIP for model structure identification can be used here to generate (12.3.10) but with different criteria: The former is data-oriented, while the latter is goal-oriented. For example, we can construct a binary-tree structure series by the zone refinement method (Sect. 7.2.3), in which $\mathbf{S}_{m+1}$ is generated by partitioning one (or several) selected zone of $\mathbf{S}_m$ into two zones. The problem is: Which zone should we select to partition? In EIP, a zone where the parameter value is most sensitive to the objective function of data fitting is selected. In contrast, in GIP, a zone where the parameter value is most sensitive to the given goal of the model application is selected. This can be done by comparing $\left\|\partial \mathbf{g}_E / \partial \boldsymbol{\theta}_j\right\|$ for all zones of $\mathbf{S}_m$, where $\boldsymbol{\theta}_j$ is the parameter value of zone *j*. Similarly, we can construct a multiscale structure series, in which $\mathbf{S}_{m+1}$ is generated by refining a selected block of $\mathbf{S}_m$. In this case, a block where the parameter value is most sensitive to the given goal of model application is refined. We can also use the nearest neighboring parameterization method to construct a series of Voronoi diagrams (see Sect. 7.2.2), in which $\mathbf{S}_{m+1}$ is generated from $\mathbf{S}_m$ by adding new generators to an area where the parameter value is most sensitive to the given goal of model application. In this case, the structural series is non-nested. As in EIP, we can optimize the structural pattern to make the structural series converge faster, for example, by optimizing the locations of generators.

How to Calculate the Structure Error Because $\mathbf{S}_{m+1}$ is generated from $\mathbf{S}_m$ by increasing the complexity slightly and locally, the difference between them is easy to account. For example, when $\mathbf{S}_{m+1}$ is generated by partitioning one zone of $\mathbf{S}_m$ into two zones, to find the projection of $(\mathbf{S}_{m+1}, \hat{\boldsymbol{\theta}}_{m+1})$ onto $\mathbf{S}_m$ for calculating the WCP by (12.3.9), we only need to solve a CIP with one unknown parameter component. When $\mathbf{S}_{m+1}$ and $\mathbf{S}_m$ are different in *k* zones, where *k* is a small number, to find the projection of $(\mathbf{S}_{m+1}, \hat{\boldsymbol{\theta}}_{m+1})$ onto $\mathbf{S}_m$, we only need to solve a CIP with *k* unknown parameter components (see Example 12.3). As a result, the WCP can be found easily and the structure error can be calculated effectively.

How to Find a Complex Enough Model Structure The maximum model application error $AE(\mathbf{S}_M, \mathbf{S}_m)$ in the GIP problem (12.2.8) is used as a constraint. When $AE(\mathbf{S}_M, \mathbf{S}_m) < \varepsilon_g$, we can conclude that $\mathbf{S}_m$ can be used to replace the true model structure for identifying a goal-acceptable model. Using the abovementioned process of increasing the structure complexity, the structure error decreases gradually and tends to zero. Therefore, with the increase of *m*, we can use $AE(\mathbf{S}_{m+k}, \mathbf{S}_m) < \varepsilon_g$, where $k \geq 1$, as a criterion to determine the goal-acceptance of structure $\mathbf{S}_m$.

How to Complete the Process of GIP Solution The GIP solution is obtained by solving inverse problems (12.2.8) for a series of goal-oriented model structures. There are two options: with and without optimizing the structural pattern. For the former, we have to solve EIP

$$(\hat{\mathbf{S}}_m, \hat{\boldsymbol{\theta}}_m) = \arg \min_{(\mathbf{S}_m, \boldsymbol{\theta}_m)} \left\| \mathbf{u}_D^{obs} - \mathbf{u}_D(\mathbf{S}_m, \boldsymbol{\theta}_m) \right\|, m = 1, 2, \cdots. \tag{12.3.11}$$

For the latter, the problem is reduced to CIP

$$(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m) = \arg \min_{\boldsymbol{\theta}_m} \left\| \mathbf{u}_D^{obs} - \mathbf{u}_D(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m) \right\|, m = 1, 2, \cdots. \tag{12.3.12}$$

The former needs more computational effort to optimize the pattern of a structure in (12.3.10) but the series converges faster, while the latter needs less computational effort to generate a structure but the series converges slower.

In the adaptive structure identification process described in Sect. 7.2.2, we continuously increase the model complexity until a stopping criterion is met or the existing data cannot support the identification of a more complex model. But for GIP, our purpose is to find a goal-acceptable model. If this purpose can be achieved by a simple model structure, we do not need to search for more complex model structures even the existing data can support to do so. On the other hand, if the existing data cannot support to reach this goal, new data must be collected. The data insufficiency problem appears when (i) the fitting residual becomes small, i.e., $RE_{m+1} = \left\|\mathbf{u}_D^{obs} - \mathbf{u}_D(\hat{\mathbf{S}}_{m+1}, \hat{\boldsymbol{\theta}}_{m+1})\right\| < 2\varepsilon_d$, in this case, no more information can be extracted from the existing data; or (ii) the fitting residual does not decrease with the increase of model complexity, i.e., $\left|RE_m - RE_{m+1}\right| < 2\varepsilon_d$, in this case, the existing data are insensitive to the newly added complexity.

Algorithm The Solution of GIP

Based on the ε -constraint method (12.2.8), the GIP solution can be obtained using the following stepwise regression process starting from $m = 1$:

1. Solve EIP (12.3.11) to obtain a model $(\hat{\mathbf{S}}_m, \hat{\boldsymbol{\theta}}_m)$. Let RE_m be the fitting residual.
2. Generate a more complex structure $\hat{\mathbf{S}}_{m+1}$ from $\hat{\mathbf{S}}_m$, solve EIP (12.3.11) to obtain a model $(\hat{\mathbf{S}}_{m+1}, \hat{\boldsymbol{\theta}}_{m+1})$, obtain the fitting residual RE_{m+1}, and then calculate the structural error for model application $AE_m = AE(\hat{\mathbf{S}}_{m+1}, \hat{\mathbf{S}}_m)$.
3. Consider three cases:

   i. If $AE_m \geq \varepsilon_g$ and the existing data can support the identification of a more complex model structure, replace *m* by $m+1$ and return to step 1.
   ii. If $AE_m \geq \varepsilon_g$ but the existing data cannot support the identification of a more complex model structure, go to step 4.
   iii. If $AE_m < \varepsilon_g$, stop, and use $(\hat{\mathbf{S}}_m, \hat{\boldsymbol{\theta}}_m)$ as the GIP solution $(\hat{\mathbf{S}}, \hat{\boldsymbol{\theta}})$.

4. Design an experiment for new data collection.

Without structural pattern optimization, EIP (12.3.11) reduces to CIP (12.3.12), the optimized structure $\hat{\mathbf{S}}_m$ in the above algorithm becomes a fixed structure $\mathbf{S}_m$, and $AE(\mathbf{S}_{m+1}, \mathbf{S}_m)$ is used as AE_m in step 2. In this case, more iteration steps are needed.

Comparing the adaptive model structure identification algorithm in Chap. 7 for EIP with the above algorithm for GIP, we can find that their refinement criteria and stopping criteria are different. In the former, the identified model structure is data-oriented and there is no criterion to judge the sufficiency of data, but in the latter, the identified model structure is goal-oriented and we can judge the sufficiency of the existing data. Examples of solving GIP are given in the next subsection.

12.3.4 The Maximum Homogenization Error

In the solution of GIP, it is important to calculate the maximum structure error when a complex structure is replaced by a simple one. For the true model structure $\mathbf{S}^t$, the maximum structure error is its homogenization error $SE(\mathbf{S}^t, \mathbf{S}_1)$. If we do not have prior structure information, the dimensions of $\mathbf{S}^t$ could be as high as the number of grid blocks of the simulation model. The calculation of $SE(\mathbf{S}^t, \mathbf{S}_1)$ is thus a very hard problem. In this case, we can use a multigrid process to find the WCP associated with $SE(\mathbf{S}^t, \mathbf{S}_1)$ gradually and approximately. Let us consider a series of grids with gradually increased numbers of blocks

$$M_1, M_2, \cdots, M_k, \cdots, M. \tag{12.3.13}$$

The number of blocks of a grid is the maximum DOF of the unknown parameter at that scale, and M is the number of blocks of the finest grid used in the simulation model. For simplicity, let $WCP(M_k)$ be the WCP associated with the homogenization error $SE(M_k) = SE[\mathbf{S}(M_k), \mathbf{S}_1]$. When M_k is not large (less than 50, for example), we can use GA to search the vertices of a M_k-dimensional box to find $WCP(M_k)$ and obtain $SE(M_k)$ according to (12.3.9). Assume that we have found

$$WCP(M_1), WCP(M_2), \cdots, WCP(M_k) \tag{12.3.14}$$

and

$$SE(M_1), SE(M_2), \cdots, SE(M_k). \tag{12.3.15}$$

The structure error series (12.3.15) is a monotonically increasing series and tends to $SE(M) = SE(\mathbf{S}^t, \mathbf{S}_1)$. If we find that the difference between $SE(M_k)$ and $SE(M_{k-1})$ becomes very small, we may stop and use $WCP(M_k)$ as an approximation of $WCP(M)$ for calculating the maximum structure error $SE(M)$. As shown in the following example, series (12.3.15) often converges very fast. Otherwise, if the series does not converge until M_k becomes large, the computational effort of using GA to find $WCP(M_k)$ will become unaffordable. Fortunately, WCP often has a cluster structure (see the following example). After the WCP is found for one scale, we can use GA to search only the border elements of the cluster structure for the next finer scale. We can also use any available prior information on the model structure to reduce the population of possible model structures.

Example 12.4 *Calculating the maximum homogenization error*
Let us return again to the problem presented in Example 12.2. Because there is no prior structure information, the number of DOF of $K(x, y)$ could be as high as the number of blocks ($M = 1,281$) of the numerical model. Attempting to use GA to find the WCP by searching all 2^{1281} vertices of the admissible region is infeasible. Using the multiscale method for $M_1 = 3, M_2=6$, $M_3 = 24$ and $M_4 = 96$, we obtain

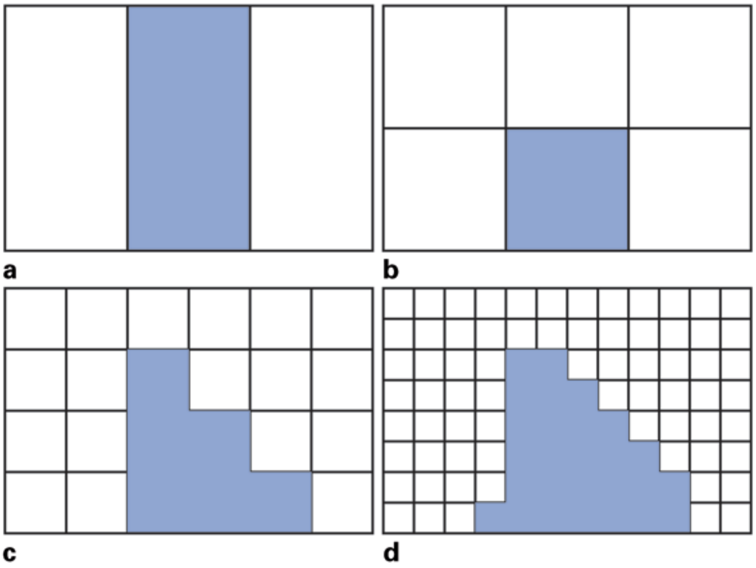


Fig. 12.5 The WCPs for different scales: **a** $WCP(3)$, **b** $WCP(6)$, **c** $WCP(24)$, and **d** $WCP(96)$

Table 12.1 Homogenization error versus structure complexit

Structure complexity	$SE(1)$	$SE(3)$	$SE(6)$	$SE(24)$	$SE(96)$
Homogeniza-tion error	0.	16.79	17.28	17.41	17.67

$WCP(3)$, $WCP(6)$, $WCP(24)$, and $WCP(96)$, as shown in Fig. 12.5a, b, c, and d, respectively, where the parameter reaches its lower bound (10 m/day) in the shaded area and upper bound (50 m/day) in the white colored area. $WCP(3)$, $WCP(6)$ and $WCP(24)$ are obtained with GA by searching all vertices of the 3D, 6D, and 24D hyperboxes, respectively, but $WCP(96)$ is obtained by searching only the border blocks of $WCP(24)$. Their corresponding homogenization errors, $SE(3)$, $SE(6)$, $SE(24)$, and $SE(96)$, are listed in Table 12.1. All of these results are obtained based on the algorithm and data used in Example 12.3 but letting $\mathbf{S}_A = \mathbf{S}_M$, $\mathbf{S}_B = \mathbf{S}_1$, and setting $\mu = 1, \nu = 0$ and $\lambda = 0$. We can find from Table 12.1 that the structure error series (12.3.15) converges very fast. $WCP(96)$, or even $WCP(24)$, can be used as an approximation of $WCP(M)$, and the maximum homogenization error of $K(x, y)$ for the given model application can be estimated as 18 m.

Example 12.5 *Solving GIP for identifying the WCP in Example 12.4*
Let us find the GIP solution by assuming $WCP(96)$ obtained in *Example 12.4* is the true parameter. The "accurate heads" are obtained by running the forward model to simulate the experimental design D described in Example 12.2 with $WCP(96)$ as the input parameter. The "observed data" used for inversion are then obtained by adding random observation error with $\varepsilon_d = 0.1$ m to the accurate head. Following the above algorithm of solving GIP, we carry out the following steps.

- The homogeneous structure $\mathbf{S}_1$ is used to fit the observed data. By solving CIP, the optimal parameter value can be obtained as shown in Fig. 12.6a. The fitting residual $RE_1 = 1.44$ m.
- Because RE_1 is larger than $2\varepsilon_d$, the data can support the identification of a two-zone structure. By solving the EIP with two zones, the optimal structure pattern $\hat{\mathbf{S}}_2$ (based on Voronoi diagram parameterization) and associated parameter values are shown in Fig. 12.6b. The fitting residual is $RE_2 = 0.22$ m. The model application error of using the single-zone model to replace the two-zone model is $AE_1 = 9.62$ m, which is the distance between the two known models in the model application space.
- Because RE_2 is still larger than $2\varepsilon_d$, the data can support the identification of a three-zone structure. By solving the EIP with three zones, the optimal structure pattern $\hat{\mathbf{S}}_3$ and associated parameter values are shown in Fig. 12.6c. The fitting residual $RE_3 = 0.07$ m and the application error $AE_2 = 0.84$ m.
- At this point, because $AE_2 < \varepsilon_g = 1.0$, we can stop our search and use the three-zone model as the GIP solution and conclude that the data are sufficient for identifying a goal-oriented model.

If we continue the GIP solution process, a four-zone model is identified, as shown in Fig. 12.6d. This calculation is not really necessary because RE_3 is already less

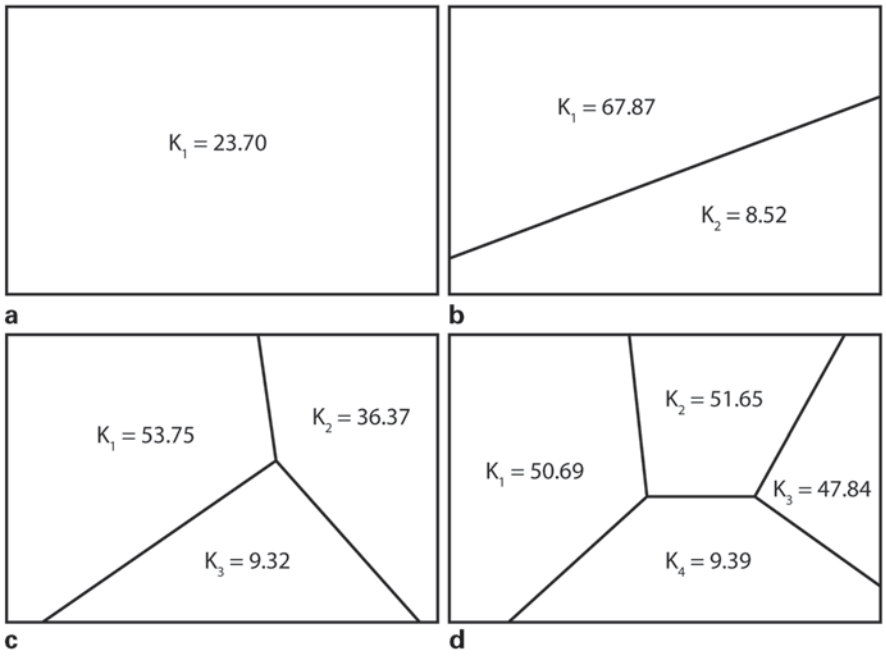


Fig. 12.6 The GIP solution: **a** the identified single-zone model, **b** two-zone model, **c** three-zone model, and **d** four-zone model

Table 12.2 The GIP solution and verification results

–	RE (m)	AE (m)	Predicted head (m)		
–	–	–	W_1	W_2	W_3
WCP(96)	0	0	87.34	57.77	86.54
$\mathbf{S}_1$	1.44	–	78.51	68.39	77.83
$\mathbf{S}_2$	0.22	9.62	88.00	57.45	86.08
$\mathbf{S}_3$	0.07	0.84	87.53	58.04	86.44
$\mathbf{S}_4$	0.06	0.19	87.41	57.94	86.55

than $2\varepsilon_d$. In other words, although the four-zone model and the three-zone model are different in the parameter space, they are equivalent for the given goal of model application.

In this example, the reliability of the GIP solution can be verified because the "true" parameter is known. Running the prediction model with $WCP(96)$ as the model input parameter and increasing the pumping rates to the assigned values, the "true steady-state heads" in the three wells are 87.34, 57.77, and 86.54 m, respectively, while the corresponding results given by the three-zone model are 87.53, 58.04, and 86.44 m. The norm of the error is 0.35 m, which is indeed less than the accuracy requirement $\varepsilon_g = 1.0$ m. All the GIP solution and verification results of this example are summarized in Table 12.2.

Example12.6 *GIP solution of a mass transport problem*
Accurately predicting the evolution and transport of a contaminant plume in a heterogeneous aquifer is not easy because detailed knowledge of the spatial distribution of hydraulic conductivity is required. For the objectives of groundwater quality management and remediation design, however, we may care only about some indicators, such as the arrival times and peak concentrations at assigned locations, instead of the actual shape of the plume. In this case, developing a goal-oriented model will be more practical and cost effective.

Let us return to Example 12.2 again and assume that W_2 is the only pumping well, which is labeled as • in Fig. 12.7a. Now, suppose that the inflow water from the boundary section *AB* is contaminated and we want to estimate the arrival time (τ_a) to the pumping well W_2. Here, the concentration of inflow water is assumed to be $C_B = 100\ \mu\text{g/l}$ and τ_a is defined as the time when concentration of the pumped water at W_2 starts to exceed $5\ \mu\text{g/l}$. We find that when the value of hydraulic conductivity varies between 10 and 50 m/day, τ_a may vary from a minimum of 176 days to a maximum of 929 days. Now we want to find a representative hydraulic conductivity $(\hat{\mathbf{S}}, \hat{\boldsymbol{\theta}})$ such that the following accuracy requirement can be satisfied

$$\left|\tau_a(\mathbf{S}^t, \boldsymbol{\theta}^t) - \tau_a(\hat{\mathbf{S}}, \hat{\boldsymbol{\theta}})\right| < \varepsilon_g = 10 \text{ [d]}. \tag{12.3.16}$$

This requirement actually is very difficult to satisfy because of the unknown variability of the hydraulic conductivity. A pumping test is designed for this goal-oriented inversion that includes one pumping well (•) and four existing observation wells (o) as shown in Fig. 12.7a. We find that the WCP for this coupled flow and mass transport problem is not identifiable even if the pumping test lasts for 3 days and the pumping rate increases to 20000 m^3/day. Information content of the pumping test can be increased by increasing the number of observation wells instead of simply increasing the pumping rate. After adding seven new observation wells (**x**) as shown in Fig. 12.7a for head observations, the WCP becomes identifiable when the pumping rate reaches 8100 m^3/day. Figure 12.7b shows the WCP associated with a 24-zone structure and obtained by a GA search. It still has a cluster structure but its shape is more complicated than that of the flow problem considered in Example 12.5. In other words, it contains more degrees of freedom.

The sufficiency of a design for WCP identification can be tested by solving a GIP as what we did in the last example. We find that a seven-zone structure shown in Fig. 12.7c can be used to replace the WCP for the given goal of model application. During this process, for each increase of model structure complexity, the following EIP must be solved to optimize the structure pattern:

$$(\hat{\mathbf{S}}_m, \hat{\boldsymbol{\theta}}_m) = \arg \min_{(\mathbf{S}_m, \boldsymbol{\theta}_m)} \left\{ \begin{array}{l} w_h \left\| \mathbf{h}_D^{obs} - \mathbf{h}_D(\mathbf{S}_m, \boldsymbol{\theta}_m) \right\| \\ + w_C \left\| \mathbf{C}_D^{obs} - \mathbf{C}_D(\mathbf{S}_m, \boldsymbol{\theta}_m) \right\| \end{array} \right\}. \tag{12.3.17}$$

We set the weighting coefficients $w_h = 1$ and $w_C = 0$ in this example because there is no concentration observation. But a mass transport model must be involved in the calculation of the model application error $\left|\tau_a(\mathbf{S}_{m+1}, \boldsymbol{\theta}_{m+1}) - \tau_a(\mathbf{S}_m, \boldsymbol{\theta}_m)\right|$. The dispersion parameters can only be roughly estimated in the mass transport model unless a tracer test is conducted.

This numerical example tells us that: (i) Constructing a goal-oriented mass transport model is feasible but more difficult than constructing a goal-oriented flow model because the structure of WCP associated with mass transport is generally more complex; (ii) if the dispersion parameters can be estimated roughly, the hydraulic conductivity identified from the head observations only can be used for mass transport modeling provided that the goals of model application are specified and incorporated into the solution of GIP.

A real case study on using GIP for conjunctive use planning of surface water and groundwater is presented in Chiu et al. (2009).

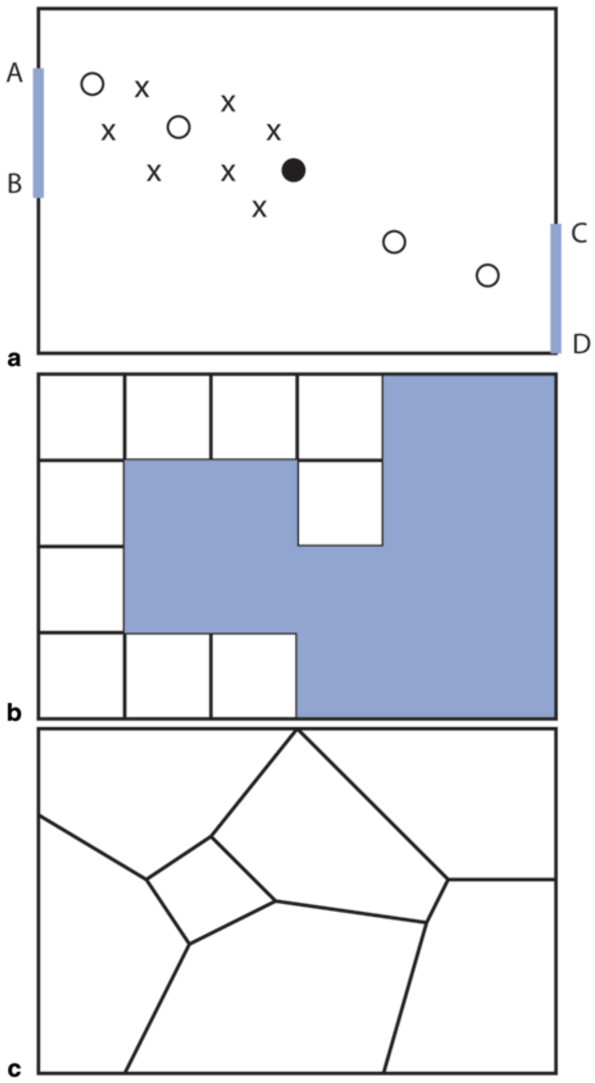


Fig. 12.7 **a** Pumping and observation wells of the pumping test design; **b** the identified WCP; and **c** the representative structure of the WCP

12.4 Goal-Oriented Experimental Design

12.4.1 Goal-Oriented Identifiability

A more complex model has more parameters, the identification of more parameters would need the support of more data, and to provide more data would require the increase of experimental cost. This relationship is called "the cost of complexity" in the field of dynamic system control (Rojas et al. 2010). The cost could be very high, or even unaffordable, for identifying a complexity system, especially when a distributed parameter system with multiscale variability is involved. The problem is how to quantify the cost and reduce it.

If the goal of model construction is for a preset model application instead of parameter identification, the data requirement or the experimental cost can be significantly decreased. As we have seen in the last chapter, the preset model application may be sensitive only to a part of the model parameters and the insensitive parameters are allowed to have large uncertainty. Toward this end, INI-based experimental designs are derived in Sects. 11.4.2 and 11.4.3 under the assumption of no model structure error. In this chapter, we have found another way to decrease the experimental cost based on model reduction, in which the original system is represented by a goal-acceptable model that has less complexity and thus needs less data to make it identifiable. The goal-oriented identifiability introduced below considers the existence of the model structure error (Sun 2005; Sun and Yeh 2007b).

An experimental design D is said to be a *sufficient design for GIP solution*, if it can provide sufficient data to support the identification of a goal-acceptable model $(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m)$ by inverse solution, namely,

$$\left\| \mathbf{g}_E(\mathbf{S}^t, \boldsymbol{\theta}^t) - \mathbf{g}_E(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m) \right\| < \varepsilon_g. \tag{12.4.1}$$

When such a design exists, the unknown parameter is said to be *goal-oriented identifiable.* We defined the "identifiability" for CIP in Chap. 2 that requires existence and uniqueness of the inverse solution and thus is difficult to be achieved. In contrast, the goal-oriented identifiability is easier to achieve because (i) it does not require the uniqueness; (ii) a goal-acceptable model structure $\mathbf{S}_m$ can always be found from the condition $AE(\mathbf{S}^t, \mathbf{S}_m) < \varepsilon_g$; and (iii) identifying a model with a simplified structure needs less data. Sun (2005) gives the following condition on goal-oriented identifiability:

Proposition: *If we can find a model structure* $\mathbf{S}_m$ *and an experimental design D such that the following condition is satisfied for any two models* $(\mathbf{S}_m, \boldsymbol{\theta}_{m,1})$ *and* $(\mathbf{S}_m, \boldsymbol{\theta}_{m,2})$:

$$\begin{aligned}
&\left\| \mathbf{u}_D(\mathbf{S}_m, \boldsymbol{\theta}_{m,1}) - \mathbf{u}_D(\mathbf{S}_m, \boldsymbol{\theta}_{m,2}) \right\| < 2[RE(\mathbf{S}^t, \mathbf{S}_m) + \varepsilon_d] \\
&\text{implies } \left\| \mathbf{g}_E(\mathbf{S}_m, \boldsymbol{\theta}_{m,1}) - \mathbf{g}_E(\mathbf{S}_m, \boldsymbol{\theta}_{m,2}) \right\| < \varepsilon_g - AE(\mathbf{S}^t, \mathbf{S}_m),
\end{aligned} \tag{12.4.2}$$

then the unknown parameter must be goal-oriented identifiable (i.e., condition (12.4.1) is satisfied). In the above equation, $RE(\cdot,\cdot)$ *and* $AE(\cdot,\cdot)$ *are defined in Sect. 12.3.2.*

In fact, using the following inequalities

$$\begin{aligned}
&\left\| \tilde{\mathbf{u}}_D(\mathbf{S}^t, \boldsymbol{\theta}^t) - \mathbf{u}_D(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m) \right\| \le \left\| \tilde{\mathbf{u}}_D(\mathbf{S}^t, \boldsymbol{\theta}^t) - \mathbf{u}_D(\mathbf{S}_m, \boldsymbol{\theta}_m^t) \right\| \\
&\le \left\| \tilde{\mathbf{u}}_D(\mathbf{S}^t, \boldsymbol{\theta}^t) - \mathbf{u}_D(\mathbf{S}^t, \boldsymbol{\theta}^t) \right\| + \left\| \mathbf{u}_D(\mathbf{S}^t, \boldsymbol{\theta}^t) - \mathbf{u}_D(\mathbf{S}_m, \boldsymbol{\theta}_m^t) \right\| \\
&\le RE(\mathbf{S}^t, \mathbf{S}_m) + \varepsilon_d,
\end{aligned} \tag{12.4.3}$$

where $\tilde{\mathbf{u}}_D$ means that observation error is added to $\mathbf{u}_D$ and $(\mathbf{S}_m, \boldsymbol{\theta}_m^t)$ is the projection of $(\mathbf{S}^t, \boldsymbol{\theta}^t)$ onto $\mathbf{S}_m$, we have

$$\begin{aligned} &\left\|\mathbf{u}_D(\mathbf{S}_m, \boldsymbol{\theta}_m^t) - \mathbf{u}_D(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m)\right\| \le \left\|\mathbf{u}_D(\mathbf{S}_m, \boldsymbol{\theta}_m^t) - \mathbf{u}_D(\mathbf{S}^t, \boldsymbol{\theta}^t)\right\| \\ &+ \left\|\mathbf{u}_D(\mathbf{S}^t, \boldsymbol{\theta}^t) - \tilde{\mathbf{u}}_D(\mathbf{S}^t, \boldsymbol{\theta}^t)\right\| + \left\|\tilde{\mathbf{u}}_D(\mathbf{S}^t, \boldsymbol{\theta}^t) - \mathbf{u}_D(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m)\right\| \\ &\le 2[RE(\mathbf{S}^t, \mathbf{S}_m) + \varepsilon_d]. \end{aligned} \tag{12.4.4}$$

Replacing $\boldsymbol{\theta}_{m,1}$ by $\boldsymbol{\theta}_m^t$ and $\boldsymbol{\theta}_{m,2}$ by $\hat{\boldsymbol{\theta}}_m$ in (12.4.4) and substituting it into (12.4.2), we have

$$\left\|\mathbf{g}_E(\mathbf{S}_m, \boldsymbol{\theta}_m^t) - \mathbf{g}_E(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m)\right\| < \varepsilon_g - AE(\mathbf{S}^t, \mathbf{S}_m). \tag{12.4.5}$$

Then, because of

$$\left\|\mathbf{g}_E(\mathbf{S}^t, \boldsymbol{\theta}^t) - \mathbf{g}_E(\mathbf{S}_m, \boldsymbol{\theta}_m^t)\right\| \le AE(\mathbf{S}^t, \mathbf{S}_m), \tag{12.4.6}$$

finally, we obtain

$$\begin{aligned} \left\|\mathbf{g}_E(\mathbf{S}^t, \boldsymbol{\theta}^t) - \mathbf{g}_E(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m)\right\| &\le \left\|\mathbf{g}_E(\mathbf{S}^t, \boldsymbol{\theta}^t) - \mathbf{g}_E(\mathbf{S}_m, \boldsymbol{\theta}_m^t)\right\| \\ &+ \left\|\mathbf{g}_E(\mathbf{S}_m, \boldsymbol{\theta}_m^t) - \mathbf{g}_E(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m)\right\| < \varepsilon_g. \end{aligned} \tag{12.4.7}$$

The goal-oriented identifiable condition (12.4.1) is thus satisfied.

Condition (12.4.2) shows that the information contained in the data must be able to counteract the effects of both observation and structure errors. If $AE(\mathbf{S}^t, \mathbf{S}_m) \ge \varepsilon_g$ (i.e., when the complexity of a model is insufficient), it is not qualified to be a goal-acceptable model no matter how its parameters are identified. When the complexity of $\mathbf{S}_m$ is increased continuously, the structure error $SE(\mathbf{S}^t, \mathbf{S}_m)$ will become smaller and smaller. When both $AE(\mathbf{S}^t, \mathbf{S}_m)$ and $RE(\mathbf{S}^t, \mathbf{S}_m)$ can be ignored, Condition (12.4.2) reduces to

$$\begin{aligned} &\left\|\mathbf{u}_D(\mathbf{S}_m, \boldsymbol{\theta}_{m,1}) - \mathbf{u}_D(\mathbf{S}_m, \boldsymbol{\theta}_{m,2})\right\| < 2\varepsilon_d \\ &\text{implies } \left\|\mathbf{g}_E(\mathbf{S}_m, \boldsymbol{\theta}_{m,1}) - \mathbf{g}_E(\mathbf{S}_m, \boldsymbol{\theta}_{m,2})\right\| < \varepsilon_g, \end{aligned} \tag{12.4.8}$$

This condition clearly shows the requirement of data sufficiency: For any two models, if they cannot be differentiated in the observation space, they must be as close as required in the goal space. We can recognize that (12.4.8) is exactly the same as Eq. (11.4.26) in Sect. 11.4.3 used to define the INI in the case without the model error. Equation (12.4.2) is a general form of goal-oriented INI when the model error exists.

12.4.2 Data Sufficiency for GIP

We learned that the algorithm of GIP solution itself provides the answer for whether or not the existing data are sufficient. First, we need to find a model structure $\mathbf{S}_m$ that is complex enough for replacing the true model structure $\mathbf{S}^t$, and second, we must have sufficient data to support the identification of a parameter vector $\hat{\boldsymbol{\theta}}_m$ such that $(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m)$ is an goal-acceptable model. But, can we answer the data sufficiency problem before the data are actually collected and the GIP is solved? This question is difficult to answer because the model structure is unknown, and a sufficient design must be robust not only to the unknown parameter values, but also to the unknown model structures.

For a given design, of course, we can use condition (12.4.2) to test its sufficiency. But this condition is too difficult to test for all admissible structures. The Monte Carlo-based method is also ineffective for this case because of the infinite variability in model complexity and structure pattern. Fortunately, there is another alternative.

When $\mathbf{S}'$ is a simplified structure of $\mathbf{S}^t$, we have $RE(\mathbf{S}', \mathbf{S}_m) \le RE(\mathbf{S}^t, \mathbf{S}_m)$ and $AE(\mathbf{S}', \mathbf{S}_m) \le AE(\mathbf{S}^t, \mathbf{S}_m)$. Therefore, if Condition (12.4.2) holds for $\mathbf{S}^t$, it must hold for $\mathbf{S}'$ because the right-hand side of its first equation increases and the right-hand side of its second equation decreases when $\mathbf{S}^t$ is replaced by $\mathbf{S}'$. We have seen in Sect. 12.2.2 that the homogenization error $AE(\mathbf{S}^t, \mathbf{S}_1)$ is the maximum structure error of model application, the WCP associated with $AE(\mathbf{S}^t, \mathbf{S}_1)$ is the most difficult one to be replaced by a simplified model, and its identification needs more information than the identification of any other parameter vectors in the admissible region. Thus, we have the following proposition (Sun 2005):

Proposition: *If a design D can make the WCP associated with $AE(\mathbf{S}^t, \mathbf{S}_1)$ to be goal-oriented identifiable, the design must be sufficient to make all parameters in the admissible region with the structure $\mathbf{S}^t$ or a simplification of $\mathbf{S}^t$ to be goal-oriented identifiable.*

In other words, a sufficient design for the WCP associated with $AE(\mathbf{S}^t, \mathbf{S}_1)$ is a robust design. But how it can be found because $\mathbf{S}^t$ is unknown? If we do not have any prior information on the true model structure, as explained in Sect. 12.3.3, we can use a multiscale process to find an approximation of the WCP and its associated homogenization error, such as $WCP(M_k)$ and $AE(M_k)$ in (12.3.14) and (12.3.15), respectively. Once the WCP associated with $AE(\mathbf{S}^t, \mathbf{S}_1)$ is found, according to the above proposition, the data sufficiency problem of a design or the identifiability problem of the WCP can be tested directly by solving a GIP.

Algorithm Test the Sufficiency of a Design for GIP Solution

1. Find the WCP associated with $AE(\mathbf{S}^t, \mathbf{S}_1)$ according to the given goals of model application.
2. Use the WCP as a model parameter vector to simulate the experimental design (with control variable $\mathbf{q}_D$) to generate "observation data" $\mathbf{u}_D^{obs} = \tilde{\mathbf{u}}_D(WCP)$.

3. Use the WCP as a model parameter vector to simulate the model application (with control variable $\mathbf{q}_E$) to generate the "goal values" $\mathbf{g}_E^t = \mathbf{g}_E(WCP)$.
4. Use the algorithm of GIP solution to test the data sufficiency. After $(\hat{\mathbf{S}}_m, \hat{\boldsymbol{\theta}}_m)$ is identified from (12.3.11), we have $AE_m = \left\| \mathbf{g}_E(WCP) - \mathbf{g}_E(\hat{\mathbf{S}}_m, \hat{\boldsymbol{\theta}}_m) \right\|$. Consider three cases: (a) If $AE_m \geq \varepsilon_g$ and the "observation data" $\tilde{\mathbf{u}}_D(WCP)$ can support the identification of a more complex model structure, move to the next structure $\mathbf{S}_{m+1}$; (b) if $AE_m \geq \varepsilon_g$ but $\tilde{\mathbf{u}}_D(WCP)$ cannot support the identification of a more complex model structure, we can conclude that the design may not be sufficient; and (c) if $AE_m < \varepsilon_g$, we can conclude that the design is sufficient for identifying the WCP and thus a robust one.

Example 12.5 shows that the design given in Example 12.2 is a robust design for the required head prediction, and Example 12.6 shows how to find a robust design for predicting the arrival time according to a given accuracy requirement.

12.4.3 Cost-Effective Experimental Design for GIP

After we have learned how to determine the sufficiency of a design for GIP solution, the remaining problem is how to make a design for new data collection when the existing data are insufficient. A cost-effective goal-oriented optimal experimental design problem is formulated in Sun (2005) and Sun and Yeh (2007b). It consists of two objectives: minimizing the total cost and minimizing the uncertainty of model application, which are also subject to the robustness and feasibility constraints,

$$\begin{aligned} &\min_D Q(D) \\ &\min_D \phi[\mathbf{J}_E(\mathbf{J}_D^T\mathbf{J}_D)^{-1}\mathbf{J}_E^T] \\ &s.t.\ \ D \in \{D_{ad}\} \end{aligned} \tag{12.4.9}$$

where $Q(D)$ is the total cost of the experiment; $\phi(\cdot)$ is a matrix norm depending on which optimality criterion is selected, such as the G-optimality or the I-optimality in Sect. 11.4.3; $\mathbf{J}_E = [\partial \mathbf{g}_E / \partial \boldsymbol{\theta}]$, $\mathbf{J}_D = [\partial \mathbf{u}_D / \partial \boldsymbol{\theta}]$, and both of which are evaluated at the WCP associated with $AE(\mathbf{S}^t, \mathbf{S}_1)$. The admissible region $\{D_{ad}\}$ consists of all feasible and robust designs (sufficient for making the WCP to be goal-oriented identifiable).

Solving the problem (12.4.9) by finding a set of Pareto optimal solutions could be computationally difficult because of the robustness constraint. The values of its two objective functions cannot be assigned arbitrarily because of the robustness constraint. When this constraint is not satisfied, the cost of the experiment must be increased and the information content must be maximized at the new cost level. The following heuristic algorithm can find a solution of (12.4.9) for nonlinear and distributed parameter systems with less computational effort.

Algorithm Find the OED for GIP Solution

1. Collect the existing data and prior information on both model structures and parameter values and find the WCP associated with $AE(\mathbf{S}^t, \mathbf{S}_1)$ according to the given goals of model application.
2. Determine an initial experimental design D_0 and calculate its cost Q_0.
3. Test if D_0 is a sufficient design for making the WCP goal-oriented identifiable. If yes, decrease the cost to $Q_1 = Q_0 - \Delta Q$, otherwise, increase the cost to $Q_1 = Q_0 + \Delta Q$.
4. Solve the following optimal experimental design (OED) problem to obtain a modified design

$$\min_D \phi[\mathbf{J}_E(\mathbf{J}_D^T \mathbf{J}_D)^{-1} \mathbf{J}_E^T], \quad \text{s. t. } Q(D) \le Q_1 \tag{12.4.10}$$

   This is a linear problem because all sensitivity coefficients in this equation are evaluated at the WCP and is independent of D.
5. Use the modified design to replace D_0 and return to step 3 with the gradually decreased value of ΔQ until a cost-effective and robust design is found.

Example 12.7 *Find the OED for the GIP problem in Example 12.2*
For this problem, the WCP associated with the maximum model application error has been represented approximately by WCP(96) in Example 12.4 and shown in Fig. 12.4d. Our purpose is to design a cost-effective pumping test for identifying a goal-accepted model of $K(x, y)$. Design variables of the test consist of the pumping rates, the pumping period, and the locations/frequencies of head observations. Based on the results of sensitivity analysis and also the purpose of decreasing the cost of the experiment, five existing wells shown in Fig. 12.2 will be used as observation wells. Thus, we only need to design the pumping rates $[q(W_1), q(W_2), q(W_3)]$ of the three pumping wells, observation frequencies, and the pumping period T. The feasibility of a design requires that the total pumping rate be less than 6000 m^3/d and the pumping period less than 4 days. Furthermore, assuming the cost of the experiment is proportional to the total amount of pumped water,

$$Q(D) \propto [q(W_1) + q(W_2) + q(W_3)]T. \tag{12.4.11}$$

The effect of the observation frequency to the cost is ignored.

The pumping test mentioned in Example 12.2 is used as the initial design that includes (1) pumping 500, 2000, and 1000 m^3/day, respectively, from wells W_1, W_2, and W_3; (2) five observation locations at W_1, W_2, W_3, O_1, and O_2; (3) the heads at these locations are measured at times $t = 0.05$, 0.1, 0.5, 1.0, and 3.0 days, respectively, and the total pumping period is 3 days. Obviously, this is a feasible design. And, as shown in Example 12.5, it is also a sufficient design. But, is it the most cost-effective design? Using the above algorithm, we can find the answer.

With increments $\Delta Q = 100$ m^3/day and $\Delta T = 0.5$ d in the algorithm, the results of solution are summarized in Table 12.3. The region of all feasible and robust designs is shown in Fig. 12.8. Any design in the region is acceptable for the GIP

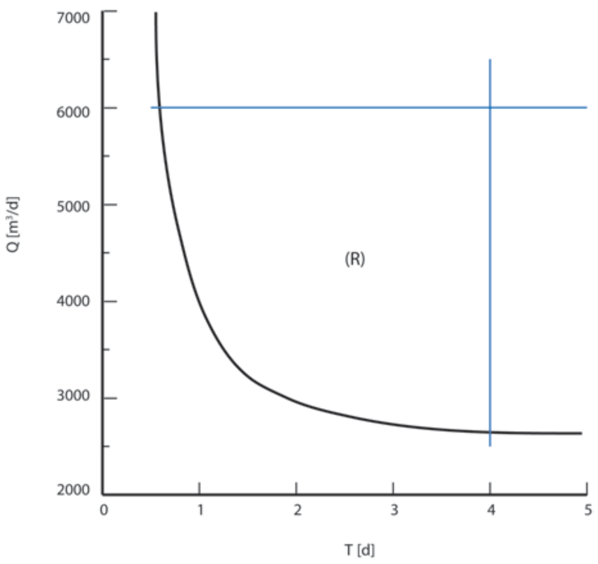


Fig. 12.8 Region (R) is the set of all feasible and sufficient designs

Table 12.3 Solutions of the goal-oriented design problem.

Pumping period (days)	Pumping rates (m^3/day)	Total volume pumped (m^3)	Sufficiency	Feasibility
0.5	7,600	3800	Yes	No
1.0	4200	4200	Yes	Yes
1.5	3300	4900	Yes	Yes
2.0	3000	6000	Yes	Yes
3.0	2800	8400	Yes	Yes
5.0	2600	13,000	Yes	No

solution, but the most cost-effective one, according to Fig. 12.8 and Table 12.3, consists of (1) one day pumping; (2) pumping 525, 2625, and 1050 m^3/day, respectively, from wells W_1, W_2, and W_3, and (3) 12 observation times at t=0.01, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.40, 0.55, 0.70, 0.85, and 1.00 days. We can also find from the table that the design with a 0.5-day pumping period is more cost-effective. But, it is infeasible because the required pumping rate exceeds the pumping capacity.

Detailed discussions of this numerical example can be found in Sun and Yeh (2007b), where the reliability of the optimal design is tested by many different structures of $K(x, y)$, including various types of continuous, discrete, and randomly generated structures.

12.5 Statistical Goal-Oriented Inverse Problem

12.5.1 Formulation of GIP in the Statistical Framework

In the statistical framework, the unknown parameter is considered as a random vector and the inverse problem is formulated with the Bayesian inference

$$p(M \mid D, I) \propto p(M \mid I)p(D \mid M, I), \tag{12.5.1}$$

where M represents the model to be estimated, D represents a set of data, and I represents prior information. In the formulation of statistical CIP (Sect. 4.1), it was assumed that the model structure is known exactly and only the value vector $\boldsymbol{\theta}$ needs to be estimated. In that case, the unknown model M is $\boldsymbol{\theta}$ and its posterior distribution can be obtained by rewriting (12.5.1) into

$$p_*(\boldsymbol{\theta}) \propto p_0(\boldsymbol{\theta})L(\boldsymbol{\theta}). \tag{12.5.2}$$

In the formulation of EIP (Sect. 7.1), it required that both the model structure $\mathbf{S}$ and the value vector $\boldsymbol{\theta}$ be estimated from prior information and data. In that case, the unknown model M is $(\mathbf{S}, \boldsymbol{\theta})$ and its joint posterior distribution can be obtained by rewriting (12.5.1) into

$$p_*(\mathbf{S}, \boldsymbol{\theta}) \propto p_0(\mathbf{S}, \boldsymbol{\theta})L(\mathbf{S}, \boldsymbol{\theta}). \tag{12.5.3}$$

In GIP, however, we have an additional dataset, the goals of model application. The Bayesian inference with two datasets is given by (Sect. 4.2.5)

$$p(M \mid D_1, D_2, I) \propto p(M \mid I)p(D_1 \mid M, I)p(D_2 \mid M, D_1, I) \tag{12.5.4}$$

Let the model M be $(\mathbf{S}, \boldsymbol{\theta})$, dataset D_1 be $\mathbf{u}_D^{obs}$, dataset D_2 be $\mathbf{g}_E(\boldsymbol{\theta}^t)$, prior information I be $(\mathbf{S}^0, \boldsymbol{\theta}^0)$, and assume that the distributions of errors in I, D_1, and D_2 are all Gaussian, the maximum a posteriori (MAP) estimate obtained from (12.5.4) is given by

$$(\hat{\mathbf{S}}, \hat{\boldsymbol{\theta}}) = \arg\min_{(\mathbf{S}, \boldsymbol{\theta})} f(\mathbf{S}, \boldsymbol{\theta}), \tag{12.5.5}$$

where

$$\begin{aligned} f(\mathbf{S}, \boldsymbol{\theta}) = {} & [\mathbf{u}_D^{obs} - \mathbf{u}_D(\mathbf{S}, \boldsymbol{\theta})]^T \mathbf{C}_D^{-1} [\mathbf{u}_D^{obs} - \mathbf{u}_D(\mathbf{S}, \boldsymbol{\theta})] \\ & + [\mathbf{g}_E(\mathbf{S}^t, \boldsymbol{\theta}^t) - \mathbf{g}_E(\mathbf{S}, \boldsymbol{\theta})]^T \mathbf{C}_E^{-1} [\mathbf{g}_E(\mathbf{S}^t, \boldsymbol{\theta}^t) - \mathbf{g}_E(\mathbf{S}, \boldsymbol{\theta})] \\ & + [(\mathbf{S}, \boldsymbol{\theta}) - (\mathbf{S}^0, \boldsymbol{\theta}^0)]^T \mathbf{C}_P^{-1} [(\mathbf{S}, \boldsymbol{\theta}) - (\mathbf{S}^0, \boldsymbol{\theta}^0)] \end{aligned} \tag{12.5.6}$$

When the norms in the observation and goal spaces are defined by $\|\bullet\|_{\mathbb{F}}^2 = \bullet^T \mathbf{C}_D^{-1} \bullet$, and $\|\bullet\|_{\mathbb{G}}^2 = \bullet^T \mathbf{C}_E^{-1} \bullet$, respectively, where $\mathbf{C}_D = \sigma_D^2 \mathbf{I}$, $\mathbf{C}_E = \sigma_E^2 \mathbf{I}$, and $\mathbf{C}_P = \sigma_P^2 \mathbf{I}$, equa-

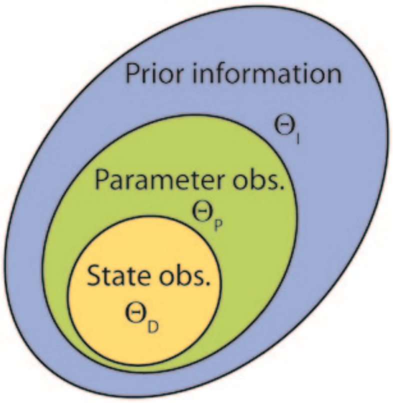


Fig. 12.9 Geostatistical inversion as a conditioning process

tions (12.5.5) and (12.5.6) are identical to Eqs. (12.2.5) and (12.2.6) derived in the deterministic framework with $\mu = 1/\sigma_D^2$, $\nu = 1/\sigma_E^2$ and $\lambda = 1/\sigma_P^2$. Therefore, in the statistical framework, these weights (as hyperparameters) can be estimated objectively.

In the statistical framework, when the ε-constraint method is used for GIP solution, the goal constraint in (12.2.8) is replaced by

$$prob\left\{AE(\mathbf{S},\boldsymbol{\theta}) = \left\|\mathbf{g}_E(\mathbf{S},\boldsymbol{\theta}) - \mathbf{g}_E(\mathbf{S}^t,\boldsymbol{\theta}^t)\right\| < \varepsilon_g\right\} > 1-\alpha, \tag{12.5.7}$$

where $(1-\alpha)\%$ is a given confidence coefficient, for example, 95 %.

12.5.2 Geostatistical Parameterization for GIP

In geostatistical inversion (Sect. 7.4.3), the estimated random field is approximated by a representative random field or parameterized by a few statistical structure parameters. Geostatistical inversion actually is a conditioning process. First, after conditioned by prior information, we obtain an admissible region Θ_I containing all models that cannot be rejected by prior information. Second, after conditioned by parameter measurements, we obtain a subset $\Theta_P \subset \Theta_I$ containing all models that cannot be rejected by these parameter measurements; the uncertainty of the estimated parameter is decreased. Third, after conditioned by state observations, we obtain a subset $\Theta_D \subset \Theta_P$ containing all models that cannot be rejected by these state observations; the uncertainty of the estimated parameter is further decreased. Figure 12.9 illustrates the conditioning process described herein and the hierarchy of subsets.

Without the parameterization error, the variance of estimation is given by cokriging estimation variance, Eq. (7.4.57),

$$Var[\hat{\theta}(\mathbf{x}) - \theta^t(\mathbf{x})] = \sigma_\theta^2 - \sum_{i=1}^{m} \lambda_i(\mathbf{x}) C_{D,\theta\theta}(\mathbf{x},\mathbf{x}_i) - \sum_{j=1}^{n} \mu_j(\mathbf{x}) C_{D,\theta u}(\mathbf{x},\mathbf{x}_j). \tag{12.5.8}$$

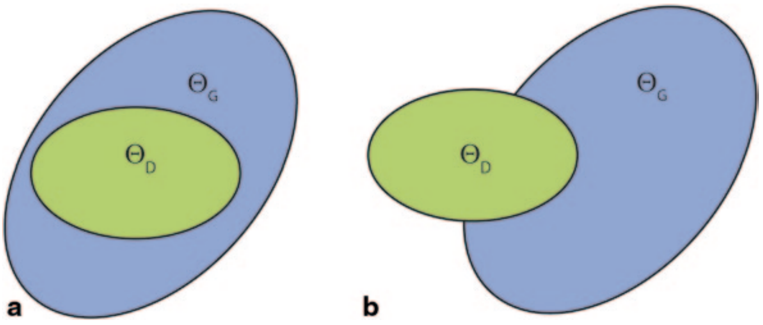


Fig. 12.10 **a** Θ_D is a subset of Θ_G; **b** Θ_D is not a subset of Θ_G

This equation gives a confidence region of the estimated $\hat{\theta}(\mathbf{x})$, which can be used as Θ_D. Now, if we can identify a set Θ_G containing all models that cannot be rejected by the preset goals of a model application and that Θ_D is a subset of Θ_G as shown in Fig. 12.10a, we can conclude that the model estimated by geostatistical inversion based on the existing data is goal-acceptable. Otherwise, as shown in Fig. 12.10b, we need more data to support the estimation of a more complex model. Unfortunately, because $\mathbf{g}_E(\mathbf{S}^t, \boldsymbol{\theta}^t)$ are the results of model application, we do not have actual measurement data for the identification of Θ_G. Furthermore, because Θ_D is a continuum containing not only plausible realizations but also other parameter distributions, a Monte Carlo-based sampling method in this case will be ineffective for testing whether or not Θ_D is a subset of Θ_G. As a result, geostatistical inversion gives only an "as is" model determined by the existing data; the effect of the model structure error and the reliability of model application are not taken into account. Moreover, when we find that the identified model is not reliable because of the large model structure error, there is no effective way to increase the model complexity and judge the data sufficiency. In this case, we can use the goal-oriented pilot point method to be introduced in the next subsection.

12.5.3 A Goal-Oriented Pilot Point Method

With the pilot point method (Sect. 7.4.4), the model complexity can be increased by adding more pilot points. Using the idea of the last section, we can develop an effective goal-oriented pilot point method, in which the number of pilot points and their locations are determined by the criteria of GIP. The unknown parameter is parameterized in the pilot point method by

$$\hat{\theta}(\mathbf{x}) = \sum_{i=1}^{m} \theta(\mathbf{x}_i)\lambda_i(\mathbf{x}, \mathbf{v}, \boldsymbol{\psi}), \quad \mathbf{x} \in (\Omega) \tag{12.5.9}$$

where $\{\lambda_i(\mathbf{x}, \mathbf{v}, \boldsymbol{\psi})\}$ are the kriging coefficients that are functions of shape parameters $\mathbf{v}$ (the coordinates of pilot points) and statistical parameters $\boldsymbol{\psi}$. Model (12.5.9) can be represented by $(\mathbf{S}_m, \boldsymbol{\theta}_m)$, where $\mathbf{S}_m = (m, \mathbf{v}, \boldsymbol{\psi})$ is the model structure and $\boldsymbol{\theta}_m = \{\theta(\mathbf{x}_i)\}$ is the parameter value vector at the locations of pilot points. Note that, unlike in the deterministic framework, here $(\mathbf{S}_m, \boldsymbol{\theta}_m)$ is not a single model when the effect of plausibility is considered (Sect. 7.4.4). The purpose of the goal-oriented pilot point method is to find an appropriate model complexity under the support of data such that $\Theta_D \subset \Theta_G$ would be satisfied (i.e., a model must be a goal-acceptable one if it cannot be rejected by data).

As in the deterministic framework, the goal-oriented pilot point method constructs a series of structures

$$\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_m, \mathbf{S}_{m+1}, \cdots. \tag{12.5.10}$$

The structure complexity in this series is increased gradually by adding more and more pilot points until a termination criterion is satisfied. We explain below how this series can be constructed.

How to Generate a Structure Series Let the number of pilot points of $\mathbf{S}_m$ be k_m. In the original pilot point method, the locations of pilot points are considered as shape parameters to be identified or simply determined based on the sensitivity analysis of observations with respect to the estimated parameter (Sect. 11.4.4). But, for GIP solution, they are determined based on the sensitivity analysis of both observations ($\mathbf{u}_D$) and goals ($\mathbf{g}_E$) with respect to the estimated parameter. After model $(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m)$ is obtained in the series, we can use the adjoint-state method to calculate $\{\partial \mathbf{u}_D / \partial \boldsymbol{\theta}\}$ and $\{\partial \mathbf{g}_E / \partial \boldsymbol{\theta}\}$ evaluated at $\hat{\boldsymbol{\theta}}_m$ by running the simulation model and the application model, respectively, and then calculate the following weighted sensitivities for all nodes $j = 1, 2, \cdots, N$:

$$s_j^2 = w \sum_{i=1}^{n} (\partial u_{D,i} / \partial \theta_j)^2 + (1-w) \sum_{k=1}^{p} (\partial g_{E,k} / \partial \theta_j)^2 \tag{12.5.11}$$

where n is the number of observations, p is the number of goals, and $0 < w < 1$ is a weight. The nodes that are associated with the first k_{m+1} largest values of $\{s_j\}$ will be used as the locations of pilot points to define the structure $\mathbf{S}_{m+1}$. In order to decrease the computational effort and also to prevent the pilot points from clustering in only a few local areas, we prefer to use a coarse grid for pilot point locating, where the block length is a half or a quarter of the correlation length.

During this process, pilot points are added gradually to those areas where the parameter values have a significant effect on the designed observations and the preset goals of model application.

Successively Conditioning the Confidence Region As mentioned in the last subsection, after conditioned by data with geostatistical inversion (kriging/cokriging), we obtain a confidence region $\Theta_{D,0}$. When the reliability requirement $\Theta_{D,0} \subset \Theta_G$ is not satisfied, pilot point models with structures in (12.5.10) are identified one by one and the confidence region of estimation is shrunken gradually. As a result, we have a series of confidence regions,

$$\Theta_{D,0} \supset \Theta_{D,1} \supset \cdots \Theta_{D,m} \supset \Theta_{D,m+1} \supset \cdots. \tag{12.5.12}$$

If the existing data can support the identification of such a model $(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m)$ that $\Theta_{D,m} \subset \Theta_G$ is satisfied, then stop and use the model as a GIP solution, otherwise, new data have to be collected.

How to Calculate the Structure Error The problem now becomes how to determine whether or not $\Theta_{D,m} \subset \Theta_G$ is true. Using a sampling approach to find the conclusion is infeasible because $\Theta_{D,m}$ is an *M*-dimensional hyperbox, where *M* is the number of grid blocks; as a continuum, $\Theta_{D,m}$ contains not only the plausible realizations, but also other realizations.

To deal with this difficulty, we can borrow the concepts and methods used in the deterministic framework to define the structure error of replacing $\mathbf{S}_{m+1}$ by $\mathbf{S}_m$

$$\mathrm{A}E(\mathbf{S}_{m+1}, \mathbf{S}_m) = \min_{\boldsymbol{\theta}_m} \left\| \mathbf{g}_E(\mathbf{S}_{m+1}, \tilde{\boldsymbol{\theta}}_{m+1}) - \mathbf{g}_E(\mathbf{S}_m, \boldsymbol{\theta}_m) \right\|, \ \boldsymbol{\theta}_m \in \Theta_{D,m} \tag{12.5.13}$$

Where $\tilde{\boldsymbol{\theta}}_{m+1} \in \Theta_{D,m+1}$ is the WCP of replacing $\mathbf{S}_{m+1}$ by $\mathbf{S}_m$. It generates the maximum model application error when $\mathbf{S}_{m+1}$ is reduced to $\mathbf{S}_m$. Therefore, when $\mathbf{S}_{m+1}$ can be regarded as the true parameter structure and $\mathrm{A}E(\mathbf{S}_{m+1}, \mathbf{S}_m) < \varepsilon_g$, we can conclude that $\Theta_{D,m} \subset \Theta_G$.

But, how do we find $\tilde{\boldsymbol{\theta}}_{m+1}$? Because the difference between $\Theta_{D,m+1}$ and $\Theta_{D,m}$ is limited to those areas where new pilot points are added, we can find $\tilde{\boldsymbol{\theta}}_{m+1}$ from $\tilde{\boldsymbol{\theta}}_m$ by searching only a part of the vertices of $\Theta_{D,m+1}$.

Algorithm The Goal-Oriented Pilot Point Method

This algorithm is basically the same as that is given in Sect. 12.3.4, except that the pilot point method is used to solve the CIP, the model complexity is increased by adding new pilot points, and a different method is used to calculate the structure error. Starting from $m = 1$, the algorithm consists of the following major steps:

1. Use the pilot point inversion method to obtain a model $(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m)$ and estimate its $(1-\alpha)$ confidence region $\Theta_{D,m}$. Let $RE_m = \left\| \mathbf{u}_D(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m) - \mathbf{u}_D^{obs} \right\|$ be the fitting residual.
2. Generate a more complex structure $\mathbf{S}_{m+1}$ from $\mathbf{S}_m$ by adding new pilot points, use the pilot point inversion method to obtain a new model $(\mathbf{S}_{m+1}, \hat{\boldsymbol{\theta}}_{m+1})$, estimate its $(1-\alpha)$ confidence region $\Theta_{D,m+1}$, calculate the fitting residual RE_{m+1}, and then calculate $AE_m = AE(\mathbf{S}_{m+1}, \mathbf{S}_m)$ by (12.5.13).
3. Consider three cases:

   i. If $AE_m \geq \varepsilon_g$ and the existing data can support the identification of a more complex model structure, replace *m* by $m+1$ and return to step 1.
   ii. If $AE_m \geq \varepsilon_g$ but the existing data cannot support the identification of a more complex model structure (either $RE_m < 2\varepsilon_d$ or $\left| RE_m - RE_{m+1} \right| < 2\varepsilon_d$), go to step 4.

iii. If $AE_m < \varepsilon_g$, stop, and use $(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m)$ as the GIP solution.

4. Design an experiment for new data collection.

Identifiability, Data Sufficiency, and Experimental Design In the statistical framework, a model is said to be goal-oriented identifiable, if the data can support the identification of a realization $(\mathbf{S}_m, \hat{\boldsymbol{\theta}}_m)$ such that all parameter vectors within its confidence region are goal-acceptable. This means that the GIP solution is a robust one: Even if the representative random field and the model structure are inexact, the model could still be useful. The above algorithm itself can answer whether or not the existing data are sufficient for so defined identifiability.

When Case (ii) in step 3 of the above algorithm is seen for a number *m*, we need to design an experimental design to collect more data to make the WCP of $\Theta_{D,m+1}$ identifiable. The designed observations should be as sensitive as possible to the parameters associated with the new pilot points.

Note that the sufficiency of a design can be tested before the experiment is actually conducted. As in the deterministic framework, after we have a WCP, we can use it as the input parameter vector to generate the "observations" by running the simulation model and predict the "goal values" by running the application model. Then, we can use these data in the goal-oriented pilot point algorithm to find out if the WCP is identifiable. If the answer is yes, the design is sufficient and we can consider how to modify it to decrease the experimental cost; otherwise, we have to modify the design to provide more information with the minimum increase of the experimental cost.

Example 12.8 *A numerical example on the goal-oriented pilot point method*

Problem Setup The groundwater flow model in Example 12.2 is used here again, but we now assume that the unknown hydraulic conductivity $K(x, y)$ is a realization of a random field. Further, assume that the random field $Y = \ln K(x, y)$ is Gaussian and characterized by three statistical parameters: mean $\mu_Y = 3.0$, standard deviation $\sigma_Y = 0.8$, and correlation distance $\ell_Y = 250\,\text{m}$. In this case, the 95 % confidence interval of $K(x, y)$ is (4.18, 96.38). Comparing with the deterministic case, its values now vary in a larger range but all of them are realizations of a given probability distribution.

The goal of the model application is to predict the steady-state heads at ten assigned locations (□ in Fig. 12.11) when the pumping rate of the three pumping wells, W_1, W_2, and W_3, reaches their planned maximum values 2000, 10000, and 4000 m^3/day, respectively. It requires that the maximum difference between the true (unknown) and the model-predicted heads at all these locations is less than 1.0 m.

The data used for inversion are provided by a designed experiment that includes: (1) pumping 500, 2000, and 1000 m^3/day, respectively, from wells W_1, W_2, and W_3; (2) heads are observed at 10 locations that are the same locations assigned by the goal of model prediction (□ in Fig. 12.11); (3) heads at these locations are measured at times $t = 0.05$, 0.5, and 2.0 days, respectively.

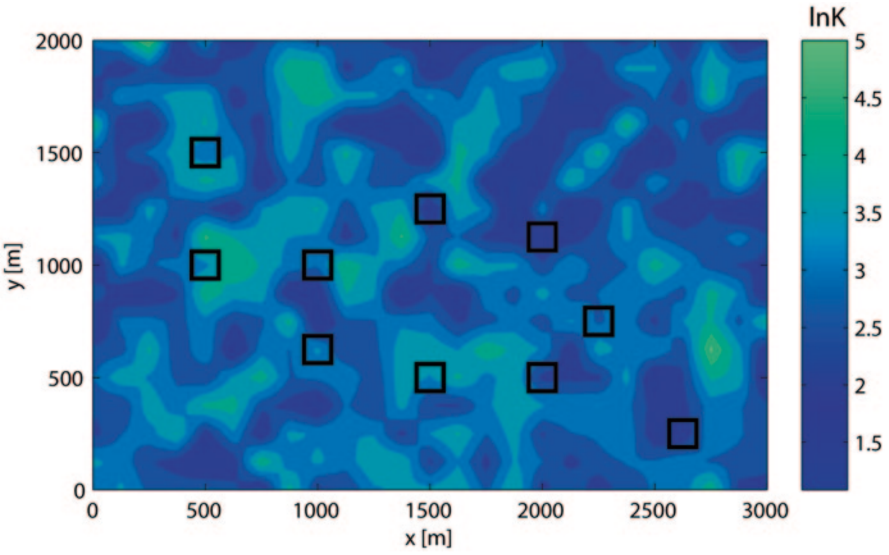


Fig. 12.11 The "true" parameter distribution to be estimated, in which the square symbols denote locations where accurate head predictions are required

Randomly sampling the *Y*-field, we obtain a realization as shown in Fig. 12.11. For testing purposes, this realization will be regarded as the true parameter. Using this as the model input to simulate the experiment, a set of model calculated observations are obtained. After adding 5 % observation error, a set of "observed data" used for inversion is obtained.

Solution The first estimation $(\mathbf{S}_0, \hat{\boldsymbol{\theta}}_0)$ shown in Fig. 12.12 is obtained by the geostatistical inversion method (see Sect. 7.4.3). Let its confidence region be $\Theta_{D,0}$. The problem is how to find out if this estimation is goal-acceptable. Using a Monte Carlo-based approach to determine the reliability of model prediction is ineffective in this case because of the high dimensionality and the variability in model structures. The goal-oriented pilot point method constructs a series of model structures (12.5.10). In this example, two pilot points are added each time when increasing the model complexity. Thus, $\mathbf{S}_1$ has only one pilot point, $\mathbf{S}_2$ has three pilot points, $\mathbf{S}_3$ has five pilot points, and so forth. According to the rank of $\{s_j\}$ defined in (12.5.11), the selected 11 pilot points for the structure $\mathbf{S}_6$ are shown as ○ in Fig. 12.13.

With the goal-oriented pilot point algorithm, the confidence regions in (12.5.12) are generated one by one for the structure series, the WCP associated with them are identified, and the structure errors between the structures are calculated. With the increase of pilot points, the confidence region of estimation is shrunken gradually, the range of WCP is decreased gradually, and the structure error becomes smaller and smaller. Figure 12.14 shows $\Theta_{D,1}$, $\Theta_{D,3}$, and $\Theta_{D,6}$. Figure 12.15 shows their WCPs. The corresponding model application errors are listed in Table 12.4. Because AE_4 is already less than 1.0 m, we can accept model $(\mathbf{S}_6, \hat{\boldsymbol{\theta}}_6)$ as a GIP solution of the problem. A further increase of the number of pilot points is unnecessary.

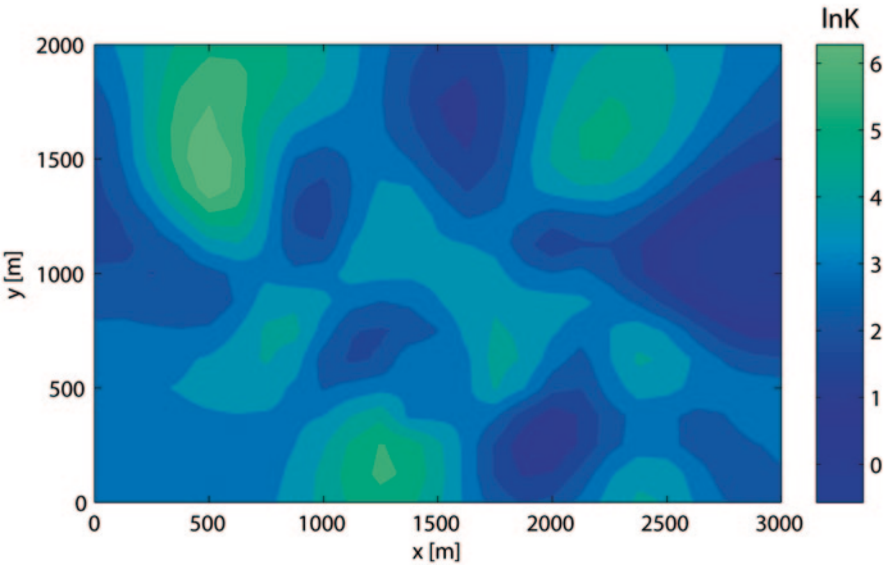


Fig. 12.12 The initial model obtained by geostatistical inversion

The identified model $(\mathbf{S}_6, \hat{\boldsymbol{\theta}}_6)$ is shown in Fig. 12.16a. It is a modification of the initial geostatistical inversion model shown in Fig. 12.12 by appropriately increasing the model complexity and extracting more information from the data. Both the structure pattern and parameter values are optimized based on the three criteria of GIP. As a result, the inverse solution is not only goal-acceptable but also data-

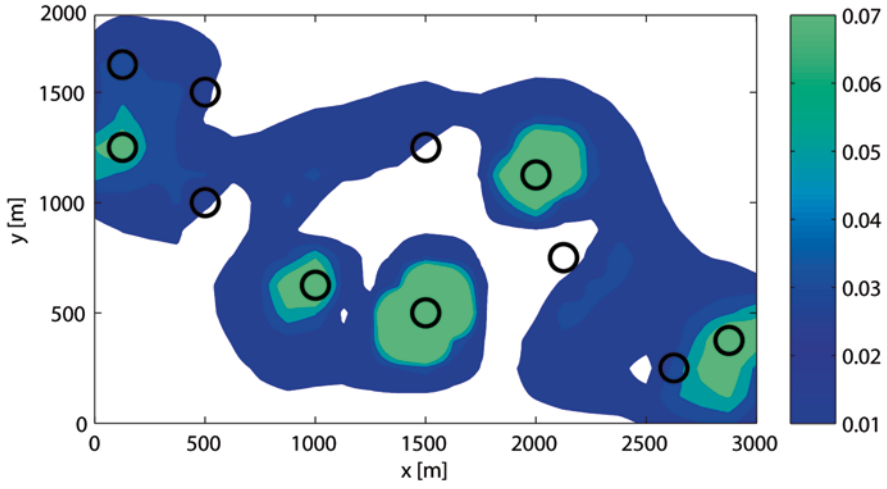


Fig. 12.13 Weighted sensitivity distribution and pilot points (circles) of $\mathbf{S}_6$

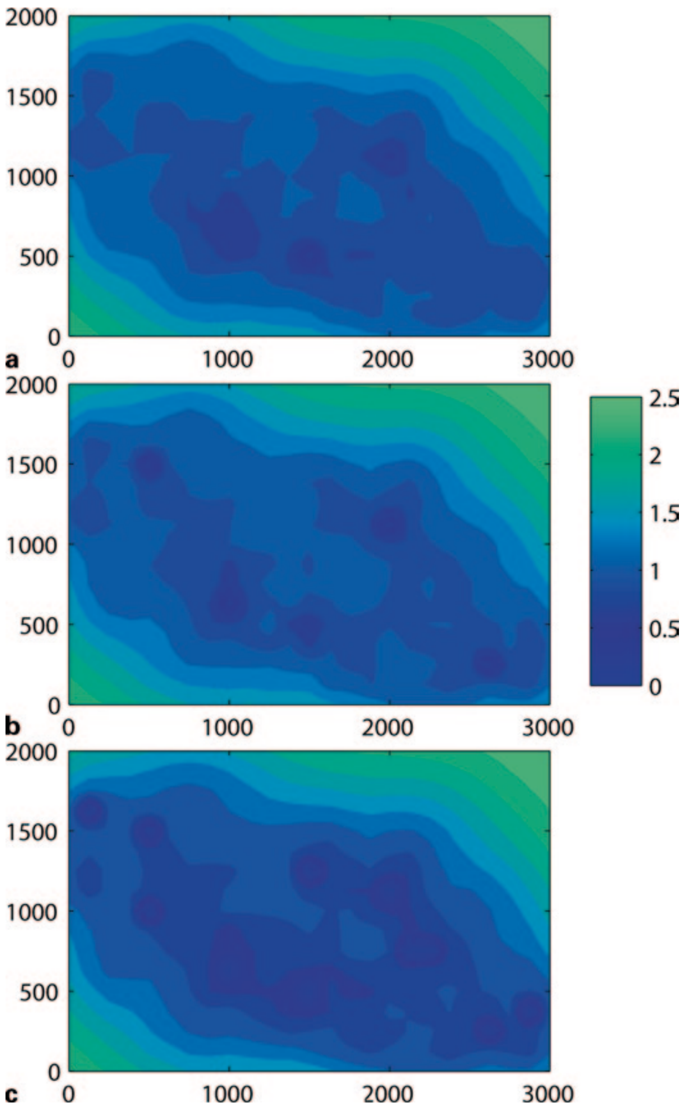


Fig. 12.14 Confidence regions associated with (a) $\mathbf{S}_1$, (b) $\mathbf{S}_3$, and (c) $\mathbf{S}_6$

acceptable, and in fact, more accurate. Figure 12.16b shows the difference between the "true" parameter distribution in Fig. 12.11 and the identified model $(\mathbf{S}_6, \hat{\boldsymbol{\theta}}_6)$ in Fig. 12.16a.

Table 12.5 shows the reliability and stability of the identified model. Without the observation error, the errors of model prediction at the ten assigned locations are negligible. After randomly adding relative errors up to 5 and 10 % to the observation data, the model prediction errors are still acceptable. Generally, GIP solution

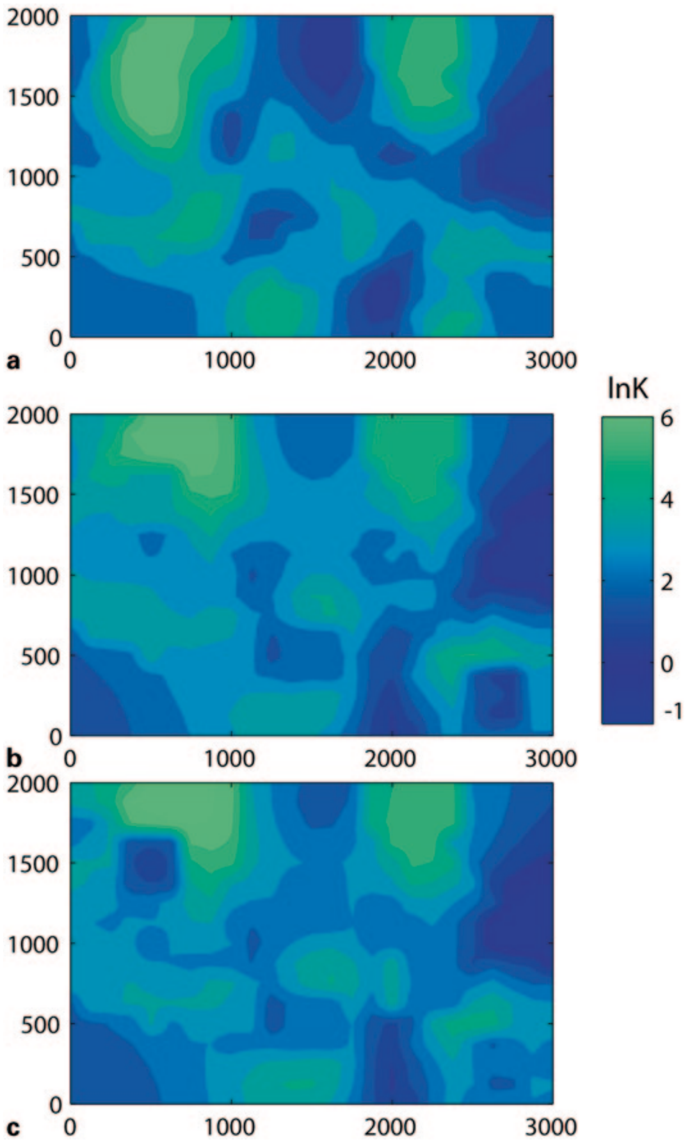


Fig. 12.15 WCPs for (a) $\mathbf{S}_1$, (b) $\mathbf{S}_3$, and (c) $\mathbf{S}_6$

Table 12.4 The reduction of the model application error with the increase of pilot points

Measure	AE_1	AE_2	AE_3	AE_4	AE_5	AE_6
L_2-norm	1.97	1.15	0.89	0.56	0.46	0.28
L_∞-norm	2.94	1.70	1.38	0.91	0.67	0.57

has better stability because it seeks the simplest structure to satisfy the accuracy requirement of a given model application. During the solution process, only one or two pilot points are added at a time, and once a goal-acceptable model complexity is found, the search process is ended. Otherwise, the algorithm prompts to collect new data. The overparameterization problem can thus be automatically avoided.

Robustness In the above goal-oriented inversion algorithm, the search process is stopped after the structure $\mathbf{S}_6$ is found. Our logic is when the WCP associated with $\Theta_{D,6}$ is within the goal-acceptable region Θ_G, we must have $\Theta_{D,6} \subset \Theta_G$; therefore, all elements of $\Theta_{D,6}$ are goal-oriented identifiable. Because the unknown parameter is an element of $\Theta_{D,6}$, it must be goal-oriented identifiable. Note that "all elements of $\Theta_{D,6}$ are goal-oriented identifiable" is a robust inference as $\Theta_{D,6}$ is a continuum that contains not only the unknown parameter and its plausible realizations but also

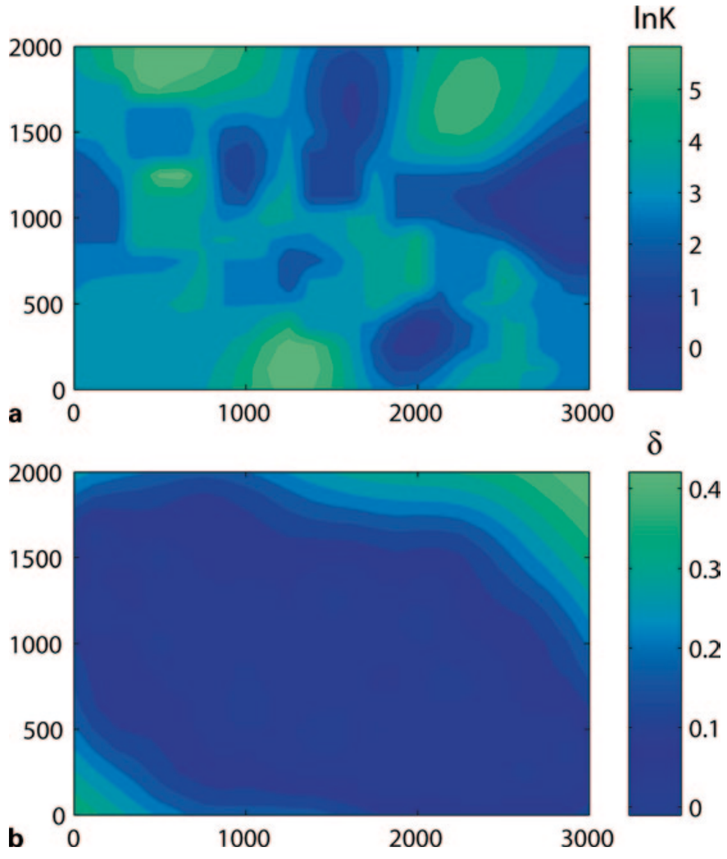


Fig. 12.16 **a** The identified model $(\mathbf{S}_6, \hat{\boldsymbol{\theta}}_6)$, and **b** difference (δ) between the "true" parameter and the identified model

Table 12.5 Errors of model prediction at the 10 assigned locations for different levels of observation error

Location	No obs. error	5 % obs. error	10 % obs. error
1	0.09	0.26	0.41
2	0.03	0.23	0.38
3	0.02	0.36	0.61
4	0.04	0.35	0.54
5	0.07	0.25	0.33
6	0.06	0.27	0.41
7	0.04	0.21	0.22
8	0.05	0.32	0.57
9	0.01	0.31	0.47
10	0.02	0.12	0.37

others. Let us consider several important cases to verify the robustness of the identified structure.

Statistically speaking, 95 % realizations of the nominal random field should be goal-oriented identifiable with structure $\mathbf{S}_6$. For testing, a hundred realizations are generated as the "true" parameter to be identified with structure $\mathbf{S}_6$. We find that the inverse solutions of them are all goal-acceptable, even after a 5 % relative error is randomly added to the observation data. The maximum head prediction error at the 10 assigned locations is 0.86 m and the average head prediction error is only 0.28 m.

We have known in Chap. 7 that accurately estimating the hyperparameters of a random field is not easy. Fortunately, we find that goal-acceptable inverse solutions can still be obtained when there are significant deviations in μ_Y, σ_Y, and ℓ_Y. For testing, we use a set of erroneous statistical parameters $\mu_Y' = 3.5$, $\sigma_Y' = 2.0$, and $\ell_Y' = 500$ m to replace their accurate values during inversion. This means that the true parameter and the inverse solution are from two different lognormal random fields. Using the same structure $\mathbf{S}_6$ for parameter estimation, a model $(\mathbf{S}_6, \hat{\boldsymbol{\theta}}_6')$ is obtained, where the optimal $\hat{\boldsymbol{\theta}}'$ is different from $\hat{\boldsymbol{\theta}}$. We find that this model is goal-acceptable too because its corresponding maximum head prediction error at the 10 assigned locations is 0.69 m and the average head prediction error is 0.39 m, both of which satisfy the given accuracy requirement. In this case, the effect of inaccurate hyperparameters is compensated by adjusting the parameter values automatically during inversion. As a result, the accuracy requirement on the hyperparameters is not very strict for GIP solution.

Moreover, when the estimated parameter is not a realization of a random field, then with a 95 % chance it can be goal-oriented identifiable, provided that it is within the confidence region $\Theta_{D,6}$. For testing, spatially homogeneous $\pm 10\%$ relative fluctuations are added to the "true" parameter. As a result, it is no longer a realization of a lognormal random field. After it is estimated with structure $\mathbf{S}_6$, we find that the inverse solution is indeed goal-acceptable. After randomly adding a 5 % relative observation error, the maximum head prediction error at the 10 assigned locations is 0.38 m and the average head prediction error is only 0.21 m. The preset accuracy requirement is satisfied. When 10 % observation error is added, however, the maximum prediction error becomes 1.51 m. The preset accuracy requirement is violated.

In summary, this example shows that the objective-oriented pilot point method is different from the traditional pilot point method in many aspects: It can determine the number of pilot points and their locations automatically during the solution process; it can determine whether or not the existing data are sufficient; and it can guarantee the reliability of model application in a robust sense, not only to plausible realizations, but also to errors in the assumed statistical distribution and its hyper-parameters. The cost of data collection for constructing a goal-oriented model is minimized but the computational demand is significantly increased, and different models must be constructed for different goals of model application.

12.6 Review Questions

1. Answer the following questions: Why is finding the reduced-order models or metamodels necessary in EWR modeling; why is finding an application-independent reduced-order model generally impossible; and why does finding an application-dependent model make sense?
2. Explain: How is the model complexity problem treated in CIP, EIP, and GIP; why is CIP a special case of EIP and EIP a special case of GIP; and why does the solution of GIP need the solution of EIP and the solution of EIP the solution of CIP?
3. Identify the three types of information (i.e., prior information, observation data, and model application goals) given in Example 12.2.
4. What is the motivation that we introduce the concept of the structure error in Sect. 12.3.1? Prove all statements on the structure error listed in below equation (12.2.5).
5. Explain the meanings of all notations in Fig. 12.3 by their definitions given in the text. Then, use this figure to explain the algorithm of calculating the structure error.
6. Give the details of calculating AE_m in step 2 of the GIP algorithm. Can we use $AE(\hat{\mathbf{S}}_{m+1}, \hat{\boldsymbol{\theta}}_{m+1}; \hat{\mathbf{S}}_m, \hat{\boldsymbol{\theta}}_m)$ to simply replace AE_m in the algorithm?
7. Why is it important to find the homogenization error of a structure? How is the homogenization error calculated?
8. Four types of identifiability are defined: the classical identifiability (CI), the quasi-identifiability (QI), the interval identifiability (INI), and the goal-oriented identifiability (GOI). Give their definitions. What are the differences between them? Why is INI a special case of GOI? If a model is not identifiable, what can we do to make it identifiable? Between INI and QI, which one needs more data? Between INI and GOI, which one needs more data?

9. Explain why the OED problem for GIP becomes easier for nonlinear models after the WCP associated with homogenization WCP is found.
10. Compare the goal-oriented statistical problem defined in this chapter and the classical geostatistical inverse problem learned in Chap. 4.
11. What are the disadvantages associated with the geostatistical inversion method learned in Chap. 7?
12. Compare the algorithm learned in Chap. 7 for the pilot point method and the algorithm for the goal-oriented pilot point algorithm. How the identifiability, data sufficiency, and experimental design are included in the latter?

Appendix

Appendix A: Functional Spaces and Mapping

In this Appendix, the basic concepts, including *vector*, *length*, *distance*, *series*, *limit*, *inner product*, and *coordinates* that we are already familiar with in the study of 2-D or 3-D spaces, are extended to infinite-dimensional function spaces where a point or a vector can be a series, a function, a data file, or other things. The concepts and methods of vector algebra and calculus can then be generalized.

A.1 Function Spaces

A.1.1 Linear Space

Let $\mathbb{X}$ be a set formed by a collection of elements, in which, the operations of addition and scalar multiplication are defined, i.e., (i) $(\mathbf{x}_1 + \mathbf{x}_2) \in \mathbb{X}$ for any two elements $\mathbf{x}_1 \in \mathbb{X}$ and $\mathbf{x}_2 \in \mathbb{X}$, and (ii) $\lambda\mathbf{x} \in \mathbb{X}$ for any real number λ and $\mathbf{x} \in \mathbb{X}$. If the following conditions are valid:

$$\begin{aligned}
&\mathbf{x}_1 + \mathbf{x}_2 = \mathbf{x}_2 + \mathbf{x}_1 \\
&(\mathbf{x}_1 + \mathbf{x}_2) + \mathbf{x}_3 = \mathbf{x}_1 + (\mathbf{x}_2 + \mathbf{x}_3) \\
&\lambda(\mathbf{x}_1 + \mathbf{x}_2) = \lambda\mathbf{x}_1 + \lambda\mathbf{x}_2 \\
&\text{There is a zero element such that any } \mathbf{x} + \mathbf{0} = \mathbf{x} \\
&\text{There is an negative element } \mathbf{x}' = -\mathbf{x} \text{ such that } \mathbf{x}' + \mathbf{x} = \mathbf{0} \\
&(\lambda + \mu)\mathbf{x} = \lambda\mathbf{x} + \mu\mathbf{x} \\
&\lambda(\mu\mathbf{x}) = (\lambda\mu)\mathbf{x} \\
&1\mathbf{x} = \mathbf{x}, 0\mathbf{x} = \mathbf{0}
\end{aligned} \tag{A.1}$$

where $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}$ are any elements of $\mathbb{X}$ and λ and μ are any numbers, then $\mathbb{X}$ is called *a linear space* or a *vector space*. Note that elements of a linear space can be numbers, vectors, functions, matrices, data files, pictures, and other things.

Example A.1 It is easy to verify that an *n*-dimensional Euclidean space $\mathbb{R}^n$ is a linear space with all *n*-dimensional vectors as its elements.

Example A.2 All functions defined on a three-dimensional region (Ω) form a linear space $\mathbb{F}_\Omega = \{f(x,y,z) \mid \text{all functions defined on } (\Omega)\}$. Each function is an element or a "point" of the space. Its subspace formed by continuous functions only, $\mathbb{C}_\Omega = \{f(x,y,z) \mid \text{all continuous functions on } (\Omega)\}$, is also a linear space.

A.1.2 Normed Space

The concept of *length* defined for an *n*-dimensional Euclidean space $\mathbb{R}^n$ can be extended to the general case. For a linear space $\mathbb{X}$, if a real number $\|\mathbf{x}\|$ is defined for each of its element $\mathbf{x}$ and the following conditions are valid:

$$\begin{aligned} &\|\mathbf{x}\| > 0, \ \text{if}\, \mathbf{x} \neq \mathbf{0} \\ &\|\lambda \mathbf{x}\| = |\lambda| \|\mathbf{x}\| \\ &\|\mathbf{x}_1 + \mathbf{x}_2\| \leq \|\mathbf{x}_1\| + \|\mathbf{x}_2\| \end{aligned} \tag{A.2}$$

Then $\mathbb{X}$ is called a *normed linear space* and $\|\cdot\|$ a *norm* to be furnished. The same space can be furnished by different norms.

Example A.3 The Euclidean length of a vector $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ is an often used norm of $\mathbb{R}^n$, which is called the L_2-norm and denoted by

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2} . \tag{A.3}$$

In fact, L_2-norm is a special case of the weighted l_p-norm $(1 \leq p \leq \infty)$ defined by

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |w_i x_i|^p \right)^{1/p} , \tag{A.4}$$

where $\{w_1, w_2, \cdots, w_n\}$ is a set of weights. In particular, we have $\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |w_i x_i|$ and $\|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |w_i x_i|$.

Example A.4 For each element $f(x,y,z)$ of the function space $\mathbb{C}_\Omega$ in Example A.2, its L_2-norm is defined by

$$\|f\|_2 = \left[\int_{(\Omega)} f^2(x,y,z) d\Omega \right]^{1/2} . \tag{A.5}$$

A.1.3 Metric Space

The concept of *distance* between any two points can be extended to the general function space. If a real number $d(\mathbf{x}_1, \mathbf{x}_2)$ is defined for any two elements $\mathbf{x}_1$ and $\mathbf{x}_2$ of a function space $\mathbb{X}$, and the following conditions are satisfied:

$$\begin{aligned} &d(\mathbf{x}_1, \mathbf{x}_2) > 0, \text{if } \mathbf{x}_1 \neq \mathbf{x}_2 \\ &d(\mathbf{x}_1, \mathbf{x}_2) = d(\mathbf{x}_2, \mathbf{x}_1) \\ &d(\mathbf{x}_1, \mathbf{x}_3) \leq d(\mathbf{x}_1, \mathbf{x}_2) + d(\mathbf{x}_2, \mathbf{x}_3) \end{aligned} \tag{A.6}$$

then $\mathbb{X}$ is called *a metric space* and $d(\cdot,\cdot)$ a *metric* to be defined. Obviously, a normed linear space must be metric space because we can define

$$d(\mathbf{x}_1, \mathbf{x}_2) = \left\| \mathbf{x}_1 - \mathbf{x}_2 \right\|. \tag{A.7}$$

After defining the *distance*, the concept *limit* is ready to be extended to the metric spaces. Let

$$\left\{ \mathbf{x}_n \right\} : \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n, \cdots \tag{A.8}$$

be a sequence in a metric space $\mathbb{X}$. If for any $\varepsilon > 0$, there exists an integer n_0, such that $d(\mathbf{x}_n, \mathbf{x}_m) < \varepsilon$ for all $n \geq n_0$ and $m \geq n_0$, then $\left\{ \mathbf{x}_n \right\}$ is called a *Cauchy sequence*. A Cauchy sequence must converge to a limit $\hat{\mathbf{x}}$ (i.e., $d(\mathbf{x}_n, \hat{\mathbf{x}}) \to 0$ when $n \to \infty$), but there is no guarantee that $\hat{\mathbf{x}}$ must be a member of $\mathbb{X}$. For example, let $\mathbb{X}$ be a space of all rational numbers, a sequence of rational numbers may converge to an irrational number. A metric space $\mathbb{X}$ with metric *d* is complete if each Cauchy sequence of it converges to a limit in it.

Definition: A complete normed linear space *is called a Banach space*.

Example A.5 It is easy to verify that an *n*-dimensional space $\mathbb{R}^n$ with Euclidean norm is a Banach space. But Banach space is usually named for infinite dimensional function spaces. A direct extension of $\mathbb{R}^n$ is the l_2 space that consists of all series $\left\{ \mathbf{x} \mid (x_1, x_2, \cdots, x_n, \cdots\cdots) \right\}$ having finite $\sum_{i=1}^{\infty} x_i^2$. It is a Banach space with the norm defined by

$$\left\| \mathbf{x} \right\|_2 = \left(\sum_{i=1}^{\infty} x_i^2 \right)^{1/2}. \tag{A.9}$$

Example A.6 The space $\mathbb{C}_\Omega$ defined in Example A.2 is incomplete because a series of continuous functions may converge to a discontinuous function.

A.1.4 Inner Product Space

Let $\mathbb{X}$ be a linear space. If a real number denoted by $(\mathbf{x}_1, \mathbf{x}_2)$ is defined for any two elements $\mathbf{x}_1$ and $\mathbf{x}_2$ in the space, and the following conditions are satisfied

$$\begin{aligned} &(\mathbf{x}, \mathbf{x}) > 0, \text{if } \mathbf{x} \neq \mathbf{0} \\ &(\lambda \mathbf{x}_1, \mathbf{x}_2) = \lambda(\mathbf{x}_1, \mathbf{x}_2) \\ &(\mathbf{x}_1 + \mathbf{x}_2, \mathbf{x}_3) = (\mathbf{x}_1, \mathbf{x}_3) + (\mathbf{x}_2, \mathbf{x}_3) \end{aligned} \tag{A.10}$$

then $(\mathbf{x}_1, \mathbf{x}_2)$ is called the *inner product* of elements $\mathbf{x}_1$ and $\mathbf{x}_2$ and $\mathbb{X}$ are inner product space. An inner product space must be a metric space because we can define $\|\mathbf{x}\| = \sqrt{(\mathbf{x}, \mathbf{x})}$.

Example A.7 In *n*-dimensional space $\mathbb{R}^n$, the inner product $\mathbf{x}$ and $\mathbf{y}$ is defined by

$$(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} x_i y_i. \tag{A.11}$$

Example A.8 The function space $L_2(\Omega)$ consists of all square-integrable functions on a region (Ω), i.e., for any element $f \in L_2(\Omega)$, we must have

$$\int_{(\Omega)} [f(\mathbf{x})]^2 d\Omega < \infty. \tag{A.12}$$

For any two elements $f(\mathbf{x})$ and $g(\mathbf{x})$ in the space, their inner production can be defined by

$$(f, g) = \int_{(\Omega)} f(\mathbf{x}) g(\mathbf{x}) d\mathbf{x}. \tag{A.13}$$

Because $|f||g| \leq \frac{1}{2}\left(f^2 + g^2\right)$, the so defined inner product does make sense. Note that $L_2(\Omega)$ contains not only continuous functions, but also some discontinuous functions. From (A.13), we have

$$d^2(f, g) = \|f - g\|_2^2 = \int_{(\Omega)} [f(\mathbf{x}) - g(\mathbf{x})]^2 d\Omega. \tag{A.14}$$

Therefore, if two functions $f(\mathbf{x})$ and $g(\mathbf{x})$ are equal almost everywhere, then $\|f - g\| = 0$, and we should consider them as the same element in the space $L_2(\Omega)$.

Definition: A complete inner product space is called a *Hilbert space*.
Obviously, $\mathbb{R}^n$ is a Hilbert space. The completeness of $L_2(\Omega)$ is given by the Riesz–Fischer theorem (Meise, Vogt 1997). Therefore, $L_2(\Omega)$ is a Hilbert space.

A.1.5 Basis of a Hilbert Space

Two elements (vectors) $\mathbf{e}_1$ and $\mathbf{e}_2$ of a Hilbert space are said to be *orthogonal* if their inner product is zero. A set of vectors $\{\mathbf{e}_i \mid i = 1, 2, \cdots\}$ is called an *orthogonal system* if any two members of the set are orthogonal and the norms of all of them are equal to one, viz.

$$(\mathbf{e}_i, \mathbf{e}_j) = \begin{cases} 0, \text{when } i \neq j \\ 1, \text{when } i = j \end{cases}. \tag{A.15}$$

An orthogonal system is called the *complete* if there is no another vector in the space that is orthogonal to all of vectors of the system. A complete orthogonal system forms a *basis* of the space.

Example A.9 The basis of $\mathbb{R}^n$ consists of *n* vectors: $(1,0,0,\cdots,0)$, $(0,1,0,\cdots,0)$, $\cdots,(0,0,\cdots,0,1)$. An infinite-dimensional space, such as the l_2 space in Example A.5 has infinite basis elements: $(1,0,0,0,\cdots)$, $(0,1,0,0,\cdots),(0,0,1,0,\cdots),\cdots$.

Example A.10 The basis of a Hilbert function space is a set of functions $\{\phi_1(x), \phi_2(x), \cdots, \phi_n(x), \cdots\}$ and not unique. For the $L_2(\Omega)$ space in Example A.8, when (Ω) is in interval $[0,1]$, a possible selection of its basis is:

$$\phi_1(x) = 1,\ \phi_2(x) = \sqrt{2}\sin(2\pi x),\ \phi_3(x) = \sqrt{2}\cos(2\pi x), \cdots,$$
$$\phi_{2n}(x) = \sqrt{2}\sin(2\pi n x),\ \phi_{2n+1}(x) = \sqrt{2}\cos(2\pi n x), \cdots$$

In this book, we will see other types of basis functions used for the $L_2(\Omega)$ space.

A.1.6 Expansion and Approximation in Hilbert Spaces

Let $\{\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \cdots, \phi_n(\mathbf{x}), \cdots\}$, where $\mathbf{x} \in (\Omega)$, be a set of complete basis functions of a Hilbert function space. Each element $f(\mathbf{x})$ in the space can be represented by an expression of these basis functions as

$$f(\mathbf{x}) = \sum_{i=1}^{\infty} c_i \phi_i(\mathbf{x}). \tag{A.16}$$

Calculating the inner production of $f(\mathbf{x})$ and $\phi_n(\mathbf{x})$ gives

$$(f, \phi_n) = \sum_{i=1}^{\infty} c_i (\phi_i, \phi_n) = c_n,\ n = 1, 2, \cdots \tag{A.17}$$

Therefore, the inner production (f,ϕ_n) can be seen as the *projection* of an element (vector) f onto a basis function ϕ_n (an axis direction), and c_n the corresponding coordinate.

From the point of view of function approximation, if we want to use a linear combination of N basis functions,

$$\tilde{f}(\mathbf{x}) = \sum_{i=1}^{N} c_i \phi_i(\mathbf{x}), \tag{A.18}$$

as an approximation of a given function $f(\mathbf{x})$, the norm of approximation error is

$$\begin{aligned} \|f-\tilde{f}\| &= (f-\tilde{f}, f-\tilde{f}) \\ &= (f,f)+(\tilde{f},\tilde{f})-(f,\tilde{f})-(\tilde{f},f) \\ &= \|f\|^2 + \sum_{i=1}^{N} c_i^2 - 2\sum_{i=1}^{N} c_i (f,\phi_i) \\ &= \|f\|^2 + \sum_{i=1}^{N} [c_i - (f,\phi_i)]^2 - \sum_{i=1}^{N} (f,\phi_i)^2. \end{aligned} \tag{A.19}$$

This equation shows that the norm of approximation error will be minimized if we choose $c_i = (f,\phi_i)$ for all $i = 1,2,\cdots,N$ in (A.18). Therefore, after a set of basis functions of a function space is selected, the best approximation of a given function in the space is a truncation of its expansion (A.16), the truncation error tends to zero when $N \to \infty$, and the rate of convergence depends on the basis functions to be selected.

A.2 Mappings and Operators

A.2.1 Mapping

Let $\mathbb{X}$ and $\mathbb{Y}$ be two arbitrary spaces. A *rule* $\mathcal{F}$ that associates each element (preimage) $\mathbf{x} \in \mathbb{X}_{ad} \subset \mathbb{X}$ with a unique element (image) $\mathbf{y} \in \mathbb{Y}$ is termed a *mapping* or a *transformation* from $\mathbb{X}$ to $\mathbb{Y}$ and denoted by $\mathbf{y} = \mathcal{F}(\mathbf{x})$, $\mathbf{x} \in \mathbb{X}$, $\mathbf{y} \in \mathbb{Y}$, or by $\mathcal{F} : \mathbb{X} \to \mathbb{Y}$. The subset $\mathbb{X}_{ad}$ is its *definition domain*, and the subset formed by all images, $\mathbb{Y}_{\mathcal{F}} \subset \mathbb{Y}$, is named its range. Obviously, mapping is a generalized concept of function. The preimages and images of a mapping can be elements of function spaces, data files, figures, and others.

Example A.11 A scalar form of transformation $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^m$ is given by

$$\begin{cases} y_1 = f_1(x_1, x_2, \cdots, x_n) \\ y_2 = f_2(x_1, x_2, \cdots, x_n) \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ y_m = f_m(x_1, x_2, \cdots, x_n) \end{cases} \tag{A.20}$$

Example A.12 A mathematical model with a distributed parameter $\theta \in \mathbb{H}_1$ and a distributed state variable $u \in \mathbb{H}_2$, where $\mathbb{H}_1$ and $\mathbb{H}_2$ are two Hilbert spaces, can be represented by a mapping $u = \mathcal{M}(\theta)$, or $\mathcal{M}: \mathbb{H}_1 \to \mathbb{H}_2$.

Many concepts and methods established for functions can be extended to the general mappings, for example, the concept of continuity. A mapping $\mathbf{y} = \mathcal{F}(\mathbf{x})$ from a metric space $\mathbb{X}$ to a metric space $\mathbb{Y}$ is continuous at an element $\mathbf{x}_0$ if for every $\varepsilon > 0$ there is a $\delta > 0$ such that $\left\|\mathbf{x} - \mathbf{x}_0\right\| < \delta$ implies $\left\|\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{x}_0)\right\| < \varepsilon$.

Inverse Mapping In the above definition of mapping, it is required that each preimage can have only one image. The inverse statement, however, may not be true. An image may have multiple preimages. If the correspondence between the elements of $\mathbb{X}_{ad}$ and $\mathbb{Y}_{\mathcal{F}}$ is one-to-one, then $\mathcal{F}$ is said to be *injective*. In this case, $\mathcal{F}$ has a unique inverse mapping to be denoted by $\mathbf{x} = \mathcal{F}^{-1}(\mathbf{y})$.

A.2.2 Operator

An operator is a mapping from one vector space (or module) to another vector space. It is often used to represent an operation on a function that generates another function. For example, the differential operator (D, D_x, ∂_x) accepts a differentiable function and returns a derivative function.

Example A.13 If we define an operator as

$$\mathcal{L} \equiv \frac{\partial}{\partial t} - a^2 \frac{\partial^2}{\partial x^2} + b^2 \frac{\partial}{\partial x}, \tag{A.21}$$

the 1-D advection-diffusion PDE

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2} - b^2 \frac{\partial u}{\partial x} \tag{A.22}$$

can be represented shortly by an operator equation $\mathcal{L}(u) = 0$ or $\mathcal{L}u = 0$.

A.2.3 Linear Operator

The most common kind of operator encountered is *linear operator*. Let $\mathbb{U}$ and $\mathbb{V}$ be vector spaces. An operator $\mathcal{L}:\mathbb{U}\rightarrow\mathbb{V}$ is called linear if

$$\mathcal{L}(\alpha f+\beta g)=\alpha\mathcal{L}f+\beta\mathcal{L}g, \tag{A.23}$$

where f and $g\in\mathbb{U}$, $\mathcal{L}f$ and $\mathcal{L}g\in\mathbb{V}$, and α and β are any numbers. According to this definition, any linear PDE, such as the convection-diffusion equation in (A.21), is represented by a linear operator.

A linear operator $\mathcal{L}:\mathbb{H}_1\rightarrow\mathbb{H}_2$, where $\mathbb{H}_1$ and $\mathbb{H}_2$ are two Hilbert spaces, is called a *bounded operator*, if there is a real number *k* such that

$$\left\|\mathcal{L}f\right\|\le k\left\|f\right\|,\ \text{for all } f\in\mathbb{H}_1. \tag{A.24}$$

The *norm* of a linear operator $\mathcal{L}:\mathbb{H}_1\rightarrow\mathbb{H}_2$ denoted by $\left\|\mathcal{L}\right\|$ is the smallest number *k* satisfying the above equation, i.e.,

$$\left\|L\right\|\equiv\sup_{\left\|f\right\|\neq 0}\frac{\left\|Lf\right\|}{\left\|f\right\|}. \tag{A.25}$$

It can be shown that a linear operator $\mathcal{L}:\mathbb{H}_1\rightarrow\mathbb{H}_2$ is continuous if and only if it is bounded (Griffel 2002).

For a Hilbert space $\mathbb{H}$, the inner product (f,g) is a linear mapping from $\mathbb{H}\rightarrow\mathbb{R}$. Because $\left|(f,g)\right|\le\left\|f\right\|\left\|g\right\|$, from (A.25), we have $\left\|(f,g)\right\|=\left\|g\right\|$. The Riesz representation theorem states: for any linear mapping $\mathcal{I}(f)$ defined on $\mathbb{H}$, there exists a unique element $g\in\mathbb{H}$, such that $\mathcal{I}(f)=(f,g)$ for all $f\in\mathbb{H}$. Proof of this theorem can be found, for example, in Debnath and Mikusinski (1999).

A.2.4 Adjoint Operator

Let $\mathcal{L}:\mathbb{H}_1\rightarrow\mathbb{H}_2$ be a bounded linear operator. The inner product $(\mathcal{L}f,g)$ on $\mathbb{H}_2$ is a linear mapping $\mathcal{I}(f)$, where $g\in\mathbb{H}_2$ and for any $f\in\mathbb{H}_1$. According to the Riesz representation theorem, there exists a unique $g^*\in\mathbb{H}_1$, such that $\mathcal{I}(f)=(f,g^*)$, i.e.,

$$(\mathcal{L}f,g)=(f,g^*). \tag{A.26}$$

Now, we can introduce a new operator $\mathcal{L}^*:\mathbb{H}_2\rightarrow\mathbb{H}_1$, which is called the adjoint operator of $\mathcal{L}$ and defined by

$$\mathcal{L}^*g\equiv g^*. \tag{A.27}$$

Substituting (A.27) into (A.26), we have

$$(\mathcal{L}f,g)=(f,\mathcal{L}^*g). \tag{A.28}$$

For any bounded linear operator $\mathcal{L}$ on a Hilbert space, its adjoint operator $\mathcal{L}^*$ is existent and unique. When $\mathcal{L} = \mathcal{L}^*$, $\mathcal{L}$ is called *self-adjoint*.

For more detailed discussions on function spaces and operators, we refer the reader to the text books of Griffel (2002), Conway (1997), Meise and Vogt (1997), Hirsch and Lacombe (1999), Debnath and Mikusinski (1999), and Stein, Shakarchi (2011).

Appendix B: Probability, Statistics, and Random Fields

This appendix reviews some very basic concepts and results from the probability theory and statistics covered by college-level text books (Dekking 2005; Mendenhall et al. 2009; Walpole 2007). With this preliminary knowledge, readers are expected to have no essential difficulty in understanding the advanced statistical tools introduced in this book.

B.1 Random Variables and Probability Distributions

B.1.1 Probability

A *sampling space* $\mathbb{S}$ is a set that includes all possible outcomes for a random selection from a specified population. An *event* is a subset of a sample space. A *probability* is a numerically valued mapping $P: \mathbb{S} \to [0,1]$ that assigns a number $P(A)$ to every event *A* subject to the following conditions:

1. $P(A) \geq 0$, for any $A \subset \mathbb{S}$
2. $P(A) \geq 0$
3. If $A_1, A_2, \cdots$ is a sequence of mutually exclusive events, then

$$P(\cup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i) \tag{B.1}$$

From the above definition, we can show that:

- If *A* and *B* are mutually exclusive events, then

$$P(A \cup B) = P(A) + P(B)$$

- If $A \subseteq B$, then

$$P(A) \leq P(B)$$

- $P(\varnothing) = 0$

B.1.2 Random Variables

A continuous random variable is a variable that may take any value in a finite or an infinite interval due to uncontrollable factors. For example, when we measure the water level of a well many times, the outcomes may be any value within a range even the groundwater flow is in the steady state. A continuous random variable X is characterized completely by its *probability density function* (PDF), $p_X(x)$, satisfying the following conditions:

- $p_X(x) > 0$ for all x
- $\int_{-\infty}^{\infty} p_X(x)dx = 1$
- $P(a \le X \le b) = \int_a^b p_X(x)dx$

The *cumulative distribution function* (CDF) of X, $P_X(x)$, is defined as the probability of $X \le x$, i.e.,

$$P_X(x) = P(X \le x) = \int_{-\infty}^{x} p_X(t)dt, \; -\infty < x < \infty. \tag{B.2}$$

In the following, the subscript X of p_X and P_X will be omitted so that no confusion should occur. With PDF, the probability of the values of X falls in an interval $[a,b]$ is given by

$$P(a < x \le b) = \int_a^b p(x)dx = P(b) - P(a). \tag{B.3}$$

Example B.1 The life length of a bacterium is a random variable. Its PDF is given by an exponential function:

$$p(x) = \lambda e^{-\lambda x}, \text{ when } x > 0; \; p(x) = 0, \text{ elesewhere,} \tag{B.4}$$

where $\lambda > 0$ is a constant parameter. When $\lambda = 1/2$, the probability of the life length of a bacterium being $2 \sim 4$ h is

$$P(2 < x \le 4) = \frac{1}{2}\int_2^4 e^{-x/2}dx = 0.233.$$

According to (B.2), the CDF of the random variable is

$$P(x) = 1 - e^{-\lambda x}, \text{ when } x \ge 0; P(x) = 0, \text{ when } x < 0. \tag{B.5}$$

B.1.3 The Expected Value and Variance of a Random Variable

The *expected value* of a random variable *X* is defined by

$$E(X) = \int_{-\infty}^{\infty} xp(x)dx, \tag{B.6}$$

where $f(x)$ is the PDF of *X*. The expected value is used to measure the mean of a random variable and often denoted by μ_X, or simply, by μ.

The *variance* of random variable *X* with PDF $p(x)$ is defined by

$$Var(X) = E[(X-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 p(x)dx \tag{B.7}$$

The variance is used to measure the spread of a random variable and often denoted by σ_X^2, or simply σ^2, where σ is called the standard deviation. From the definitions (B.6) and (B.7), it is easy to show that $V(X) = E(X^2) - \mu^2$.

Example B.2 The mean life length of a bacterium considered in Example B.1 is

$$E(X) = \lambda\int_{0}^{\infty} xe^{-\lambda x}dx = \frac{1}{\lambda}.$$

The variance of the random variable is

$$V(X) = \lambda\int_{0}^{\infty} (x-\frac{2}{\lambda})^2 e^{-\lambda x}dx = \frac{1}{\lambda^2}$$

Random variable $Y = aX + b$ is a linear transformation of random variable *X*, where *a* and *b* are constant. It is easy to show that

- $E(aX+b) = aE(X) + b$
- $V(aX+b) = a^2V(X)$

B.1.4 Some Important Continuous Probability Distributions

The Normal Distribution The most widely used continuous probability distribution is the normal distribution. Its PDF is given below

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \quad -\infty < x < \infty \tag{B.8}$$

where μ and σ are two parameters. Using (B.6) and (B.7), it is easy to show that

$$E[\mathcal{N}(\mu,\sigma)] = \mu \qquad \text{(B.9)}$$
$$V[\mathcal{N}(\mu,\sigma)] = \sigma^2$$

A normally distributed random variable is often indicated simply by $\mathcal{N}(\mu,\sigma)$.

The Lognormal Distribution A random variable *X* is said to have a lognormal distribution if $Y = \ln X$ is normally distributed. The PDF of lognormal distribution is given by

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma x}\exp\left[-\frac{(\ln x-\mu)^2}{2\sigma^2}\right], \text{ when } x > 0;\ p(x) = 0, \text{when } x \le 0 \quad \text{(B.10)}$$

where $\propto$ and σ are two parameters. The probability $P(X \le x)$ can be calculated by

$$P(X \le x) = P(\ln X \le \ln x) = P(Y \le \ln x), \qquad \text{(B.11)}$$

where Y is normal distribution $\mathcal{N}(\mu,\sigma)$. The mean and variance of a lognormal distribution (B.10) are

$$E(X) = e^{\mu+\frac{\sigma^2}{2}} \qquad \text{(B.12)}$$
$$V(X) = e^{2\mu+\frac{\sigma^2}{2}}\left(e^{\sigma^2}-1\right)$$

The Gamma Distribution The PDF of the Gamma distribution with is given by

$$p(x) = \frac{1}{\Gamma(\alpha)\beta^\alpha}x^{\alpha-1}x^{-x/\beta}, \text{ when } x > 0;\ p(x) = 0, \text{when } x \le 0 \qquad \text{(B.13)}$$

where α and β are two parameters, and $\Gamma(\alpha) = \int_0^\infty x^{\alpha-1}e^{-x}dx$ is the Gamma function. The mean and variance of Gamma distribution are

$$E(x) = \alpha\beta \qquad \text{(B.14)}$$
$$V(X) = \alpha\beta^2$$

The Beta Distribution The PDF of Beta distribution is given by

$$p(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}, \text{when } x > 0;\ p(x) = 0, \text{when } x \le 0 \quad \text{(B.15)}$$

The mean and variance of Beta distribution are

$$E(X) = \frac{\alpha}{\alpha + \beta}$$
$$V(X) = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)} \tag{B.16}$$

More theoretical distributions can be found in the standard probability and statistics text books. In practice, to determine which kind of a probability distribution that a random variable may have, we can draw a histogram based on sample data and find a theoretical distribution to fit it.

B.2 Multivariate Probability Distributions

For the same sampling space, there may be two or more random variables. In many cases, they may have relationships in the sense of probability. For example, when a sample has a large value of one random variable *X*, probably it would also have a large value of another random variable *Y*. In this case, the two variables are said to be *positively correlated*. Similarly, when a sample has a large value of X, probably, it would have a small value of Y, they are *negatively correlated*. In this section, we will consider how to quantitatively describe the correlation relationships between a set of random variables.

B.2.1 Joint Probability Density Function

The *joint probability distribution* of two random variables, *X* and *Y*, is a function $p(x, y)$ satisfying the following conditions:

- $p(x, y) \geq 0$ for any x and y
- $\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p(x, y)dxdy = 1$
- $P(a \leq X \leq b; c \leq Y \leq d) = \int_a^b \int_c^d p(x, y)dxdy$

Marginal PDF The marginal PDF for *X* is defined by

$$p(x) = \int_{-\infty}^{\infty} p(x, y)dy \tag{B.17}$$

which gives the PDF of X without respect to Y. Similarly, the marginal PDF for Y is defined by

$$p(y) = \int_{-\infty}^{\infty} p(x,y)dx \tag{B.18}$$

Therefore, the PDF of any individual random variable can be obtained from the joint PDF with other random variables.

Conditional PDF Let us consider how to find the probability distribution of a random variable X when the value of a random variable Y is fixed, for example, finding the PDF of height distribution for a population having a certain weight. Conditional probability is one of the most fundamental and important concept in probability theory. The conditional probability of X being in an interval (a,b) for a fixed value of $Y = y$, when $p(y) \neq 0$, can be calculated by

$$P(a \leq X \leq b; Y = y) = \frac{\int_a^b p(x,y)dx}{\int_{-\infty}^{\infty} p(x,y)dx} = \int_a^b \frac{p(x,y)}{p(y)} dx. \tag{B.19}$$

Thus, the *conditional PDF* of X for a fixed value of $Y = y$ should be defined as

$$p(x \mid y) = \frac{p(x,y)}{p(y)}. \tag{B.20}$$

With this definition, Eq. (B.19) can be rewritten as

$$P(a \leq X \leq b; Y = y) = \int_a^b p(x \mid y)dx. \tag{B.21}$$

Independence Two random variables are said to be independent of each other, if

$$p(x,y) = p(x)p(y). \tag{B.22}$$

From this condition, we can find that

$$P(a \leq X \leq b; c \leq Y \leq d) = P(a \leq X \leq b)P(c \leq Y \leq d) \tag{B.23}$$

The independence of two random variables can also be understood from the meanings of marginal PDF and conditional PDF. When the independence condition (B.22) is satisfied, we will have

$$P(a \leq X \leq b) = P(a \leq X \leq b)P(-\infty \leq Y \leq \infty) \tag{B.24}$$

and

$$P(a \le X \le b; Y \le y) = P(a \le X \le b). \tag{B.25}$$

B.2.2 Random Vectors

The above concepts defined for two random variables can be extended to the general case of a set of *m* random variables or an *m*-dimensional random vector $\mathbf{X} = \{X_1, X_2, \cdots, X_m\}$. The joint PDF of $\mathbf{X}$, $p(\mathbf{x})$, satisfies

- $p(\mathbf{x}) \ge 0$, where $\mathbf{x} = \{x_1, x_2, \cdots, x_m\}$
- $\int_{\mathbb{R}^m} p(\mathbf{x})d\mathbf{x} = 1$, where $\mathbb{R}^m$ is the entire *m*-dimensional space
- $P(\mathbf{X} \in \Omega) = \int_{\Omega} p(\mathbf{x})d\mathbf{x}$, for any $\Omega \subset \mathbb{R}^m$

The marginal PDFs are determined by

$$p(x_i) = \int_{\mathbb{R}^{m-1}} p(\mathbf{x}_{-i})d\mathbf{x}_{-i}, \; i = 1, 2, \cdots, m \tag{B.26}$$

where $p(x_i)$ is the marginal PDF of X_i and the subscript $-i$ means X_i is excluded. The random variables $\{X_1, X_2, \cdots, X_m\}$ are called *a set of independent variables* if and only if

$$p(x_1, x_2, \cdots, x_m) = p(x_1) \cdot p(x_2) \cdot \cdots \cdot p(x_m). \tag{B.27}$$

The conditional PDF of X_i for fixed values $\mathbf{x}_{-i}$ of all other random variables is denoted by

$$p(x_i \mid \mathbf{x}_{-i}) = \frac{p(\mathbf{x})}{\int_{-\infty}^{\infty} p(\mathbf{x})dx_i} \tag{B.28}$$

B.2.3 Expected Value, Variance, and Covariance

Besides PDF, there are other numerical measurements used to characterize a set of random variables. Let us first consider the bivariate case: there are only two random variables *X* and *Y*.

Expected Values The marginal means of *X* and *Y* are defined as

$$\mu_X = E(X) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xp(x, y)dxdy \tag{B.29}$$
$$\mu_Y = E(Y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} yp(x, y)dxdy$$

Generally, the expected value of a mapping $g(X,Y)$ ($\mathbb{S}^2 \to \mathbb{R}$) is defined as

$$E[g(X,Y)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)p(x,y)dxdy. \tag{B.30}$$

Example B.3 It is easy to show the following properties

$$E(aX + bY) = aE(X) + bE(Y),\ \text{where } a \text{ and } b \text{ are constant} \tag{B.31}$$

$$E(XY) = E(X)E(Y),\ \text{if } X \text{ and } Y \text{ are independent} \tag{B.32}$$

Variances and Covariance The marginal variances of *X* and *Y* are defined as

$$\begin{aligned} Var(X) = \sigma_X^2 &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-\mu_X)^2 p(x,y)dxdy \\ Var(Y) = \sigma_Y^2 &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (y-\mu_Y)^2 p(x,y)dxdy \end{aligned} \tag{B.33}$$

The covariance between two random variables *X* and *Y* is defined as

$$\begin{aligned} Cov(X,Y) &\equiv E[(X-\mu_X)(Y-\mu_Y)] \\ &= E(XY) - \mu_X\mu_Y \end{aligned} \tag{B.34}$$

From this definition and (B.32), it is easy to derive the following properties:

$$\begin{aligned} &Cov(X,Y) = 0,\ \text{if and only if } X \text{ and } Y \text{ are independent} \\ &Cov(X,Y) = Cov(Y,X) \\ &Cov(X,X) = Var(X) \\ &Var(aX+bY) = a^2Var(X) + b^2Var(Y) + 2abCov(X,Y) \end{aligned} \tag{B.35}$$

Therefore, if and only if *X* and *Y* are independent, we will have

$$Var(X+Y) = Var(X) + Var(Y). \tag{B.36}$$

Multivariate Random Variables The above discussions are easy to be extended to the case having more than two random variables. Let $\mathbf{X} = \{X_1, X_2, \cdots, X_m\}$ be a random vector. The expected value of a mapping $g(\mathbf{X})$ is given by

$$E[g(\mathbf{x})] = \int_{\mathbb{R}^m} g(\mathbf{x})p(\mathbf{x})d\mathbf{x}. \tag{B.37}$$

Especially, when $g(\mathbf{X}) \equiv \mathbf{X}$, we have the mean of $\mathbf{X}$, viz.

$$\mu_{\mathbf{X}} = \int_{\mathbb{R}^m} \mathbf{x} p(\mathbf{x}) d\mathbf{x}. \tag{B.38}$$

All $Cov(X_i, X_j)$, $i, j = 1, 2, \cdots, m$, form the following $m \times m$ symmetric matrix called the covariance matrix of $\mathbf{X}$:

$$Cov(\mathbf{X}) = \begin{bmatrix} Var(X_1) & Cov(X_1, X_2) & \cdots & Cov(X_1, X_m) \\ Cov(X_1, X_2) & Var(X_2) & \cdots & Cov(X_2, X_m) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(X_1, X_m) & Cov(X_2, X_m) & \cdots & Var(X_m) \end{bmatrix}. \tag{B.39}$$

When all components of $\mathbf{X}$ are independent of each other, $Cov(\mathbf{X})$ reduces to a diagonal matrix. Furthermore, if $\mathrm{Var}(X_i) = \sigma^2$ for all $i = 1, 2, \cdots, m$, $Cov(\mathbf{X})$ becomes $\sigma^2 \mathbf{I}$, where $\mathbf{I}$ is the $m \times m$ identity matrix. Sometimes, we use a shortcut notation $\mathbf{C}_{\mathbf{X}}$ to represent $Cov(\mathbf{X})$.

Example B.4 The joint PDF of an *m*-dimensional normal distribution (or a multivariate Gaussian distribution) is given by

$$p(\mathbf{X}) = (2\pi)^{-m/2} (\det \mathbf{C}_{\mathbf{X}})^{-1/2} \exp\left\{ -\frac{1}{2} \left[\mathbf{X} - \mu_{\mathbf{X}} \right]^T \mathbf{C}_{\mathbf{X}}^{-1} \left[\mathbf{X} - \mu_{\mathbf{X}} \right] \right\} \tag{B.40}$$

B.3 Bayes' Theorem

Let X and Y be two random variables. Using the conditional PDF of X given Y defined in (B.20), we have

$$p(x, y) = p(y) p(x \mid y) \tag{B.41}$$

On the other hand, using the conditional PDF of Y given X, we have

$$p(x, y) = p(x) p(y \mid x). \tag{B.42}$$

Combination of above two equations yields the well-known *Bayes' theorem*:

$$p(x \mid y) = \frac{p(x) p(y \mid x)}{p(y)}. \tag{B.43}$$

Because $p(y)$ is the marginal PDF of *Y*, using (B.18) and (B.42), we obtain

$$p(y) = \int_{-\infty}^{\infty} p(x, y)dx = \int_{-\infty}^{\infty} p(x)p(y \mid x)dx. \tag{B.44}$$

Thus, the Bayes' theorem can be rewritten as

$$p(x \mid y) = \frac{p(x)p(y \mid x)}{\int_{-\infty}^{\infty} p(x)p(y \mid x)dx}. \tag{B.45}$$

Calculating the denominator in (B.45) is not easy, but fortunately it is not necessary. In fact, the denominator only plays the role of a normalizing constant to assure $\int_{-\infty}^{\infty} p(x \mid y)dx = 1$. Let its value be $1/c$, the Bayes' theorem then becomes

$$p(x \mid y) = cp(x)p(y \mid x). \tag{B.46}$$

The Bayes' theorem for two random vectors $\mathbf{X}$ and $\mathbf{Y}$ can be derived by the same way and is given by

$$p(\mathbf{x} \mid \mathbf{y}) = cp(\mathbf{x})p(\mathbf{y} \mid \mathbf{x}). \tag{B.47}$$

Bayes' theorem is the basis of *Bayesian inference* approach introduced in Chap. 4 of this book.

B.4 Random Fields, Auto- and Cross-Covariance Functions

After considering random variables and random vectors, we finally introduce the concepts of *random fields*. A random field (or a stochastic field) is a function that varies randomly in a time interval and/or a spatial domain. At each point of its definition domain, its value is a random variable, but its values at different points are generally correlated in one way or another. For examples, the influx to a like is a random field of time, the hydraulic conductivity of a heterogeneous aquifer is a random field of location, and the precipitation rate is a random function of both time and location.

We will use $X(\mathbf{t}), \mathbf{t} \in (\mathrm{T})$ to denote a random field, where $\mathbf{t}$ represents independent (time and/or spatial) variables and (T) is the definition domain of the random field. The covariance between two random variables $X(\mathbf{t}_1)$ and $X(\mathbf{t}_2)$, i.e., $Cov[X(\mathbf{t}_1), X(\mathbf{t}_2)]$, where $\mathbf{t}_1, \mathbf{t}_2 \in (\mathrm{T})$, is called the *auto-covariance* of $X(\mathbf{t})$ associated with the two points. When $\mathbf{t}_1 = \mathbf{t}_2$, it becomes the variance of $X(\mathbf{t})$ at a point. Auto-covariance characterizes the correlation structure within a random field.

Furthermore, let $X(\mathbf{t})$ and $Y(\mathbf{t})$ be two random fields defined on the same domain. For any points $\mathbf{t}_1, \mathbf{t}_2 \in (\mathrm{T})$, the covariance between two random variables $X(\mathbf{t}_1)$ and $Y(\mathbf{t}_2)$, (i.e., $Cov[X(\mathbf{t}_1), Y(\mathbf{t}_2)]$) is called a cross-covariance of $X(\mathbf{t})$ and $Y(\mathbf{t})$ associated with the two points. Cross-covariance characterizes the correlation structure between two random fields.

Random field is one of most important tool used for distributed parameter estimation. More details on this topic are given in Chap. 6 of this book, including the use of Gaussian Random Field, Markov Random Field, and others.

Appendix C Matrix Algebra and Matrix Calculus

The first section of this appendix is a brief review on matrix algebra, including basic operation rules, matrix inversion, norms, orthogonality, and eigenvalue decomposition, and etc. The second section is a brief introduction on matrix calculus. Using the notations of vector and matrix, differentiation and other operations on multivariate functions, or more generally, on mappings will become more concise and clear. This appendix covers only materials that are needed for reading this book. For more advanced topics, readers are referred to Laub (2005) and Gentle (2007).

C.1 Matrix Algebra

C.1.1 Definitions and Important Operation Rules

- *Matrix:* A matrix with *m* rows and *n* columns is denoted by $\mathbf{A} = [a_{ij}]_{m\times n}$, where a_{ij} represents an element located at the cross of row *i* and column *j*. We assume that all elements are real numbers. If $a_{ij} = a_{ji}$ for all *i* and *j*, **A** is called *symmetric matrix*. If all $a_{ij} = 0$ when $i \neq j$, $\mathbf{A}$ is called a *diagonal matrix*; Further, if all $a_{ii} = 1$, the diagonal matrix is called an *identity matrix* and denoted by **I**.
- *Transpose:* The transpose of a matrix is defined as $\mathbf{A}^{\mathrm{T}} = [a_{ji}]_{n\times m}$.
- *Vector:* An *m*-dimensional *column vector* is defined as $\mathbf{a} = [a_1, a_2, \cdots, a_m]^{\mathrm{T}}$ which can be seen as a $m \times 1$ matrix, while an *m*-dimensional *row vector* $\mathbf{a}^{\mathrm{T}}$ can be seen as a $1 \times m$ matrix.
- *Addition:* Let $\mathbf{A} = [a_{ij}]_{m\times n}$ and $\mathbf{B} = [b_{ij}]_{m\times n}$ be two $m \times n$ matrices. The summation of them denoted by $\mathbf{A} + \mathbf{B}$ is a matrix $\mathbf{C} = [c_{ij}]_{m\times n}$, where $c_{ij} = a_{ij} + b_{ij}$.
- *Multiplication:* Let $\mathbf{A} = [a_{ir}]_{m\times k}$ and $\mathbf{B} = [b_{rj}]_{k\times n}$. The product of them, denoted by $\mathbf{AB}$, is a matrix $\mathbf{C} = [c_{ij}]_{m\times n}$, where $c_{ij} = \sum_{r=1}^{k} a_{ir} b_{rj}$.

The following definitions are applicable only to square matrices.

- *Determinant:* The determinant of a square matrix $\mathbf{A}$ is denoted by $\det(\mathbf{A})$. If $\det(\mathbf{A}) = 0$, $\mathbf{A}$ is called a singular matrix.
- *Inverse matrix:* For a $m \times m$ square matrix $\mathbf{A}$, if there is a $m \times m$ square matrix $\mathbf{B}$ such that $\mathbf{AB} = \mathbf{BA} = \mathbf{I}$, then $\mathbf{B}$ is called the inverse of $\mathbf{A}$ and denoted by $\mathbf{A}^{-1}$. When $\mathbf{A}^{-1}$ exists, $\mathbf{A}$ is called invertible. A nonsingular matrix is invertible. For a rectangular matrix, we can find its generalized inverse (see Sect. 2.2.2).
- *Orthogonality:* A square matrix $\mathbf{A}$ is orthogonal when $\mathbf{A}^{\mathrm{T}}\mathbf{A} = \mathbf{A}\mathbf{A}^{\mathrm{T}} = \mathbf{I}$. In other words, for an orthogonal matrix $\mathbf{A}$, we must have $\mathbf{A}^{\mathrm{T}} = \mathbf{A}^{-1}$.
- *Positive definite matrix:* A symmetric $m \times m$ square matrix $\mathbf{A}$ is called *positive definite*, if the quadratic form $Q \equiv \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} > 0$ for any *m*-dimensional column vector $\mathbf{x}$. For the case of $Q \geq 0$, $\mathbf{A}$ is called *positive semidefinite*. Similarly, we can define negative definite and negative semi-definite matrices according to $Q < 0$ and $Q \leq 0$, respectively.
- *Trace:* The trace of an $m \times m$ square matrix $\mathbf{A}$ is the summation of all elements on its diagonal, namely, $\mathrm{Tr}(\mathbf{A}) = \sum_{i=1}^{m} a_{ii}$.

From the above definitions, the following important matrix operation rules can be obtained directly:

- $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$
- $(\mathbf{A} + \mathbf{B}) + \mathbf{C} = \mathbf{A} + (\mathbf{B} + \mathbf{C})$
- $\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$
- In general, $\mathbf{AB} \neq \mathbf{BA}$
- $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$
- $\alpha(\mathbf{A} + \mathbf{B}) = \alpha\mathbf{A} + \alpha\mathbf{B}$, where α is a scalar
- $\alpha(\mathbf{AB}) = (\alpha\mathbf{A})\mathbf{B} = \mathbf{A}(\alpha\mathbf{B})$ α
- $(\mathbf{A}^{\mathrm{T}})^{\mathrm{T}} = \mathbf{A}$
- $(\alpha\mathbf{A})^{\mathrm{T}} = \alpha\mathbf{A}^{\mathrm{T}}$
- $(\mathbf{A} + \mathbf{B})^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}} + \mathbf{B}^{\mathrm{T}}$
- $(\mathbf{AB})^{\mathrm{T}} = \mathbf{B}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}$
- $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$
- $(\mathbf{A}^{-1})^{\mathrm{T}} = (\mathbf{A}^{\mathrm{T}})^{-1}$
- $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$
- $\mathrm{Tr}(\alpha\mathbf{A}) = \alpha\mathrm{Tr}(\mathbf{A})$
- $\mathrm{Tr}(\mathbf{A} + \mathbf{B}) = \mathrm{Tr}(\mathbf{A}) + \mathrm{Tr}(\mathbf{B})$
- $\mathrm{Tr}(\mathbf{AB}) = \mathrm{Tr}(\mathbf{BA})$

C.1.2 Eigenvalue Decomposition

The eigenvalues and their associated eigenvectors of an $m \times m$ square matrix $\mathbf{A}$ satisfy the eigenvector equation

$$\mathbf{Ax} = \lambda\mathbf{x} \text{ or } (\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}. \tag{C.1}$$

All nonzero eigenvalues then can be solved from the following characteristic equation

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0. \tag{C.2}$$

The MATLAB command eig can be used to find all eigenvalues and associated eigenvectors of a square matrix. Eigenvalue-eigenvector pairs have the following important properties:

- Matrix A and all eigenvalues of a symmetric matrix are real.
- The value of $det(\mathbf{A})$ is equal to the product of the absolute values of all eigenvalues of $\mathbf{A}$. Thus, if all eigenvalues are nonzero, then $\det(\mathbf{A}) \neq 0$, and as a result, $\mathbf{A}$ is nonsingular and invertible.
- Matrix $\mathbf{A}$ and its inverse $\mathbf{A}^{-1}$ have the same eigenvalues.
- Eigenvectors corresponding to different eigenvalues are linear independent of each other. Furthermore, when the matrix is symmetric, these vectors are perpendicular to each other.

Now assume that matrix $\mathbf{A}$ has *m* nonzero eigenvalues $\{\lambda_1, \lambda_2, \cdots, \lambda_m\}$ with corresponding linear independent eigenvectors $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$. Using $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ as columns and using $\{\lambda_1, \lambda_2, \cdots, \lambda_m\}$ as diagonals, respectively, we can compose a matrix $\mathbf{Q}$ and a diagonal matrix Λ as shown below

$$\mathbf{Q} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \end{bmatrix} \text{and } \Lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_m \end{bmatrix}. \tag{C.3}$$

Using (C.1), we have

$$\begin{aligned} \mathbf{AQ} &= \mathbf{A}\begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_m \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{Ax}_1 & \mathbf{Ax}_2 & \cdots & \mathbf{Ax}_m \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1\mathbf{x}_1 & \lambda_2\mathbf{x}_2 & \cdots & \lambda_m\mathbf{x}_m \end{bmatrix} = \mathbf{Q}\Lambda. \end{aligned} \tag{C.4}$$

This equation gives the following eigenvalue decomposition of $\mathbf{A}$

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^{-1}. \tag{C.5}$$

When a matrix $\mathbf{A}$ has the above eigenvalue decomposition, it is called a diagonalizable matrix. Not all matrices are diagonalizable. But, according to the above mentioned properties of eigenvalues, symmetric matrices must be diagonalizable, and moreover, because all eigenvectors of a symmetric matrix are perpendicular to each other, we have $\mathbf{Q}^{-1} = \mathbf{Q}^{\mathrm{T}}$. Therefore, for a symmetric matrix $\mathbf{A}$, (C.5) can be replaced by the following and easier to be calculated form

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^{\mathrm{T}}. \tag{C.6}$$

Powers of a Matrix For a square matrix $\mathbf{A}$, its *k* power, $\mathbf{A}^k$, is the product of *k* $\mathbf{A}$'s, (i.e., $\mathbf{A}^2 = \mathbf{AA}$), $\mathbf{A}^3 = \mathbf{AAA}$, and so forth. When *k* is large, $\mathbf{A}^k$ is not easy to be calculated. But, for a diagonal matrix $\mathbf{A} = \left[a_{ii}\right]$, it is easy to find that $\mathbf{A}^k = \left[a_{ii}^k\right]$. Furthermore, when $\mathbf{A}$ is diagonalizable, we can use its eigenvalue decomposition (C.5) to find

$$\mathbf{A}^k = \mathbf{Q}\Lambda\mathbf{Q}^{-1}\mathbf{Q}\Lambda\mathbf{Q}^{-1}\cdots\mathbf{Q}\Lambda\mathbf{Q}^{-1} = \mathbf{Q}\Lambda^k\,\mathbf{Q}^{-1}, \tag{C.7}$$

where the diagonal matrix $\Lambda^k = \left[\lambda_i^k\right]$.

The Square Root of a Matrix A matrix $\mathbf{B}$ is said to be the square root of a square matrix $\mathbf{A}$ and denoted by $\mathbf{A}^{1/2}$ if $\mathbf{BB} = \mathbf{A}$. For a diagonal matrix $\mathbf{A} = \left[a_{ii}\right]$, we have $\mathbf{A}^{1/2} = \left[a_{ii}^{1/2}\right]$. When $\mathbf{A}$ is diagonalizable, we have $\mathbf{A}^{1/2} = \mathbf{Q}\Lambda^{1/2}\mathbf{Q}^{-1}$. Furthermore, when $\mathbf{A}$ is symmetric and positive definite, its square root is unique and given by $\mathbf{A}^{1/2} = \mathbf{Q}\Lambda^{1/2}\mathbf{Q}^{\mathrm{T}}$.

C.1.3 Matrix Norms

We have defined the concept *norm* for general function spaces in Appendix A. Let us consider an operator space $\mathbb{A}$ consisting of all real matrices with *m* rows and *n* columns. Each element $\mathbf{A} \in \mathbb{A}$ is an operator that transforms vector $\mathbf{x} \in \mathbb{R}^n$ into vector $\mathbf{y} \in \mathbb{R}^m$. With the L_p norms defined in Example A.3, for any vector $\mathbf{x} \in \mathbb{R}^n$, we have

$$\left\|\mathbf{Ax}\right\|_p \le c\left\|\mathbf{x}\right\|_p. \tag{C.8}$$

Moreover, from the operation rules given in C.1.1, for any two vectors $\mathbf{x}_1 \in \mathbb{R}^n$ and $\mathbf{x}_2 \in \mathbb{R}^n$, we have

$$\mathbf{A}(\alpha\mathbf{x}_1 + \beta\mathbf{x}_2) = \alpha\mathbf{A}\mathbf{x}_1 + \beta\mathbf{A}\mathbf{x}_2. \tag{C.9}$$

Comparing the above two equations with Eq. (A.23) and Eq. (A.24), we know that matrix $\mathbf{A}$ is a bounded linear operator. Thus, according to Eq. (A.25), the norm of $\mathbf{A}$ can be defined as

$$\left\|\mathbf{A}\right\|_p \equiv \sup_{\|\mathbf{x}\|\neq 0} \frac{\left\|\mathbf{A}\mathbf{x}\right\|_p}{\left\|\mathbf{x}\right\|_p}. \tag{C.10}$$

Especially, for $p = 1, 2, \infty$, we have

$$\left\|\mathbf{A}\right\|_1 = \max_{1\le j\le n} \sum_{i=1}^{m} \left|a_{ij}\right|, \text{ the maximum of the comumn sums,} \tag{C.11}$$

$$\left\|\mathbf{A}\right\|_2^2 = \lambda_{\max}(\mathbf{A}^{\mathrm{T}}\mathbf{A}), \text{the maximum eigenvalue of } \mathbf{A}^{\mathrm{T}}\mathbf{A}, \tag{C.12}$$

$$\left\|\mathbf{A}\right\|_\infty = \max_{1\le i\le m} \sum_{j=1}^{n} \left|a_{ij}\right|, \text{ the maximum of the row sums.} \tag{C.13}$$

Two norms, $\left\|\cdot\right\|_p$ and $\left\|\cdot\right\|_q$, are called equivalent if there are constant numbers *c* and *d* such that the following inequality is satisfied:

$$c\left\|\cdot\right\|_q \le \left\|\cdot\right\|_p \le d\left\|\cdot\right\|_q. \tag{C.14}$$

An equivalent and alternative norm of $\left\|\mathbf{A}\right\|_2^2$ given in (C.12) is the Frobenius norm defined by

$$\left\|\mathbf{A}\right\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2 = \operatorname{Tr}(\boldsymbol{A}^T\boldsymbol{A}), \text{the trace of } (\boldsymbol{A}^T\boldsymbol{A}). \tag{C.15}$$

C.2 Matrix Calculus

C.2.1 Definitions

We have known how to differentiate a smooth scalar function $y = f(x)$ with a scalar variable x. When the input and output relationships of a mode are considered as a transformation $\mathbf{y} = \mathcal{F}(\mathbf{x})$, differentiation of vector and matric functions will becomes necessary. Let us consider several cases used in this book.

Scalar w.r.t. Vector $\left[y = f(\mathbf{x}) : \ \mathbb{R}^m \to \mathbb{R} \right]$ In this case, y is a scalar and $\mathbf{x}$ is an m-dimensional column vector (or an $m \times 1$ matrix),

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix}^{\mathrm{T}}. \tag{C.16}$$

The gradient of $f(\mathbf{x})$ is defined as the following column vector (or $m \times 1$ matrix):

$$\nabla_{\mathbf{x}} y \equiv \frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} & \dfrac{\partial f}{\partial x_2} & \cdots & \dfrac{\partial f}{\partial x_m} \end{bmatrix}^{\mathrm{T}}. \tag{C.17}$$

The second-order gradient or the Hessian matrix of $f(\mathbf{x})$ is defined as

$$\nabla_{\mathbf{x}}^2 y \equiv \frac{\partial^2 f}{\partial \mathbf{x}^2} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_m} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_m \partial x_1} & \dfrac{\partial^2 f}{\partial x_m \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_m^2} \end{bmatrix}. \tag{C.18}$$

Vector w.r.t. scalar $\left[\mathbf{y} = \mathbf{f}(x) : \ \mathbb{R} \to \mathbb{R}^n \right]$ In this case, x is a scalar variable and $\mathbf{y}$ is an n-dimensional column vector

$$\mathbf{y} = \begin{bmatrix} f_1 & f_2 & \cdots & f_n \end{bmatrix}^{\mathrm{T}}. \tag{C.19}$$

The gradient of $\mathbf{y}$ w.r.t x is defined as the following row vector

$$\nabla_x \mathbf{y} \equiv \frac{\partial \mathbf{f}}{\partial x} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_2}{\partial x} & \cdots & \dfrac{\partial f_m}{\partial x} \end{bmatrix}, \tag{C.20}$$

while the second-order gradient of $\mathbf{y}$ is

$$\nabla_x^2 \mathbf{y} \equiv \frac{\partial^2 \mathbf{f}}{\partial x^2} = \begin{bmatrix} \dfrac{\partial^2 f_1}{\partial x^2} & \dfrac{\partial^2 f_2}{\partial x^2} & \cdots & \dfrac{\partial^2 f_m}{\partial x^2} \end{bmatrix}. \tag{C.21}$$

Vector w.r.t. Vector $\left[\mathbf{y}=\mathbf{f}(\mathbf{x}):\ \mathbb{R}^m \to \mathbb{R}^n\right]$ In this case, vector x is defined in (C.16) and vector $\mathbf{y}$ is defined in (C.19). The derivative of **y** w.r.t. **x** is defined as the following Jacobian matrix

$$\mathbf{J} \equiv \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \equiv \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial x_m} & \frac{\partial f_2}{\partial x_m} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix}. \tag{C.22}$$

Scalar w.r.t. Matrix $\left[y = f(\mathbf{X}):\ \mathbb{R}^{m\times n} \to \mathbb{R}\right]$ In this case, y is a scalar and $\mathbf{X}$ is an $m \times n$ matrix $\left[x_{ij}\right]$. The gradient of y w.r.t. $\mathbf{X}$ is defined as

$$\nabla_{\mathbf{X}} y \equiv \frac{\partial f}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial f}{\partial x_{11}} & \frac{\partial f}{\partial x_{12}} & \cdots & \frac{\partial f}{\partial x_{1n}} \\ \frac{\partial f}{\partial x_{21}} & \frac{\partial f}{\partial x_{22}} & \cdots & \frac{\partial f}{\partial x_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial x_{m1}} & \frac{\partial f}{\partial x_{m2}} & \cdots & \frac{\partial f}{\partial x_{mn}} \end{bmatrix}. \tag{C.23}$$

Matrix w.r.t. scalar $\left[\mathbf{Y} = f(x):\ \mathbb{R} \to \mathbb{R}^{m\times n}\right]$ In this case, x is a scalar and $\mathbf{Y}$ is an $m \times n$ matrix $\left[y_{ij}\right]$, and the gradient of $\mathbf{Y}$ w.r.t. x is defined as

$$\nabla_x \mathbf{Y} \equiv \frac{\partial \mathbf{Y}}{\partial x} = \begin{bmatrix} \frac{\partial y_{11}}{\partial x} & \frac{\partial y_{12}}{\partial x} & \cdots & \frac{\partial y_{1n}}{\partial x} \\ \frac{\partial y_{21}}{\partial x} & \frac{\partial y_{22}}{\partial x} & \cdots & \frac{\partial y_{2n}}{\partial x} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_{m1}}{\partial x} & \frac{\partial y_{m2}}{\partial x} & \cdots & \frac{\partial y_{mn}}{\partial x} \end{bmatrix}. \tag{C.24}$$

Other types of differentiation, such as the gradient of vector with respect to matrix and various high-order gradients, are not required for reading this book.

C.2.2 Important Operation Rules

The operation rules on matrix differentiation, including the chain rules, can be derived directly from the definitions of gradients given above. For the gradient of scalar w.r.t vector, $\dfrac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$, we have

$\dfrac{\partial(au)}{\partial \mathbf{x}} = a\dfrac{\partial u}{\partial \mathbf{x}}$	a is a constant $u = u(\mathbf{x})$
$\dfrac{\partial(u+v)}{\partial \mathbf{x}} = \dfrac{\partial u}{\partial \mathbf{x}} + \dfrac{\partial v}{\partial \mathbf{x}}$	$u = u(\mathbf{x})$ $v = v(\mathbf{x})$
$\dfrac{\partial(\mathbf{u}^T\mathbf{v})}{\partial \mathbf{x}} = \dfrac{\partial \mathbf{u}}{\partial \mathbf{x}}\mathbf{v} + \dfrac{\partial \mathbf{v}}{\partial \mathbf{x}}\mathbf{u}$	$\mathbf{u} = \mathbf{u}(\mathbf{x})$ $\mathbf{v} = \mathbf{v}(\mathbf{x})$
$\dfrac{\partial g}{\partial \mathbf{x}} = \dfrac{\partial g}{\partial u}\dfrac{\partial u}{\partial \mathbf{x}}$	$g = g(u(\mathbf{x}))$, Chain rule
$\dfrac{\partial(\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x})}{\partial \mathbf{x}} = 2\mathbf{A}\mathbf{x}$	$\mathbf{A}$ is a matrix independent of $\mathbf{x}$
$\dfrac{\partial^2(\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x})}{\partial^2 \mathbf{x}} = \mathbf{A} + \mathbf{A}^{\mathrm{T}}$	

For the gradient of vector w.r.t scalar, $\dfrac{\partial \mathbf{f}(x)}{\partial x}$, we have

$\dfrac{\partial(a\mathbf{u})}{\partial x} = a\dfrac{\partial \mathbf{u}}{\partial x}$	a is a constant $\mathbf{u} = \mathbf{u}(x)$
$\dfrac{\partial(\mathbf{A}\mathbf{u})}{\partial x} = \dfrac{\partial \mathbf{u}}{\partial x}\mathbf{A}^{\mathrm{T}}$	$\mathbf{A}$ is a matrix independent of x
$\dfrac{\partial \mathbf{u}^{\mathrm{T}}}{\partial x} = \left(\dfrac{\partial \mathbf{u}}{\partial x}\right)^{\mathrm{T}}$	$\mathbf{u} = \mathbf{u}(x)$
$\dfrac{\partial \mathbf{g}}{\partial x} = \dfrac{\partial \mathbf{u}}{\partial x}\dfrac{\partial \mathbf{g}}{\partial \mathbf{u}}$	$\mathbf{g} = \mathbf{g}(\mathbf{u}(x))$

For the gradient of vector w.r.t vector, $\dfrac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}$, we have

$\frac{\partial(a\mathbf{u})}{\partial\mathbf{x}} = a\frac{\partial\mathbf{u}}{\partial\mathbf{x}}$	a is a constant $\mathbf{u} = \mathbf{u}(x)$
$\frac{\partial(a\mathbf{u})}{\partial\mathbf{x}} = a\frac{\partial\mathbf{u}}{\partial\mathbf{x}} + \frac{\partial a}{\partial\mathbf{x}}\mathbf{u}^{\mathrm{T}}$	$a = a(\mathbf{x})$ $\mathbf{u} = \mathbf{u}(x)$
$\frac{\partial(\mathbf{A}\mathbf{x})}{\partial\mathbf{x}} = \mathbf{A}^{\mathrm{T}}$	$\mathbf{A}$ is a matrix independent of $\mathbf{x}$
$\frac{\partial(\mathbf{x}^{\mathrm{T}}\mathbf{A})}{\partial\mathbf{x}} = \mathbf{A}$	$\mathbf{A}$ is a matrix independent of $\mathbf{x}$
$\frac{\partial(\mathbf{A}\mathbf{u})}{\partial\mathbf{x}} = \frac{\partial\mathbf{u}}{\partial\mathbf{x}}\mathbf{A}^{\mathrm{T}}$	$\mathbf{A}$ is a matrix independent of $\mathbf{x}$ $\mathbf{u} = \mathbf{u}(\mathbf{x})$
$\frac{\partial\mathbf{g}}{\partial\mathbf{x}} = \frac{\partial\mathbf{u}}{\partial\mathbf{x}}\frac{\partial\mathbf{g}}{\partial\mathbf{u}}$	$\mathbf{g} = \mathbf{g}(\mathbf{u}(\mathbf{x}))$

For the gradient of scalar w.r.t matrix, $\frac{\partial f(\mathbf{X})}{\partial\mathbf{X}}$, we have

$\frac{\partial(au)}{\partial\mathbf{X}} = a\frac{\partial u}{\partial\mathbf{X}}$	a is a constant $u = u(\mathbf{X})$
$\frac{\partial(uv)}{\partial\mathbf{X}} = u\frac{\partial v}{\partial\mathbf{X}} + v\frac{\partial u}{\partial\mathbf{X}}$	$u = u(\mathbf{X})$ $v = v(\mathbf{X})$
$\frac{\partial g}{\partial\mathbf{X}} = \frac{\partial g}{\partial u}\frac{\partial u}{\partial\mathbf{X}}$	$g = g(u(\mathbf{X}))$
$\frac{\partial\mathrm{Tr}(\mathbf{X}\mathbf{A})}{\partial\mathbf{X}} = \mathbf{A}^{\mathrm{T}}$	$\mathbf{A}$ is a matrix independent of $\mathbf{X}$
$\frac{\partial\mathrm{Tr}(\mathbf{X}^{\mathrm{T}}\mathbf{A})}{\partial\mathbf{X}} = \mathbf{A}$	$\mathbf{A}$ is a matrix independent of $\mathbf{X}$
$\frac{\partial\mathrm{Tr}(\mathbf{X}^{-1}\mathbf{A})}{\partial\mathbf{X}} = -\mathbf{X}^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{X}^{-1}$	$\mathbf{A}$ is a matrix independent of $\mathbf{X}$

C.2.3 The Gradient of a Parameterized Matrix

Let $\mathbf{A}(\alpha) = \left[a_{ij}(\alpha) \right]$ be an $m \times n$ matrix parameterized by a parameter α . The gradient of $\mathbf{A}$ w.r.t. parameter α is

$$\frac{\partial \mathbf{A}}{\partial \alpha} = \begin{bmatrix} \frac{\partial a_{11}}{\partial \alpha} & \frac{\partial a_{12}}{\partial \alpha} & \cdots & \frac{\partial a_{1n}}{\partial \alpha} \\ \frac{\partial a_{21}}{\partial \alpha} & \frac{\partial a_{22}}{\partial \alpha} & \cdots & \frac{\partial a_{2n}}{\partial \alpha} \\ \cdots & \cdots & \ddots & \vdots \\ \frac{\partial a_{m1}}{\partial \alpha} & \frac{\partial a_{m2}}{\partial \alpha} & \cdots & \frac{\partial a_{mn}}{\partial \alpha} \end{bmatrix}. \tag{C.25}$$

Because we always have $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$, differentiate this equation yields

$$\frac{\partial (\mathbf{A}^{-1}\mathbf{A})}{\partial \alpha} = \mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial \alpha} + \frac{\partial \mathbf{A}^{-1}}{\partial \alpha} \mathbf{A} = \mathbf{0}. \tag{C.26}$$

From this equation, we can obtain

$$\frac{\partial \mathbf{A}^{-1}}{\partial \alpha} = -\mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial \alpha} \mathbf{A}^{-1}. \tag{C.27}$$

References

Aanonsen, S.I., Eydinov, D.: A multiscale method for distributed parameter estimation with application to reservoir history matching. Comput. Geosci. **10**(1), 97–117 (2006)

Aanonsen, S., Nævdal, G., Oliver, D., Reynolds, A., Vallès, B.: The ensemble Kalman filter in reservoir engineering—A review. SPE J. **14s**(3), 393–412 (2009)

Abbott, M.B., Refsgaard, J.C.: Distributed hydrological modelling. Water science and technology library, vol. 22. Kluwer, Dordrecht (1996)

Agbalaka, C.C., Oliver, D.S.: Application of the EnKF and localization to automatic history matching of facies distribution and production data. Math. Geosci. **40**(4), 353–374 (2008). doi:10.1007/s11004-008-9155-7

Aghasi, A., Mendoza-Sanchez, I., Miller, E.L., Ramsburg, C.A., Abriola, L.M.: A geometric approach to joint inversion with applications to contaminant source zone characterization. Inverse Probl. **29**(11), 115014 (2013)

Aghasi, A., Kilmer, M., Miller, E.L.: Parametric level set methods for inverse problems. SIAM J. Imagin. Sci. **4**(2), 618–650 (2011)

Ahlfeld, D.P., Mulvey, J.M., Pinder, G.F., Wood, E.F.: Contaminated groundwater remediation design using simulation, optimization, and sensitivity theory: 1. Model development. Water Resour. Res. **24**(3), 431–441 (1988)

Ahmed, S., De Marsily, G.: Comparison of geostatistical methods for estimating transmissivity using data on transmissivity and specific capacity. Water Resour. Res. **23**(9), 1717–1737 (1987)

Ajami, N.K., Duan, Q., Sorooshian, S.: An integrated hydrologic Bayesian multimodel combination framework: confronting input, parameter, and model structural uncertainty in hydrologic prediction. Water Resour. Res. **43**(1), W01403 (2007). doi:10.1029/2005wr004745

Akaike, H.: Fitting autoregressive models for prediction. Ann. Inst. Stat. Math. **21**, 243–247 (1969)

Akaike, H.: A new look at the statistical model identification. Autom. Control IEEE Trans. **19**(6), 716–723 (1974)

Alaña, J.E., Theodoropoulos, C.: Optimal spatial sampling scheme for parameter estimation of nonlinear distributed parameter systems. Comput. Chem. Eng. **45,** 38–49 (2012)

Alcolea, A., Carrera, J., Medina, A.: Pilot points method incorporating prior information for solving the groundwater flow inverse problem. Adv. Water Resour. **29**(11), 1678–1689 (2006)

Alcolea, A., Carrera, J., Medina, A.: Regularized pilot points method for reproducing the effect of small scale variability: application to simulations of contaminant transport. J. Hydrol. **355**(1), 76–90 (2008)

Alfaro, I., Yvonnet, J., Chinesta, F., Cueto, E.: A study on the performance of natural neighbour-based Galerkin methods. Int. J. Numer. Methods Eng. **71**(12), 1436–1465 (2007)

Altmann–Dieses, A.E., Schlöder, J.P., Bock, H.G., Richter, O.: Optimal experimental design for parameter estimation in column outflow experiments. Water Resour. Res. **38**(10), 4–11 (2002)

N-Z. Sun, A. Sun, *Model Calibration and Parameter Estimation,*
DOI 10.1007/978-1-4939-2323-6

Ameur, H.B., Chavent, G., Jaffré, J.: Refinement and coarsening indicators for adaptive parametrization: application to the estimation of hydraulic transmissivities. Inverse Probl. **18**(3), 775 (2002)

Ameur, H.B., Clément, F., Weis, P., Chavent, G.: The multidimensional refinement indicators algorithm for optimal parameterization. J. Inverse Ill-Posed Probl. **16**(2), 107–126 (2008)

Andersen, M., Dahl, J., Liu, Z., Vandenberghe, L. (eds.): Interior-point methods for large-scale cone programming. MIT Press, Cambridge (2011)

Anderson, J.L.: An ensemble adjustment Kalman filter for data assimilation. Mon. Weather Rev. **129**(12), 2884–2903 (2001)

Anderson, J.L.: Spatially and temporally varying adaptive covariance inflation for ensemble filters. Tellus A. **61**(1), 72–83 (2009)

Anderson, J.L., Anderson, S.L.: A Monte Carlo implementation of the nonlinear filtering problem to produce ensemble assimilations and forecasts. Mon. Weather Rev. **127**(12), 2741–2758 (1999)

Anderson, B.D.O., Moore, J.B.: Optimal filtering. Prentice-Hall information and system sciences series. Prentice-Hall, Englewood Cliffs (1979)

Andrienko, G., Andrienko, N., Jankowski, P., Keim, D.A., Kraak, M.-J., Maceachren, A.M., Wrobel, S.: Geovisual analytics for spatial decision support: setting the research agenda. In. Universität Konstanz (2007)

Andrieu, C., Doucet, A., Tadic, V.B.: On-line parameter estimation in general state-space models. In: Decision and control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on, 12–15 December 2005. pp. 332–337 (2005)

ASCE.: Committee on the application of ANNs in hydrology artificial neural networks in hydrology I: preliminary concepts. J. Hydrol. Eng. **5**(2), 115–123 (2000)

Ascough, J.C., Maier, H.R., Ravalico, J.K., Strudley, M.W.: Future research challenges for incorporation of uncertainty in environmental and ecological decision-making. Ecol. Model. **219**(3–4), 383–399 (2008)

Asefa, T., Kemblowski, M., McKee, M., Khalil, A.: Multi-time scale stream flow predictions: the support vector machines approach. J. Hydrol. **318**(1), 7–16 (2006)

Aster, R.C., Borchers, B., Thurber, C.H.: Parameter estimation and inverse problems. 2nd edn. Academic, Waltham (2013)

Atkinson, P.M.: GeoENV VII—geostatistics for environmental applications. Springer, New York (2010)

Atkinson, A.C., Donev, A.N., Tobias, R.D.: Optimum experimental designs, with SAS. Oxford University Press, Oxford (2007)

Ayvaz, M.T.: Simultaneous determination of aquifer parameters and zone structures with fuzzy c-means clustering and meta-heuristic harmony search algorithm. Adv. Water Resour. **30**(11), 2326–2338 (2007)

Ayyub, B.M., Klir, G.J.: Uncertainty modeling and analysis in engineering and the sciences. Chapman & Hall/CRC, Boca Raton (2006)

Baalousha, H., Köngeter, J.: Stochastic modelling and risk analysis of groundwater pollution using FORM coupled with automatic differentiation. Adv. Water Resour. **29**(12), 1815–1832 (2006)

Bammer, G., Smithson, M.: Uncertainty and risk: multidisciplinary perspectives. Earthscan risk in society series. Earthscan, London (2008)

Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. (TOMS). **22**(4), 469–483 (1996)

Bard, Y.: Nonlinear parameter estimation. Academic, New York (1974)

Barhen, J., Reister, D.B.: Uncertainty analysis on sensitivities generated using automatic differentiation, Computational Science and its Applications, ICCSA: PT 2, 70–77 (2003)

Barthelmann, V., Novak, E., Ritter, K.: High dimensional polynomial interpolation on sparse grids. Adv. Comput. Math. **12**(4), 273–288 (2000)

Basak, D., Pal, S., Patranabis, D.C.: Support vector regression. Neural. Info. Process. Lett. Rev. **11**(10), 203–224 (2007)

Bates, J.M., Granger, C.W.J.: The combination of forecasts. OR, 451–468 (1969)

Batu, V.: Applied flow and solute transport modeling in aquifers: fundamental principles and analytical and numerical methods. Taylor & Francis, Boca Raton (2006)

Baú, D.A., Mayer, A.S.: Optimal design of pump-and-treat systems under uncertain hydraulic conductivity and plume distribution. J. Contam. Hydrol. **100**(1), 30–46 (2008)

Bear, J.: Hydraulics of groundwater. (McGraw-Hill series in water resources and environmental engineering). McGraw-Hill International Book Co., London (1979)

Bear, J., Sun, Y.: Optimization of pump-treat-inject (PTI) design for the remediation of a contaminated aquifer: multi-stage design with chance constraints. J. Contam. Hydrol. **29**(3), 225–244 (1998)

Bear, J., Cheng, A. H.-D.: Modeling Groundwater Flow and Contaminant Transport, Springer Heidelberg, Germany (2010)

Beck, J.V., Arnold, K.J.: Parameter estimation in engineering and science. (Wiley series in probability and mathematical statistics). Wiley, New York (1977)

Beck, M., Gupta, H., Rastetter, E., Shoemaker, C., Tarboton, D., Butler, R., Edelson, D., Graber, H., Gross, L., Harmon, T.: Grand challenges of the future for environmental modeling. White Paper. In: National Science Foundation, Arlington, Virginia, p. 153. (2009)

Belkin, M., Niyogi, P.: laplacian eigenmaps for dimensionality reduction and data representation. Neural. Comput. **15**(6), 1373–1396 (2003). doi:10.1162/089976603321780317

Bellman, R.: Dynamic programming. Princeton University Press, Princeton (1957)

Ben-Tal, A., Nemirovski, A.: Robust convex optimization. Math. Oper. Res. **23**(4), 769–805 (1998)

Berg, B.A.: Markov chain Monte Carlo simulations and their statistical analysis: with web-based Fortran code. World Scientific, Hackensack (2004)

Berger, M.J., Oliger, J.: Adaptive mesh refinement for hyperbolic partial differential equations. J. Comput. Phys. **53**(3), 484–512 (1984)

Bernardo, J.M., Smith, A.F.M.: Bayesian theory. Wiley, Chichester (1994)

Berre, I., Lien, M., Mannseth, T.: A level-set corrector to an adaptive multiscale permeability prediction. Comput. Geosci. **11**(1), 27–42 (2007)

Berre, I., Lien, M., Mannseth, T.: Multi-level parameter structure identification for two-phase porous-media flow problems using flexible representations. Adv. Water Resour. **32**(12), 1777–1788 (2009)

Berz, M., Bischof, C., Corliss, G., Griewank, A.: Computational differentiation: techniques, applications, and tools. (SIAM Proceedings in Applied Mathematics Series vol. 89). Society for Industrial and Applied Mathematics, Philadelphia (1996)

Besag, J., Green, P.J.: Spatial statistics and Bayesian computation. J. R. Stat Soc. Ser. B (Methodological), 25–37 (1993)

Besag, J., Green, P., Higdon, D., Mengersen, K.: Bayesian computation and stochastic systems. Statist. Sci. **10**(1), 3–41 (1995)

Beven, K.: Prophecy, reality and uncertainty in distributed hydrological modelling. Adv. Water Res. **16**(1), 41–51 (1993)

Beven, K.J.: Rainfall-runoff modelling: the primer. Wiley, Chichester (2001)

Beven, K.J.: Environmental modelling: an uncertain future?: an introduction to techniques for uncertainty estimation in environmental prediction. Routledge, London (2009)

Beven, K.J.: Rainfall-runoff modelling: the primer, 2nd edn. Wiley-Blackwell, Chichester (2012)

Beven, K., Binley, A.: The future of distributed models: model calibration and predictive uncertainty. Hydrol. Processes **6**, 279–298 (1992)

Beven, K., Binley, A.: GLUE: 20 years on. Hydrological processes, n/a-n/a (2013). doi:10.1002/hyp.10082

Beven, K., Freer, J.: Equifinality, data assimilation, and uncertainty estimation in mechanistic modelling of complex environmental systems using the GLUE methodology. J. Hydrol. **249**(1), 11–29 (2001)

Bhark, E.W., Jafarpour, B., Datta–Gupta, A.: A generalized grid connectivity-based parameterization for subsurface flow model calibration. Water. Resour. Res. **47**(6), W06517 (2011)

Bishop, C.M.: Pattern recognition and machine learning. Information science and statistics. Springer, New York (2006)

Bishop, C.H., Etherton, B.J., Majumdar, S.J.: Adaptive sampling with the ensemble transform Kalman filter. Part I: theoretical aspects. Mon. Weather Rev. **129**(3), 420–436 (2001)

Bivand, R.S., Pebesma, E.J., Rubio, V.G.: Applied spatial data analysis with R. Springer, New York (2008)

Blasone, R.-S., Madsen, H., Rosbjerg, D.: Uncertainty assessment of integrated distributed hydrological models using GLUE with Markov chain Monte Carlo sampling. J. Hydrol. **353**(1), 18–32 (2008)

Bogachev, V.I.: Measure theory. Springer, Berlin (2007)

Bolstad, W.M.: Understanding computational Bayesian statistics. (Wiley series in computational statistics). Wiley, Hoboken (2010)

Bottou, L., Lin, C.-J. (eds.): Support vector machine solvers. MIT Press, Cambridge (2007)

Bouveyron, C., Girard, S., Schmid, C.: High-dimensional data clustering. Comput. Stat. Data An. **52**(1), 502–519 (2007)

Bowden, G.J., Dandy, G.C., Maier, H.R.: Input determination for neural network models in water resources applications. Part 1—background and methodology. J. Hydrol. **301**(1), 75–92 (2005)

Box, G.E.P., Jenkins, G.M.: Time series analysis: forecasting and control. Holden-day series in time series analysis. Holden-Day, San Francisco (1970)

Boyacioglu, H., Boyacioglu, H., Gunduz, O.: Application of factor analysis in the assessment of surface water quality in Buyuk Menderes river basin. Eur. Water. **9**(10), 43–49 (2005)

Boyd, S.P., Vandenberghe, L.: Convex optimization. Cambridge University Press, Cambridge (2004)

Boyle, D.P., Gupta, H.V., Sorooshian, S.: Multicriteria calibration of hydrologic models. Water Sci. Appl. **6,** 185–196 (2003)

Branke, J.: Multiobjective optimization: interactive and evolutionary approaches. Lecture notes in computer science, vol. 5252. Springer, Berlin (2008)

Bratvold, R.B., Bickel, J.E., Lohne, H.P.: Value of information in the oil and gas industry: past, present, and future. Paper presented at the SPE Annual Technical Conference and Exhibition, Anaheim, California, November 11–14, (2007)

Bredehoeft, J.: The conceptualization model problem—surprise. Hydrogeol. J. **13**(1), 37–46 (2005)

Breiman, L.: The little bootstrap and other methods for dimensionality selection in regression: X-fixed prediction error. J Am Stat Asso **87**(419), 738–754 (1992)

Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)

Brezis, H.: Functional analysis, Sobolev spaces and partial differential equations. Springer, New York (2010)

Brochu, E., Cora, V.M., De Freitas, N.: A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599 (2010)

Brockwell, P.J., Davis, R.A.: Introduction to time series and forecasting, 2nd edn. (Springer texts in statistics). Springer, New York (2002)

Brooks, S.P., Roberts, G.O.: Convergence assessment techniques for Markov chain Monte Carlo. Statist. Comput. **8**(4), 319–335 (1998). doi:10.1023/a:1008820505350

Buhmann, M.D.: Radial basis functions: theory and implementations. Cambridge monographs on applied and computational mathematics, vol. 12. Cambridge University Press, Cambridge (2003)

Bui-Thanh, T., Willcox, K., Ghattas, O., van Bloemen Waanders, B.: Goal-oriented, model-constrained optimization for reduction of large-scale systems. J. Comput. Phys. **224**(2), 880–896 (2007)

Bungay, H.R.: Environmental systems engineering. Kluwer, Boston (1998)

Burg, J.P.: A new analysis technique for time series data. In: Childers, D.G. (ed.) Modern spectrum analysis, NATO advanced study institute of signal processing with emphasis on underwater Acoustics. IEEE Press, New York (1968)

Burger, M.: A framework for the construction of level set methods for shape optimization and reconstruction. Interf. Free Bound. **5**(3), 301–330 (2003)

Burger, M., Osher, S.J.: A survey on level set methods for inverse problems and optimal design. Eur. J. Appl. Math. **16**(02), 263–301 (2005)

Burgers, G., Jan van Leeuwen, P., Evensen, G.: Analysis scheme in the ensemble Kalman filter. Mon. Weather Rev. **126**(6), 1719–1724 (1998)

Burges, C.J.: A tutorial on support vector machines for pattern recognition. Data. Min. Knowl. Discov. **2**(2), 121–167 (1998)

Burkardt, J., Gunzburger, M., Lee, H.-C.: POD and CVT-based reduced-order modeling of Navier–Stokes flows. Comput. Methods Appl. Mech. Eng. **196**(1), 337–355 (2006)

Burnham, K.P., Anderson, D.R.: Model selection and multi-model inference: a practical information-theoretic approach. Springer, New York (2002)

Butts, M.B., Payne, J.T., Kristensen, M., Madsen, H.: An evaluation of the impact of model structure on hydrological modelling uncertainty for streamflow simulation. J. Hydrol. **298**(1), 242–266 (2004)

Buzzard, G.T.: Global sensitivity analysis using sparse grid interpolation and polynomial chaos. Reliab. Eng. Syst. Saf. **107,** 82–89 (2012)

Cacuci, D.G., Ionescu-Bujor, M., Navon, I.M.: Sensitivity and uncertainty analysis. Chapman & Hall, Boca Raton (2003)

Caers, J.: History matching under training-image-based geological model constraints. SPE. J. **8**(3), 218–226 (2003)

Caers, J.: Petroleum geostatistics. Society of Petroleum Engineers, Richardson (2005)

Caers, J., Zhang, T.: Multiple-point geostatistics: a quantitative vehicle for integration geologic analogs into multiple reservoir model. In: Grammer, G.M. (ed.) Integration of outcrop and modern analog data in reservoir models, vol. 80, pp. 384–394. American Association of Petroleum Geologists (2004) (AAPG Memoir)

Cannon, A.J., Whitfield, P.H.: Downscaling recent streamflow conditions in British Columbia, Canada using ensemble neural network models. J. Hydrol. **259**(1), 136–151 (2002)

Cao, G., Kyriakidis, P.C., Goodchild, M.F.: Combining spatial transition probabilities for stochastic simulation of categorical fields. Int. J. Geogr. Inform. Sci. **25**(11), 1773–1791 (2011)

Cappé, O., Godsill, S.J., Moulines, E.: An overview of existing methods and recent advances in sequential Monte Carlo. Proc. IEEE. **95**(5), 899–924 (2007)

Cardiff, M., Kitanidis, P.: Bayesian inversion for facies detection: an extensible level set framework. Water Resour. Res. **45**(10) (2009)

Cardoso, M., Durlofsky, L., Sarma, P.: Development and application of reduced–order modeling procedures for subsurface flow simulation. Int. J. Numer. Methods Eng. **77**(9), 1322–1350 (2009)

Carlberg, K., Farhat, C.: A low–cost, goal–oriented 'compact proper orthogonal decomposition' basis for model reduction of static systems. Int. J. Num. Methods Eng. **86**(3), 381–402 (2011)

Carrera, J.: State of the art of the inverse problem applied to the flow and solute transport equations. In: Groundwater flow and quality modelling. pp. 549–583. Springer (1988)

Carle, S.F., Fogg, G.E.: Transition probability-based indicator geostatistics. Math. Geol. **28**(4), 453–476 (1996)

Carle, S.F., Fogg, G.E.: Modeling spatial variability with one- and multi-dimensional continuous Markov chains. Math. Geol. **29**(7), 891–918 (1997)

Castaings, W., Dartus, D., Le Dimet, F.-X., Saulnier, G.-M.: Sensitivity analysis and parameter estimation for distributed hydrological modeling: potential of variational methods. Hydrol. Earth Syst. Sci. **13,** 503–517 (2009)

Catania, F., Paladino, O.: Optimal sampling for the estimation of dispersion parameters in soil columns using an iterative genetic algorithm. Environ. Model. Softw. **24**(1), 115–123 (2009)

Celia, M.A., Gray, W.G.: Numerical methods for differential equations: fundamental concepts for scientific and engineering applications. Prentice Hall, Englewood Cliffs (1992)

Chaloner, K., Verdinelli, I.: Bayesian experimental design: a review. Stat. Sci., **10**, 273–304 (1995)

Chan, T.F., Tai, X.-C.: Level set and total variation regularization for elliptic inverse problems with discontinuous coefficients. J. Comput. Phys. **193**(1), 40–66 (2004)

Chandrasekaran, S., Golub, G.H., Gu, M., Sayed, A.H.: Efficient algorithms for least squares type problems with bounded uncertainties. Recent. Adv. Total. Least. Sq. Tech. Errors-in-Var. Model. **93**, 171 (1997)

Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. ACM. Trans. Intel. Syst. Technol. (TIST). **2**(3), 27 (2011)

Chang, F.-J., Chiang, Y.-M., Chang, L.-C.: Multi-step-ahead neural networks for flood forecasting. Hydrol. Sci. J. **52**(1), 114–130 (2007). doi:10.1623/hysj.52.1.114

Chang, L.-C., Chu, H.-J., Hsiao, C.-T.: Optimal planning of a dynamic pump-treat-inject groundwater remediation system. J. Hydrol. **342**(3), 295–304 (2007)

Chang, H., Zhang, D., Lu, Z.: History matching of facies distribution with the EnKF and level set parameterization. J. Comput. Phys. **229**(20), 8011–8030 (2010)

Chavent, G.: Nonlinear least squares for inverse problems: theoretical foundations and step-by-step guide for applications. Springer, New York (2009)

Chavent, G., Bissell, R.: Indicator for the refinement of parametrization. In: Dulikravich, M.T.a.G.S. (ed.) Inverse problems in engineering mechanics (Proceedings of the international symposium on inverse problemsin engineering mechanics 1998 (ISIP'98), Nagano, Japan 1998, pp. 309–314. Elsevier, Amsterdam (1998)

Chavent, G., Dupuy, M., Lemonnier, P.: History matching by use of optimal theory. Soc. Petrol. Engg. J. **15**(1), 74–86 (1975)

Chen, Z.: Reservoir simulation: mathematical techniques in oil recovery. CBMS-NSF regional conference series in applied mathematics, vol. 77. SIAM/Society for Industrial and Applied Mathematics, Philadelphia (2007)

Chen, Y., Zhang, D.: Data assimilation for transient flow in geologic formations via ensemble Kalman filter. Adv. Water Resour. **29**(8), 1107–1122 (2006)

Chen, S., Cowan, C., Grant, P.: Orthogonal least squares learning algorithm for radial basis function networks. IEEE. T. Neural. Netw. **2**(2), 302–309 (1991)

Chen, J., Hubbard, S.S., Williams, K.H., Flores Orozco, A., Kemna, A.: Estimating the spatio-temporal distribution of geochemical parameters associated with biostimulation using spectral induced polarization data and hierarchical Bayesian models. Water Resour. Res. **48**, W05555 (2012)

Cheng, C.T., Ou, C.P., Chau, K.W.: Combining a fuzzy optimal model with a genetic algorithm to solve multi-objective rainfall-runoff model calibration. J. Hydrol. **268**(1–4), 72–86 (2002). doi:10.1016/s0022-1694(02)00122-1

Cherkassky, V., Ma, Y.: Practical selection of SVM parameters and noise estimation for SVM regression. Neural. Netw. **17**(1), 113–126 (2004)

Cherkassky, V.S., Mulier, F.: Learning from data: concepts, theory, and methods, 2nd edn. IEEE Press: Wiley-Interscience, Hoboken (2007)

Chib, S.: Marginal likelihood from the Gibbs output. J. Am. Stat. Asso. **90**(432), 1313–1321 (1995)

Chipman, H.A., George, E.I., McCulloch, R.E.: The practical implementation of Bayesian model selection. In: Lahiri, P. (ed.) Model selection. pp. 65–134. IMS, Beachwood (2001)

Chiu, Y.-C., Sun, N.-Z., Nishikawa, T., Yeh, W. W.-G.: Development of an objective-oriented groundwater model for conjunctive-use planning of surface water and groundwater. Water Resour. Res. **45**, W00B17 (2009). doi:10.1029/2007WR006662

Chou, I., Voit, E.O.: Recent developments in parameter estimation and structure identification of biochemical and genomic systems. Math. Biosci. **219**(2), 57–83 (2009)

Chow, V.T.: Open-channel hydraulics. McGraw-Hill civil engineering series. McGraw-Hill, New York (1959)

Chow, V.T., Maidment, D.R., Mays, L.W.: Applied hydrology. McGraw-Hill series in water resources and environmental engineering. McGraw-Hill, New York (1988)

Christian, W.D., Wilby, R.: An artificial neural network approach to rainfall-runoff modelling. Hydrol. Sci. J. **43**(1), 47–66 (1998)

Cirpka, O.A., Kitanidis, P.K.: Sensitivity of temporal moments calculated by the adjoint-state method and joint inversing of head and tracer data. Adv. Water Resour. **24**(1), 89–103 (2000)

Claeskens, G., Hjort, N.L.: Model selection and model averaging. Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, Cambridge (2008)

Clark, M.P., Rupp, D.E., Woods, R.A., Zheng, X., Ibbitt, R.P., Slater, A.G., Schmidt, J., Uddstrom, M.J.: Hydrological data assimilation with the ensemble Kalman filter: use of streamflow observations to update states in a distributed hydrological model. Adv. Water Resour. **31**(10), 1309–1324 (2008)

Clyde, M., George, E.I.: Model uncertainty. Stat. Sci. **19**(1), 81–94 (2004)

Coello Coello, C.A.: Evolutionary algorithms for solving multi-objective problems, 2nd edn. (Genetic and evolutionary computation series). Springer, New York (2007)

Coello Coello, C.A., Lamont, G.B., Van Veldhuisen, D.A.: Evolutionary algorithms for solving multi-objective problems. Springer, New York (2007)

Cohn, S.E.: An introduction to estimation theory. J. Meteorol. Soc. Jpn. **75,** 147–178 (1997)

Congdon, P.: Bayesian statistical modelling, 2nd edn. (Wiley series in probability and statistics). Wiley, Chichester (2006)

Conway, J.B.: A course in functional analysis. Graduate texts in mathematics, vol. 96, 2nd edn. Springer, New York (1997)

Coppola, E.A., Rana, A.J., Poulton, M.M., Szidarovszky, F., Uhl, V.W.: A neural network model for predicting aquifer water level elevations. Ground Water **43**(2), 231–241 (2005)

Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)

Costello, A.B., Osborne, J.W.: Best practices in exploratory factor analysis: four recommendations for getting the most from your analysis. Pract. Assess. Res. Eval. **10**, 1–9 (2005)

Coulibaly, P., Anctil, F., Bobée, B.: Daily reservoir inflow forecasting using artificial neural networks with stopped training approach. J. Hydrol. **230**(3–4), 244–257 (2000)

Coulibaly, P., Anctil, F., Aravena, R., Bobee, B.: Artificial neural network modeling of water table depth fluctuations. Water. Resour. Res. **37**(4), 885–896 (2001a)

Coulibaly, P., Bobee, B., Anctil, F.: Improving extreme hydrologic events forecasting using a new criterion for artificial neural network selection. Hydrol. Process. **15**(8), 1533–1536 (2001b)

Cowles, M.K., Carlin, B.P.: Markov chain Monte Carlo convergence diagnostics: a comparative review. J. Amer. Stat. Assoc. **91**(434), 883–904 (1996)

Cressie, N.A.C.: Statistics for spatial data, Rev. ed. Wiley series in probability and mathematical statistics. Applied probability and statistics. Wiley, New York (1993)

Cressie, N., Huang, H.-C.: Classes of nonseparable, spatio-temporal stationary covariance functions. J. Am. Stat. Assoc. **94**(448), 1330–1339 (1999)

Cybenko, G.: Approximation by superpositions of a sigmoidal function. Math. Control. Signal. **2**(4), 303–314 (1989)

Dagan, G., Cvetkovic, V., Shapiro, A.: A solute flux approach to transport in heterogeneous formations: 1. The general framework. Water Resour. Res. **28**(5), 1369–1376 (1992)

Davis, P.J., Rabinowitz, P.: Methods of numerical integration, 2nd edn. Dover Publications, Mineola (2007)

Dawson, C., Wilby, R.: Hydrological modelling using artificial neural networks. Prog. Phys. Geogr. **25**(1), 80–108 (2001)

De Berg, M.: Computational geometry: algorithms and applications, 2nd rev. edn. Springer, Berlin (2000)

de Marsily, G., Lavedan, C., Boucher, M., Fasanino, G.: Interpretation of interference tests in a well field using geostatistical techniques to fit the permeability distribution in a reservoir model. NATO ASI Series. In: Verly, G., David, M., Journel, A.G., Marechal, A. (eds.) Geostatistics for natural resources characterization, vol. 182, pp. 831–849. Springer, The Netherlands (1984)

Deb, K.: Multi-objective optimization using evolutionary algorithms, 1st edn. (Wiley-Interscience Series in Systems and Optimization). Wiley, Chichester (2001)

Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)

Debnath, L., Mikusiński, P.: Introduction to Hilbert spaces with applications, 2nd edn. Academic, San Diego (1999)

DeCezaro, A., Leitao, A., Tai, X.-C.: On multiple level-set regularization methods for inverse problems. Inverse Probl. **25**(3), 035004 (2009)

DeChant, C.M., Moradkhani, H.: Examining the effectiveness and robustness of sequential data assimilation methods for quantification of uncertainty in hydrologic forecasting. Water Resour. Res. **48**, W04518 (2012)

Dekking, M.: A modern introduction to probability and statistics: understanding why and how. (Springer texts in statistics). Springer, London (2005)

Del Moral, P., Doucet, A., Jasra, A.: Sequential Monte Carlo samplers. J. Roy. Stat. Soc. B. **68**(3), 411–436 (2006). doi:10.1111/j.1467-9868.2006.00553.x

Delay, F., Ackerer, P., Danquigny, C.: Simulating solute transport in porous or fractured formations using random walk particle tracking. Vadose Zone J. **4**(2), 360–379 (2005)

Delhomme, J.: Spatial variability and uncertainty in groundwater flow parameters: a geostatistical approach. Water Resour. Res. **15**(2), 269–280 (1979)

Demoment, G., Goussard, Y.: Inversion within the probabilistic framework. In: Bayesian approach to inverse problems, pp. 59–78. ISTE (2010). doi:10.1002/9780470611197.ch3

Deutsch, C.V., Journel, A.G.: GSLIB geostatistical software library and user's guide. Oxford University Press, New York (1998)

Dietrich, P., Leven, C.: Direct push-technologies. In Kirsch, R. (ed.): Groundwater geophysics. pp. 321–340. Berkin: Springer (2006)

Diggle, P., Ribeiro, P.J.: Model-based geostatistics. Springer series in statistics. Springer, New York (2007)

van Dijk, N.P., Maute, K., Langelaar, M., Van Keulen, F.: Level-set methods for structural topology optimization: a review. Struct. Multidiscipl. Optim. **48**(3), 437–472 (2013)

Ding, Y., Wang, S.S.: Optimal control of flood diversion in watershed using nonlinear optimization. Adv. Water Resour. **44,** 30–48 (2012)

Doherty, J.: Ground water model calibration using pilot points and regularization. Ground Water **41**(2), 170–177 (2003)

Doicu, A., Trautmann, T., Schreier, F.: Numerical regularization for atmospheric inverse problems. Springer-Praxis books in environmental sciences. Springer, Heidelberg (2010)

Dorn, O., Lesselier, D.: Level set methods for inverse scattering—some recent developments. Inverse Probl. **25**(12), 125001 (2009)

Dorn, O., Villegas, R.: History matching of petroleum reservoirs using a level set technique. Inverse Probl. **24**(3), 035015 (2008)

Douc, R., Cappé, O.: Comparison of resampling schemes for particle filtering. In: Image and signal processing and analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on 2005, pp. 64–69. IEEE (2005)

Doucet, A., Tadić, V.: Parameter estimation in general state-space models using particle methods. Ann. Inst. Stat. Math. **55**(2), 409–422 (2003). doi:10.1007/bf02530508

Doucet, A., Godsill, S., Andrieu, C.: On sequential Monte Carlo sampling methods for Bayesian filtering. Stat. Comput. **10**(3), 197–208 (2000). doi:10.1023/a:1008935410038

Doucet, A., De Freitas, N., Gordon, N.: Sequential Monte Carlo methods in practice, vol. 1. Springer, New York (2001a)

Doucet, A., Gordon, N.J., Krishnamurthy, V.: Particle filters for state estimation of jump Markov linear systems. IEEE Trans. Signal Process. **49,** 613–624 (2001b). doi:10.1109/78.905890

Doughty, C., Pruess, K.: Modeling supercritical carbon dioxide injection in heterogeneous porous media. Vadose. Zone. J. **3**(3), 837–847 (2004)

Draper, D.: Assessment and propagation of model uncertainty. J. R. Stat. Soc. Ser. B (Methodological), **57**(1), 45–97 (1995)

Dror, H.A., Steinberg, D.M.: Robust experimental design for multivariate generalized linear models. Technometrics **48**(4), 520–529 (2006)

Drucker, H.: Improving regressors using boosting techniques. In: Proceedings of the Fourteenth International Conference, pp. 107–115. Morgan Kaufmann Publishers Inc., Burlington (1997)

Duan, Q.: Global optimization for watershed model calibration. In: Duan, Q. (ed.) Calibration of watershed models. (AGU Water Sci. & App. 6). pp. 89–104. AGU, Washington, DC (2003)

Duan, Q., Sorooshian, S., Gupta, V.: Effective and efficient global optimization for conceptual rainfall-runoff models. Water Resour. Res. **28**(4), 1015–1031 (1992). doi:10.1029/91wr02985

Duan, Q., Ajami, N.K., Gao, X., Sorooshian, S.: Multi–model ensemble hydrologic prediction using Bayesian model averaging. Adv. Water Resour. **30**(5), 1371–1386 (2007)

Efendiev, Y., Hou, T.Y.: Multiscale finite element methods. Appl. Comput. Math. **217,** 50 (2009)

Efstratiadis, A., Koutsoyiannis, D.: One decade of multi-objective calibration approaches in hydrological modelling: a review. Hydrol. Sci. J. **55**(1), 58–78 (2010). doi:10.1080/02626660903526292

El Ghaoui, L., Lebret, H.: Robust solutions to least-squares problems with uncertain data. SIAM J. Matrix Anal. Appl. **18**(4), 1035–1064 (1997)

Eldred, M.: Recent advances in non-intrusive polynomial chaos and stochastic collocation methods for uncertainty analysis and design. In: 50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Palm Springs, California, 4–7 May (2009)

Eldred, M.S., Agarwal, H., Perez, V.M., Wojtkiewicz, S.F., Renaud, J.E.: Investigation of reliability method formulations in DAKOTA/UQ. Struct. Infrastruct. Eng. **3**(3), 199–213 (2007). doi:10.1080/15732470500254618

Elsheikh, A.H., Pain, C., Fang, F., Gomes, J., Navon, I.: Parameter estimation of subsurface flow models using iterative regularized ensemble Kalman filter. Stoch. Env. Res. Risk. A. 1–21 (2012). doi:10.1007/s00477-012-0613-x

Engl, H.W., Hanke, M., Neubauer, A.: Regularization of inverse problems. Mathematics and its applications, vol. 375. Kluwer, Dordrecht (1996)

Eppstein, M.J., Dougherty, D.E.: Simultaneous estimation of transmissivity values and zonation. Water Resour. Res. **32**(11), 3321–3336 (1996)

Evensen, G.: Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte-Carlo methods to do forecast error statistics. J. Geophys. Res. **99,** 10143–10162 (1994)

Evensen, G.: The ensemble Kalman filter: theoretical formulation and practical implementation. Ocean Dynam. **53**(4), 343–367 (2003)

Evensen, G.: Data assimilation: the ensemble Kalman filter. Springer, Berlin (2007)

Evensen, G.: The ensemble Kalman filter for combined state and parameter estimation. IEEE Contr. Syst. Mag. **29**(3), 83–104 (2009)

Fang, K., Li, R.-Z., Sudjianto, A.: Design and modeling for computer experiments. Computer science and data analysis series. Chapman & Hall/CRC, Boca Raton (2006)

Fang, F., Pain, C., Navon, I., Gorman, G., Piggott, M., Allison, P., Goddard, A.: A POD goal–oriented error measure for mesh optimization. Int. J. Numer. Methods Fluids **63**(2), 185–206 (2010)

Farmer, C.: Upscaling: a review. Int. J. Numer. Methods Fluids **40**(1–2), 63–78 (2002)

Fedorov, V.V., Hackl, P.: Model-oriented design of experiments. Lecture notes in statistics, vol. 125. Springer, New York (1997)

Ferreira, M.A.R., Lee, H.K.: Multiscale modeling: a Bayesian perspective. Springer (2007)

Feyen, L., Caers, J.: Quantifying geological uncertainty for flow and transport modeling in multimodal heterogeneous formations. Adv. Water Resour. **29**(6), 912–929 (2006)

Feyen, L., Vrugt, J.A., Nualláin, B.Ó., van der Knijff, J., De Roo, A.: Parameter optimisation and uncertainty assessment for large-scale streamflow simulation with the LISFLOOD model. J. Hydrol. **332**(3), 276–289 (2007)

Fienen, M., Hunt, R., Krabbenhoft, D., Clemo, T.: Obtaining parsimonious hydraulic conductivity fields using head and transport observations: a Bayesian geostatistical parameter estimation approach. Water Resour. Res. **45**(8), W08405 (2009)

Figueiredo, M.A.F., Jain, A.K.: Unsupervised learning of finite mixture models. IEEE T. Pattern Anal. **24**(3), 381–396 (2002)

Finkel, D., Kelley, C.: Additive scaling and the DIRECT algorithm. J. Glob Optim. **36**(4), 597–608 (2006). doi:10.1007/s10898-006-9029-9

Fleckenstein, J.H., Niswonger, R.G., Fogg, G.E.: River–aquifer interactions, geologic heterogeneity, and low–flow management. Ground Water **44**(6), 837–852 (2006)

Fletcher, R.: A new approach to variable metric algorithms. Comput. J. **13**(3), 317–322 (1970). doi:10.1093/comjnl/13.3.317

Fletcher, R.: Practical methods of optimization, 2nd edn. Wiley, Chichester (1987)

Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. Comput. J. **7**(2), 149–154 (1964)

Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In Forrest, S., (ed.): Proceedings of the Fifth Int. Conf. on Genetic Algorithms, pp. 416–423, San Mateo California, 1993

Foo, L.K., McGree, J., Eccleston, J., Duffull, S.: Comparison of robust criteria for D-optimal designs. J. Biopharm. Stat. **22**(6), 1193–1205 (2012)

Ford, I., Titterington, D., Kitsos, C.P.: Recent advances in nonlinear experimental design. Technometrics **31**(1), 49–60 (1989)

Fortune, S.: A sweepline algorithm for Voronoi diagrams. Algorithmica. **2**(1–4), 153–174 (1987)

Fraley, C., Raftery, A.E.: How many clusters? Which clustering method? Answers via model-based cluster analysis. Comput. J. **41**(8), 578–588 (1998). doi:10.1093/comjnl/41.8.578

Franceschini, G., Macchietto, S.: Model-based design of experiments for parameter precision: state of the art. Chem. Eng. Sci. **63**(19), 4846–4872 (2008)

Freeze, R.A.: A stochastic–conceptual analysis of one–dimensional groundwater flow in nonuniform homogeneous media. Water. Resour. Res. **11**(5), 725–741 (1975)

Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning, pp. 148–156. Morgan Kaufmann Publishers Inc., Burlington (1996)

Fu, J., Jaime Gómez-Hernández, J.: Uncertainty assessment and data worth in groundwater flow and mass transport modeling using a blocking Markov chain Monte Carlo method. J. Hydrol. **364**(3), 328–341 (2009)

Fu, J., Tchelepi, H.A., Caers, J.: A multiscale adjoint method to compute sensitivity coefficients for flow in heterogeneous porous media. Adv. Water Res. **33**(6), 698–709 (2010)

Ganapathysubramanian, B., Zabaras, N.: Sparse grid collocation schemes for stochastic natural convection problems. J. Comput. Phys. **225**(1), 652–685 (2007). doi:10.1016/j.jcp.2006.12.014

Gassman, P.W., Reyes, M.R., Green, C.H., Arnold, J.G.: The soil and water assessment tool: historical development, applications, and future research directions. Center for Agricultural and Rural Development, Iowa State University (2007)

Gauss, K.F.: Theoria combinationis observationum erroribus minimis obnoxiae. Werke. **4**, 1–108 (1826)

Gelas, A., Bernard, O., Friboulet, D., Prost, R.: Compactly supported radial basis functions based collocation method for level-set evolution in image segmentation. Image Process. IEEE Trans. **16**(7), 1873–1887 (2007)

Gelfand, A.E., Dey, D.K.: Bayesian model choice: asymptotics and exact calculations. J. R. Stat Soc. Ser. B **56**, (Methodological), 501–514 (1994)

Gelhar, L.W.: Stochastic subsurface hydrology from theory to applications. Water Resour. Res. **22**(9S), 135S–145S (1986)

Gelman, A., Rubin, D.B.: Inference from iterative simulation using multiple sequences. Statist. Sci. **7**(4), 457–472 (1992)

Geman, S., Geman, D.: Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. **6**(6), 721–741 (1984)

Gentle, J.E.: Matrix algebra: theory, computations, and applications in statistics. Springer texts in statistics. Springer, New York (2007)

Georgakakos, K.P., Seo, D.-J., Gupta, H., Schaake, J., Butts, M.B.: Towards the characterization of streamflow simulation uncertainty through multimodel ensembles. J Hydrol **298**(1), 222–241 (2004)

George, E.I., McCulloch, R.E.: Variable selection via Gibbs sampling. J. Am. Stat. Assoc. **88**(423), 881–889 (1993)

Gerritsen, M.G., Durlofsky, L.J.: Modeling fluid flow in oil reservoirs. Annu. Rev. Fluid Mech. **37**, 211–238 (2005)

Geweke, J.: Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In: Bernardo, J.M., Berger, J., Dawid, A.P., Smith, A.F.M. (eds.) Bayesian statistics 4, pp. 169–193. Oxford University Press, Oxford (1992)

Ghanem, R., Spanos, P.D.: Stochastic finite elements: a spectral approach. Springer, ew York (1991)

Ghosh, S., Mujumdar, P.: Statistical downscaling of GCM simulations to streamflow using relevance vector machine. Adv. Water Resour. **31**(1), 132–146 (2008)

Gilks, W.R., Richardson, S., Spiegelhalter, D.J.: Markov chain Monte Carlo in practice. Chapman & Hall, Boca Raton (1998)

Gneiting, T.: Nonseparable, stationary covariance functions for space–time data. J. Am. Stat. Assoc. **97**(458), 590–600 (2002)

Gneiting, T., Genton, M., Guttorp, P.: Geostatistical space-time models, stationarity, separability and full symmetry. Stat. Methods Spatio-Temporal Syst. **107,** 151–175 (2007)

Godsill, S.J.: On the relationship between Markov chain Monte Carlo methods for model uncertainty. J. Comput. Gr. Stat. **10**(2), 230–248 (2001)

Goldberg, D.E.: Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Boston (1989)

Golub, G.H., Van Loan, C.F.: Matrix computations, 3rd edn. (Johns Hopkins studies in the mathematical sciences). Johns Hopkins University Press, Baltimore (1996)

Golub, G.H., Heath, M., Wahba, G.: Generalized cross-validation as a method for choosing a good ridge parameter. Technometrics **21**(2), 215–223 (1979)

Gómez-Hernánez, J.J., Sahuquillo, A., Capilla, J.E.: Stochastic simulation of transmissivity fields conditional to both transmissivity and piezometric data-1. Theory J. Hydrol. (Amsterdam) **203**(1), 167–174 (1997)

Goovaerts, P.: Geostatistics for natural resources evaluation. Applied geostatistics series. Oxford University Press, New York (1997)

Goovaerts, P.: Geostatistical approaches for incorporating elevation into the spatial interpolation of rainfall. J. Hydrol. **228**(1), 113–129 (2000)

Gordon, N., Salmond, D., Smith, A.F.: Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proc. F. **140,** 107–113 (1993)

Gorelick, S.M., Evans, B., Remson, I.: Identifying sources of groundwater pollution: an optimization approach. Water Resour. Res. **19**(3), 779–790 (1983)

Gove, J.H., Hollinger, D.Y.: Application of a dual unscented Kalman filter for simultaneous state and parameter estimation in problems of surface-atmosphere exchange. J. Geophys. Res. **111**(D8), D08S07 (2006). doi:10.1029/2005jd006021

Graham, W., McLaughlin, D.: Stochastic analysis of nonstationary subsurface solute transport: 2. Conditional moments. Water Resour. Res. **25**(11), 2331–2355 (1989). doi:10.1029/WR025i011p02331

Gray, R.M.: Entropy and information theory. Springer, New York. http://dx.doi.org/10.1007/978-1-4419-7970-4 (2011)

Green, P.J.: Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika **82**(4), 711–732 (1995)

Griewank, A., Walther, A.: Evaluating derivatives: principles and techniques of algorithmic differentiation, 2nd edn. Society for Applied and Industrial Mathematics, Philadelphia (2008)

Griffel, D.H.: Applied functional analysis, Dover ed.. Dover (2002)

Grimstad, A.-A., Mannseth, T., Nævdal, G., Urkedal, H.: Adaptive multiscale permeability estimation. Comput. Geosci. **7**(1), 1–25 (2003)

Gu, Y., Oliver, D.: History matching of the PUNQ-S3 reservoir model using the ensemble Kalman filter. SPE J. **10**(2), 217–224 (2005)

Gupta, H.V., Beven, K.J., Wagener, T.: Model calibration and uncertainty estimation. In: Anderson, M.G., McDonnell, J.J. (eds.): Encyclopedia of hydrological sciences, Wiley, Chichester, pp. 1–17 (2005)

Gupta, H.V., Clark, M.P., Vrugt, J.A., Abramowitz, G., Ye, M.: Towards a comprehensive assessment of model structural adequacy. Water Resour. Res. **48**(8), W08301 (2012)

Haario, H., Laine, M., Mira, A., Saksman, E.: DRAM: efficient adaptive MCMC. Statist. Comput. **16**(4), 339–354 (2006). doi:10.1007/s11222-006-9438-0

Hadamard, J.: Sur les problèmes aux dérivées partielles et leur signification physique. Princet. Univers. Bull. **13**(49–52), 28 (1902)

Haddad, O.B., Afshar, A., Mariño, M.A.: Honey-bees mating optimization (HBMO) algorithm: a new heuristic approach for water resources optimization. Water Resour. Manag. **20**(5), 661–680 (2006)

Hadka, D., Reed, P.: Diagnostic assessment of search controls and failure modes in many-objective evolutionary optimization. Evol. Comput. **20**(3), 423–452 (2012)

Hain, C.R., Crow, W.T., Anderson, M.C., Mecikalski, J.R.: An ensemble Kalman filter dual assimilation of thermal infrared and microwave satellite observations of soil moisture into the Noah land surface model. Water Resour. Res. **48**, W11517 (2012)

Hakimi, S.: Optimum location of switching centers and the absolute centers and medians of a graph. Oper. Res. (12), 450–459 (1964)

Hamill, T.M., Whitaker, J.S., Snyder, C.: Distance-dependent filtering of background error covariance estimates in an ensemble Kalman filter. Mon. Weather Rev. **129**(11), 2776–2790 (2001)

Hammami, D., Lee, T.S., Ouarda, T.B.M.J., Lee, J.: Predictor selection for downscaling GCM data with LASSO. J. Geophys. Res. Atmos. **117**(D17), D17116 (2012). doi:10.1029/2012jd017864

Han, S.-P.: Penalty Lagrangian methods via a Quasi-Newton approach. Math.Oper. Res. **4**(3), 291–302 (1979)

Hansen, P.C.: Discrete inverse problems: insight and algorithms. Fundamentals of algorithms. Society for Industrial and Applied Mathematics, Philadelphia (2010)

Hansen, L., Salamon, P.: Neural network ensemblese. IEEE Trans. Pattern. Anal. Mach. Intell. **12**(10), 993–1001 (1990)

Härdle, W., Simar, L.: Applied multivariate statistical analysis, 3rd edn. Springer, Heidelberg (2012)

Hartikainen, J., Särkkä, S.: Optimal filtering with Kalman filters and smoothers—a manual for Matlab toolbox EKF/UKF. In: Department of Biomedical Engineering and Computational Science, Helsinki University of Technology, Finland (2008)

Hartley, R.V.: Transmission of information1. Bell. Sys. Tech. J. **7**(3), 535–563 (1928)

Harvey, C.F., Gorelick, S.M.: Mapping hydraulic conductivity: sequential conditioning with measurements of solute arrival time, hydraulic head, and local conductivity. Water Resour. Res. **31**(7), 1615–1626 (1995)

Hastie, T., Tibshirani, R., Friedman, J.H.: The elements of statistical learning: data mining, inference, and prediction, 2nd edn. (Springer series in statistics). Springer, New York (2009)

Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika. **57**, 97–109 (1970)

Hayek, M., Lehmann, F., Ackerer, P.: Adaptive multi-scale parameterization for one-dimensional flow in unsaturated porous media. Adv. Water Resour. **31**(1), 28–43 (2008)

Hayek, M., Ackerer, P., Sonnendrücker, É.: A new refinement indicator for adaptive parameterization: application to the estimation of the diffusion coefficient in an elliptic problem. J. Comput. Appl. Math. **224**(1), 307–319 (2009)

Haykin, S.S.: Neural networks: a comprehensive foundation. Macmillan, New York (1994)

He, S., Carmichael, G.R., Sandu, A., Hotchkiss, B., Damian-Iordache, V.: Application of ADIFOR for air pollution model sensitivity studies. Environ. Model. Softw. **15**(6), 549–557 (2000)

He, L., Huang, G., Lu, H.: A simulation-based fuzzy chance-constrained programming model for optimal groundwater remediation under uncertainty. Adv. Water Resour. **31**(12), 1622–1635 (2008)

Heer, J., Agrawala, M.: Design considerations for collaborative visual analytics. Inf. Vis. **7**(1), 49–62 (2008)

Helmig, R.: Multiphase flow and transport processes in the subsurface: a contribution to the modeling of hydrosystems. Environmental engineering. Springer, Berlin (1997)

Helton, J.C., Davis, F.J.: Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. Reliab. Eng. & amp; Sys. Saf. **81**(1), 23–69 (2003). doi:10.1016/s0951-8320(03)00058-9

Hendricks Franssen, H., Kinzelbach, W.: Real–time groundwater flow modeling with the ensemble Kalman filter: joint estimation of states and parameters and the filter inbreeding problem. Water Resour. Res. **44**, W09408 (2008)

Hendricks Franssen, H., Alcolea, A., Riva, M., Bakr, M., Van Der Wiel, N., Stauffer, F., Guadagnini, A.: A comparison of seven methods for the inverse modelling of groundwater flow. Application to the characterisation of well catchments. Adv. Water Resour. **32**(6), 851–872 (2009)

Hendrix, E.M., Boglárka, G.: Introduction to nonlinear and global optimization, vol. 37. Springer, New York (2010)

Hengl, T., Heuvelink, G., Stein, A.: A generic framework for spatial prediction of soil variables based on regression-kriging. Geoderma. **120**(1), 75–93 (2004)

Hill, M.C., Tiedeman, C.R.: Effective groundwater model calibration: with analysis of data, sensitivities, predictions, and uncertainty. Wiley-Interscience, Hoboken (2007)

Hirsch, F., Lacombe, G.: Elements of functional analysis. Graduate texts in mathematics, vol. 192. Springer, New York (1999)

Hjalmarsson, H.: System identification of complex and structured systems. Eur. J. Control **15**(3), 275–310 (2009)

Hoeksema, R.J., Kitanidis, P.K.: An application of the geostatistical approach to the inverse problem in two–dimensional groundwater modeling. Water Resour. Res. **20**(7), 1003–1020 (1984)

Hoeksema, R.J., Kitanidis, P.K.: Analysis of the spatial structure of properties of selected aquifers. Water. Resour. Res. **21**(4), 563–572 (1985)

Hoerl, A.E., Kennard, R.W.: Ridge regression: biased estimation for nonorthogonal problems. Technometrics **12**(1), 55–67 (1970)

Hoeting, J.A., Madigan, D., Raftery, A.E., Volinsky, C.T.: Bayesian model averaging: a tutorial. Stat. Sci. **14**(4), 382–401 (1999)

Hoffman, J.D.: Numerical methods for engineers and scientists. 2nd edn. Marcel Dekker, New York (2001)

Hol, J.D., Schon, T.B., Gustafsson, F.: On resampling algorithms for particle filters. In: Nonlinear Statistical Signal Processing Workshop, 2006 IEEE, 13–15 September 2006, pp. 79–82. (2006)

Holland, J.H.: Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor (1975)

Horn, J., Nafpliotis, N.: Multiobjective optimization using the niched pareto genetic algorithm. Technical report 930005. In. Illinois Genetic Algorithms Laboratory (IlliGAL). University of Illinois, Urbana (1993)

Hosder, S., Walters, R.W., Balch, M.: Efficient sampling for non-intrusive polynomial chaos applications with multiple uncertain input variables. In: 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Honolulu, HI, April 23–26 (2007)

Hotelling, H.: Analysis of a complex of statistical variables into principal components. J. Educ. Psychol. **24**, 417–441 (1933)

Hou, Z., Rubin, Y.: On minimum relative entropy concepts and prior compatibility issues in vadose zone inverse and forward modeling. Water Resour. Res. **41**, W12425 (2005)

Houtekamer, P.L., Mitchell, H.L.: A sequential ensemble Kalman filter for atmospheric data assimilation. Mon. Weather Rev. **129**(1), 123–137 (2001)

Hsu, N.S., Yeh, W.W.G.: Optimum experimental design for parameter identification in groundwater hydrology. Water Resour. Res. **25**(5), 1025–1040 (1989)

Hsu, K., Gupta, H.V., Sorooshian, S.: Artificial neural network modeling of the rainfall-runoff process. Water Resour. Res. **31**(10), 2517–2530 (1995)

Huan, X., Marzouk, Y.M.: Simulation-based optimal Bayesian experimental design for nonlinear systems. J. Comput. Phys. **232**(1), 288–317 (2013). doi:http://dx.doi.org/10.1016/j.jcp.2012.08.013

Huang, C., Mayer, A.S.: Pump–and–treat optimization using well locations and pumping rates as decision variables. Water Resour. Res. **33**(5), 1001–1012 (1997)

Hurvich, C.M., Tsai, C.-L.: Regression and time series model selection in small samples. Biometrika **76**(2), 297–307 (1989)

Huyer, W., Neumaier, A.: Global optimization by multilevel coordinate search. J. Glob. Optim. **14**(4), 331–355 (1999). doi:10.1023/a:1008382309369

Iglesias, M.A., McLaughlin, D.: Level-set techniques for facies identification in reservoir modeling. Inverse Probl. **27**(3), 035008 (2011)

Iman, R.L., Shortencarier, M.J.: A Fortran 77 Program and User's Guide for the Generation of Latin Hypercube Samples for Use with Computer Models. NUREG/CR-3624, Technical Report SAND83-2365. In. Sandia National Laboratories, Albuquerque (1984)

Irsa, J., Zhang, Y.: A direct method of parameter estimation for steady state flow in heterogeneous aquifers with unknown boundary conditions. Water Resour. Res. **48**, W09526 (2012)

Isaaks, E.H., Srivastava, R.M.: Applied geostatistics. Oxford University Press, New York (1989)
Jafarpour, B., McLaughlin, D.B.: History matching with an ensemble Kalman filter and discrete cosine parameterization. Comput. Geosci. **12**(2), 227–244 (2008)
Janssen, H.: Monte-Carlo based uncertainty analysis: sampling efficiency and sampling convergence. Reliab. Eng. Sys. Saf. **109**, 123–132 (2013)
Janssen, G.M., Valstar, J.R., van der Zee, S.E.: Measurement network design including traveltime determinations to minimize model prediction uncertainty. Water Resour. Res. **44**(2), W02405 (2008)
Jaynes, E.T.: On the rationale of maximum-entropy methods. Proc. IEEE **70**(9), 939–952 (1982)
Jaynes, E.T., Bretthorst, G.L.: Probability theory: the logic of science. Cambridge University Press, Cambridge (2003)
Jha, M., Datta, B.: Three-dimensional groundwater contamination source identification using adaptive simulated annealing. J. Hydrol. Eng. **18**(3), 307–317 (2012)
Jha, S.K., Mariethoz, G., Evans, J.P., McCabe, M.F.: Demonstration of a geostatistical approach to physically consistent downscaling of climate modeling simulations. Water Resour. Res. **49,** 245–259 (2013)
Jhurani, C., Demkowicz, L.: Multiscale modeling using goal-oriented adaptivity and numerical homogenization. Part I: mathematical formulation and numerical results. Computer methods in applied mechanics and engineering 213, 399–417 (2012)
Jolliffe, I.T.: Principal component analysis, 2nd edn. (Springer series in statistics). Springer, New York (2002)
Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. J. Optim. Theory Appl. **79**(1), 157–181 (1993). doi:10.1007/bf00941892
Journel, A.G.: Geostatistics: roadblocks and challenges. In: Soares, A. (ed.) Geostatistics-Troia, vol. 1, pp. 213–224. Kluwer, Dordrecht (1992)
Journel, A.G., Huijbregts, C.J.: Mining geostatistics. Academic, London (1978)
Journel, A., Isaaks, E.: Conditional indicator simulation: application to a Saskatchewan uranium deposit. J. Int. Assoc. Math. Geol. **16**(7), 685–718 (1984)
Ju, Q., Yu, Z., Hao, Z., Ou, G., Zhao, J., Liu, D.: Division-based rainfall-runoff simulations with BP neural networks and Xinanjiang model. Neurocomputing **72**(13–15), 2873–2883 (2009). doi:10.1016/j.neucom.2008.12.032
Julier, S.J., Uhlmann, J.K.: Unscented filtering and nonlinear estimation. Proc. IEEE **92**(3), 401–422 (2004)
Julier, S.J., Uhlmann, J.K., Durrant-Whyte, H.F.: A new approach for filtering nonlinear systems. In: American Control Conference, 1995. Proceedings of the, 21–23 June 1995, vol. 1623, pp. 1628–1632. (1995)
Julier, S.J., Uhlmann, J., Durrant-Whyte, H.F.: A new method for the nonlinear transformation of means and covariances in filters and estimators. IEEE Trans. Automat. Contr. **45**(3), 477–482 (2000)
Jung, Y., Imhoff, P., Finsterle, S.: Estimation of landfill gas generation rate and gas permeability field of refuse using inverse modeling. Transp. Porous Media **90**(1), 41–58 (2011a)
Jung, Y., Ranjithan, R.S., Mahinthakumar, G.: Subsurface characterization using a D-optimality based pilot point method. J. Hydroinform. **13**(4), 775–793 (2011b)
Kaiser, H.F.: The varimax criterion for analytic rotation in factor analysis. Psychometrika **23,** 187–200 (1958)
Kaleta, M.P., Hanea, R.G., Heemink, A.W., Jansen, J.-D.: Model-reduced gradient-based history matching. Comput. Geosci. **15**(1), 135–153 (2011)
Kalman, R.E.: A new approach to linear filtering and prediction problems. J. Basic Eng. T. ASME. **82,** 35–45 (1960)
Kalnay, E.: Atmospheric modeling, data assimilation, and predictability. Cambridge University Press, New York (2003)
Kalra, A., Ahmad, S.: Using oceanic-atmospheric oscillations for long lead time streamflow forecasting. Water Resour. Res. **45**(3), W03413 (2009). doi:10.1029/2008wr006855

Kannan, N., Jeong, J., Srinivasan, R.: Hydrologic modeling of a canal-irrigated agricultural watershed with irrigation best management practices: case study. J. Hydrol. Eng. **16**(9), 746–757 (2010)

Katkovnik, V., Egiazarian, K., Astola, J.: Local approximation techniques in signal and image processing. SPIE Press Book, Bellingham (2006)

Kégl, B.: Intrinsic dimension estimation using packing numbers. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in neural information processing systems, vol. 15. MIT Press, Cambridge (2003)

Keim, D., Andrienko, G., Fekete, J.D., Görg, C., Kohlhammer, J., Melançon, G.: Visual analytics: definition, process, and challenges. Inf Vis. -Human-Centered Issues and Perspectives, Springer, pp. 154–175 (2008)

Kennedy, M.C., O'Hagan, A.: Bayesian calibration of computer models. J. R. Stat. Soc.: Ser. B (Statistical Methodology) **63**(3), 425–464 (2001)

Kim, S., Eyink, G.L., Restrepo, J.M., Alexander, F.J., Johnson, G.: Ensemble filtering for nonlinear dynamics. Mon. Weather Rev. **131**(11), 2586–2594 (2003)

Kitanidis, P.K.: Generalized covariance functions in estimation. Math. Geol. **25**(5), 525–540 (1993)

Kitanidis, P.K.: Quasi–linear geostatistical theory for inversing. Water Resour. Res. **31**(10), 2411–2419 (1995)

Kitanidis, P.K.: Introduction to geostatistics: applications to hydrogeology. Cambridge University Press, Cambridge (1997)

Kitanidis, P.K.: Generalized priors in Bayesian inversion problems. Adv. Water Resour. **36,** 3–10 (2012)

Kitanidis, P.K., Bras, R.L.: Real-time forecasting with a conceptual hydrologic model: 2. Applications and results. Water Resour. Res. **16**(6), 1034–1044 (1980)

Klimke, A., Wohlmuth, B.: Algorithm 847: Spinterp: piecewise multilinear hierarchical sparse grid interpolation in MATLAB. ACM Trans. Math. Softw. **31**(4), 561–579 (2005). doi:10.1145/1114268.1114275

Knopman, D.S., Voss, C.I.: Multiobjective sampling design for parameter estimation and model discrimination in groundwater solute transport. Water Resour. Res. **25**(10), 2245–2258 (1989)

Knopman, D.S., Voss, C.I., Garabedian, S.P.: Sampling design for groundwater solute transport: tests of methods and analysis of Cape Cod tracer test data. Water Resour. Res. **27**(5), 925–949 (1991)

Knotters, M., Heuvelink, G.B.M., Hoogland, T., Walvoort, D.J.J.: A disposition of interpolation techniques, WOt-werkdocument 190. In: Statutory research tasks unit for nature and the environment, Wageningen UR, Wageningen, Netherlands, p. 90 (2010)

Knowles, J.D., Corne, D.W.: Approximating the nondominated front using the Pareto archived evolution strategy. Evol. Comput. **8**(2), 149–172 (2000)

Knowles, J., Corne, D., Deb, K.: Multiobjective problem solving from nature: from concepts to applications. Natural computing series. Springer, Berlin (2008)

Kolditz, O.: Computational methods in environmental fluid mechanics. Springer, Berlin (2002)

Kollat, J.B., Reed, P.M.: Comparing state-of-the-art evolutionary multi-objective algorithms for long-term groundwater monitoring design. Adv. Water Res. **29**(6), 792–807 (2006)

Kollat, J.B., Reed, P.M., Maxwell, R.: Many-objective groundwater monitoring network design using bias-aware ensemble Kalman filtering, evolutionary optimization, and visual analytics. Water Resour. Res. **47**, W02529 (2011)

Koltermann, C.E., Gorelick, S.M.: Heterogeneity in sedimentary deposits: a review of structure-imitating, process-imitating, and descriptive approaches. Water Resour. Res. **32**(9), 2617–2658 (1996)

Körkel, S., Kostina, E., Bock, H.G., Schlöder, J.P.: Numerical methods for optimal control problems in design of robust optimal experiments for nonlinear dynamic processes. Optim. Methods Softw. **19**(3–4), 327–338 (2004)

Kotecha, J.H., Djuric, P.M.: Gaussian sum particle filtering. IEEE T. Signal Process. **51**(10), 2602–2612 (2003)

Koutsourelakis, P.S.: A multi-resolution, non-parametric, Bayesian framework for identification of spatially-varying model parameters. J. Comput. Phys. **228**(17), 6184–6211 (2009). doi:http://dx.doi.org/10.1016/j.jcp.2009.05.016

Kowalsky, M.B., Finsterle, S., Rubin, Y.: Estimating flow parameter distributions using ground-penetrating radar and hydrological measurements during transient flow in the vadose zone. Adv. Water Resour. **27**(6), 583–599 (2004)

Krajewski, W.F.: Cokriging radar–rainfall and rain gage data. J. Geophys. Res. Atmos. **92**(D8), 9571–9580 (1987)

Krause, A., Guestrin, C.: Near-optimal observation selection using submodular functions. In: AAAI 2007, pp. 1650–1654

Krause, A., Singh, A., Guestrin, C.: Near-optimal sensor placements in Gaussian processes: theory, efficient algorithms and empirical studies. J. Mach. Learn. Res. **9**, 235–284 (2008)

Kullback, S.: Information theory and statistics. (Dover books on mathematics). Dover, Mineola (1997)

Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Statist. **22**(1), 79–86 (1951)

Kyriakidis, P., Journel, A.: Geostatistical space–time models: a review. Math. Geol. **31**(6), 651–684 (1999). doi:10.1023/a:1007528426688

Kyung, M., Gill, J., Ghosh, M., Casella, G.: Penalized regression, standard errors, and Bayesian lassos. Bayesian. Anal. **5**(2), 369–411 (2010)

Lapidus, L., Pinder, G.F.: Numerical solution of partial differential equations in science and engineering. Wiley, New York (1982)

Laub, A.J.: Matrix analysis for scientists and engineers. Society for Industrial and Applied Mathematics, Philadelphia (2005)

LaVenue, A.M., Pickens, J.F.: Application of a coupled adjoint sensitivity and kriging approach to calibrate a groundwater flow model. Water Resour. Res. **28**(6), 1543–1569 (1992)

LaVenue, A.M., RamaRao, B.S., De Marsily, G., Marietta, M.G.: Pilot point methodology for automated calibration of an ensemble of conditionally simulated transmissivity fields: 2. Application. Water Resour. Res. **31**(3), 495–516 (1995)

Le Ravalec-Dupin, M.: Pilot block method methodology to calibrate stochastic permeability fields to dynamic data. Math. Geosci. **42**(2), 165–185 (2010)

Leamer, E.E.: Specification searches: ad hoc inference with nonexperimental data. Wiley New York, (1978)

Lee, A.W.F., Sweldens, W., Schröder, P., Cowsar, L., Dobkin, D.: MAPS: multiresolution adaptive parameterization of surfaces. Paper presented at the Proceedings of the 25th annual conference on Computer graphics and interactive techniques (1998)

Lee, H.K.H., Higdon, D.M., Bi, Z., Ferreira, M.A.R., West, M.: Markov random field models for high-dimensional parameters in simulations of fluid flow in porous media. Technometrics **44**(3), 230–241 (2002)

Lehmann, E.L., Casella, G.: Theory of point estimation, 2nd edn. (Springer texts in statistics). Springer, New York (1998)

Lehoucq, R.B., Sorensen, D.C., Yang, C.: ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods. SIAM, Philadelphia (1998) (Software, environments, tools.)

Lempert, R.J., Groves, D.G., Popper, S.W., Bankes, S.C.: A general, analytic method for generating robust strategies and narrative scenarios. Manag. Sci. **52**(4), 514–528 (2006)

Leube, P., Geiges, A., Nowak, W.: Bayesian assessment of the expected data impact on prediction confidence in optimal sampling design. Water Resour. Res. **48**, W02501 (2012)

Leven, C., Dietrich, P.: What information can we get from pumping tests?-comparing pumping test configurations using sensitivity coefficients. J Hydrol **319**(1), 199–215 (2006)

Levina, E., Bickel, P.J.: Maximum likelihood estimation of intrinsic dimension. Adv. Neural. Inf. Process. Syst. **18**, 777–784 (2004)

Lewis, J.M., Lakshmivarahan, S., Dhall, S.K.: Dynamic data assimilation: a least squares approach. Encyclopedia of mathematics and its applications 104. Cambridge University Press, Cambridge (2006)

Li, X., Tsai, F.T.-C.: Bayesian model averaging for groundwater head prediction and uncertainty analysis using multimodel and multimethod. Water Resour. Res. **45**, W09403 (2009)

Li, J., Xiu, D.: A generalized polynomial chaos based ensemble Kalman filter with high accuracy. J. Comput. Phys. **228**(15), 5454–5469 (2009)

Li, H., Zhang, D.: Probabilistic collocation method for flow in porous media: comparisons with other stochastic methods. Water Resour. Res. **43**(9), W09409 (2007). doi:10.1029/2006wr005673

Li, J., Zou, J.: A multilevel model correction method for parameter identification. Inverse Probl. **23**(5), 1759 (2007)

Li, W., Nowak, W., Cirpka, O.A.: Geostatistical inverse modeling of transient pumping tests using temporal moments of drawdown. Water Resour. Res. **41**(8), W08403 (2005)

Li, J., Li, J., Xiu, D.: An efficient surrogate-based method for computing rare failure probability. J. Comput. Phys. **230**(24), 8683–8697 (2011)

Liao, Q., Zhang, D.: Probabilistic collocation method for strongly nonlinear problems: 1. Transform by location. Water Resour. Res. **49**(12), 7911–7928 (2013)

Lien, M., Mannseth, T.: Facies estimation through data assimilation and structure parameterization. Comput. Geosci. **18**(5), 869–882 (2014)

Lien, M., Berre, I., Mannseth, T.: Combined adaptive multiscale and level-set parameter estimation. Multiscale Model. Simul. **4**(4), 1349–1372 (2005)

Lin, G.-F., Wu, M.-C.: An RBF network with a two-step learning algorithm for developing a reservoir inflow forecasting model. J. Hydrol. **405**(3–4), 439–450 (2011). doi:http://dx.doi.org/10.1016/j.jhydrol.2011.05.042

Lin, H.-T., Tan, Y.-C., Chen, C.-H., Yu, H.-L., Wu, S.-C., Ke, K.-Y.: Estimation of effective hydrogeological parameters in heterogeneous and anisotropic aquifers. J. Hydrol. **389**(1), 57–68 (2010)

Lindley, D.V.: Bayesian statistics: a review. SIAM, Philadelphia (1972)

Liu, J.: A multiresolution method for distributed parameter estimation. SIAM J. Sci. Comput. **14**(2), 389–405 (1993)

Liu, J.S.: Metropolized independent sampling with comparisons to rejection sampling and importance sampling. Stat. Comput. **6**(2), 113–119 (1996)

Liu, J.S.: Monte Carlo strategies in scientific computing. (Springer series in statistics). Springer, New York (2001)

Liu, J.S., Chen, R.: Sequential Monte Carlo methods for dynamic systems. J. Am. Stat. Assoc. **93**(443), 1032–1044 (1998)

Liu, Y., Gupta, H.V.: Uncertainty in hydrologic modeling: toward an integrated data assimilation framework. Water Resour. Res. **43**(7), W07401 (2007). doi:10.1029/2006wr005756

Liu, M., Liu, G.: Smoothed particle hydrodynamics (SPH): an overview and recent developments. Arch. Comput. Methods Eng. **17**(1), 25–76 (2010)

Liu, J., West, M.: Combined parameter and state estimation in simulation-based filtering. In: Doucet, A., Gordon, N. (eds.) Sequential Monte Carlo methods in practice. Springer, New York (2001)

Liu, C.-W., Lin, K.-H., Kuo, Y.-M.: Application of factor analysis in the assessment of groundwater quality in a blackfoot disease area in Taiwan. Sci. Total. Environ. **313**(1–3), 77–89 (2003a). doi:10.1016/s0048-9697(02)00683-6

Liu, J., Savenije, H.H.G., Xu, J.: Forecast of water demand in Weinan City in China using WDF-ANN model. Phys. Chem. Earth. Parts A/B/C. **28**(4–5), 219–224 (2003b). doi:http://dx.doi.org/10.1016/S1474-7065(03)00026-3

Liu, X., Illman, W., Craig, A., Zhu, J., Yeh, T.-C.J.: Laboratory sandbox validation of transient hydraulic tomography. Water Resour. Res. **43**(5), W05430 (2007)

Liu, X., Cardiff, M.A., Kitanidis, P.K.: Parameter estimation in nonlinear environmental problems. Stoch. Env. Res. Risk A. **24**(7), 1003–1022 (2010)

Liu, Y., Sun, A.Y., Nelson, K., Hipke, W.E.: Cloud computing for integrated stochastic groundwater uncertainty analysis. Int. J. Digit. Earth 1–25 (2012a). doi:10.1080/17538947.2012.687778

Liu, Y., Weerts, A.H., Clark, M., Hendricks Franssen, H.J., Kumar, S., Moradkhani, H., Seo, D.J., Schwanenberg, D., Smith, P., van Dijk, A.I.J.M., van Velzen, N., He, M., Lee, H., Noh, S.J., Rakovec, O., Restrepo, P.: Advancing data assimilation in operational hydrologic forecasting: progresses, challenges, and emerging opportunities. Hydrol. Earth Syst. Sci. **16**(10), 3863–3887 (2012b). doi:10.5194/hess-16-3863-2012

Loaiciga, H.A., Charbeneau, R.J., Everett, L.G., Fogg, G.E., Hobbs, B.F., Rouhani, S.: Review of ground-water quality monitoring network design. J. Hydraul. Eng. **118**(1), 11–37 (1992)

Loeven, G.J.A., Witteveen, J.A.S., Bijlz, H.: Probabilistic collocation: an effcient non-intrusive approach for arbitrarily distributed parametric uncertainties. Paper presented at the 45th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 8–11 (2007)

Lopes, H.F., Tsay, R.S.: Particle filters and Bayesian inference in financial econometrics. J. Forecast. **30**(1), 168–209 (2011). doi:10.1002/for.1195

Lorenc, A.C.: The potential of the ensemble Kalman filter for NWP—a comparison with 4D-Var. Q. J. Roy. Meteor. Soc. **129**(595), 3183–3203 (2003)

Lu, G.Y., Wong, D.W.: An adaptive inverse-distance weighting spatial interpolation technique. Comput. Geosci. **34**(9), 1044–1055 (2008)

Luenberger, D.G., Ye, Y.: Linear and nonlinear programming, 3rd edn. International series in operations research and management science. Springer, New York (2008)

Lukas, M.A.: Regularization Methods. In: Encyclopedia of Environmetrics 5. Wiley (2006)

Lynch, D.R.: Numerical partial differential equations for environmental scientists and engineers: a first practical course. Springer, New York (2005)

Ma, X., Zabaras, N.: An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations. J. Comput. Phys. **228**(8), 3084–3113 (2009). doi:10.1016/j.jcp.2009.01.006

Madigan, D., Raftery, A.E.: Model selection and accounting for model uncertainty in graphical models using Occam's window. J. Am. Stat. Assoc. **89**(428), 1535–1546 (1994)

Madigan, D., York, J., Allard, D.: Bayesian graphical models for discrete data. International Statistical Review/Revue Internationale de Statistique, **63**(2), 215–232 (1995)

Madsen, H.: Automatic calibration of a conceptual rainfall-runoff model using multiple objectives. J. Hydrol. **235**(3–4), 276–288 (2000). doi:10.1016/s0022-1694(00)00279-1

Maidment, D.R.: Handbook of hydrology. McGraw-Hill, New York (1993)

Maier, H.R., Dandy, G.C.: The use of artificial neural networks for the prediction of water quality parameters. Water Resour. Res. **32**(4), 1013–1022 (1996)

Maier, H.R., Dandy, G.C.: Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. Environ. Model. Softw. **15**(1), 101–124 (2000)

Maier, H.R., Jain, A., Dandy, G.C., Sudheer, K.P.: Methods used for the development of neural networks for the prediction of water resource variables in river systems: current status and future directions. Environ. Model. Softw. **25**(8), 891–909 (2010). doi:10.1016/j.envsoft.2010.02.003

Majdalani, S., Ackerer, P.: Identification of groundwater parameters using an adaptative multiscale method. Ground Water **49**(4), 548–559 (2011)

Makowski, D., Wallach, D., Tremblay, M.: Using a Bayesian approach to parameter estimation; comparison of the GLUE and MCMC methods. Agronomie **22**(2), 191–203 (2002)

Mandel, J.: Use of the singular value decomposition in regression analysis. Am. Stat. **36**(1), 15–24 (1982)

Mariethoz, G., Kelly, B.F.: Modeling complex geological structures with elementary training images and transform-invariant distances. Water Resour. Res. **47**(7), W07527 (2011)

Mariethoz, G., Renard, P., Straubhaar, J.: The direct sampling method to perform multiple–point geostatistical simulations. Water Resour. Res. **46**(11), W11536 (2010)

Marler, R.T., Arora, J.S.: Survey of multi-objective optimization methods for engineering. Struct. Multidiscip. Optim. **26**(6), 369–395 (2004). doi:10.1007/s00158-003-0368-6

Marsily, G.d.: Quantitative hydrogeology: groundwater hydrology for engineers. Academic, Orlando (1986)

Matheron, G.: Traité de géostatistique appliquée. Memoires, 14, Bureau de Recherches Geologiques et Minieres (1962)

Matheron, G.: The theory of regionalized variables and its applications. In: Les Cahiers du Centre de Morphologie; Mathematique de Fontainebleau, Ecole des Mines de Paris (1971)

Mattera, D., Haykin, S.: Support vector machines for dynamic reconstruction of a chaotic system. In: Schölkopf, B., Burges, J., Smola, A. (eds.) Advances in Kernel methods: support vector machine. MIT Press, Cambridge (1999)

Mayer, A.S., Kelley, C., Miller, C.T.: Optimal design for problems involving flow and transport phenomena in saturated subsurface systems. Adv. Water Res. **25**(8), 1233–1256 (2002)

McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. **5**, 115–133 (1943)

McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**, 239–245 (1979)

McLaughlin, D., Townley, L.R.: A reassessment of the groundwater inverse problem. Water Resour. Res. **32**(5), 1131–1161 (1996). doi:10.1029/96wr00160

McPhee, J., Yeh, W.W.-G.: Multiobjective optimization for sustainable groundwater management in semiarid regions. J. Water Resour. Plan. Manag. **130**(6), 490–497 (2004)

McPhee, J., Yeh, W.W.-G.: Groundwater management using model reduction via empirical orthogonal functions. J. Water Resour. Plan. Manag. **134**(2), 161–170 (2008)

Medina, A., Carrera, J.: Coupled estimation of flow and solute transport parameters. Water Resour. Res. **32**(10), 3063–3076 (1996)

Medina, A., Carrera, J.: Geostatistical inversion of coupled problems: dealing with computational burden and different types of data. J. Hydrol. **281**(4), 251–264 (2003)

Meise, R., Vogt, D.: Introduction to functional analysis. Oxford graduate texts in mathematics, vol. 2. Oxford University Press, Oxford (1997)

Melas, V.B.: Functional approach to optimal experimental design. Lecture notes in statistics, vol. 184. Springer, New York (2006)

Mendenhall, W., Beaver, R.J., Beaver, B.M.: Introduction to probability and statistics, 13th edn. Brooks/Cole, Cengage Learning, Belmont (2009)

Mercer, J.W., Faust, C.R.: Ground-water modeling. National Water Well Association Worthington, Ohio (1981)

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equations of state calculations by fast computing machines. J. Chem. Phy. **21**, 1087–1091 (1953)

Meyer, P.D., Valocchi, A.J., Eheart, J.W.: Monitoring network design to provide initial detection of groundwater contamination. Water Resour. Res. **30**(9), 2647–2659 (1994)

Michael, H.A., Li, H., Boucher, A., Sun, T., Caers, J., Gorelick, S.M.: Combining geologic-process models and geostatistics for conditional simulation of 3-D subsurface heterogeneity. Water Resour. Res. **46**(5), W05527 (2010)

Michalak, A.M., Kitanidis, P.K.: Estimation of historical groundwater contaminant distribution using the adjoint state method applied to geostatistical inverse modeling. Water Resour. Res. **40**, W08302 (2004)

Michalak, A.M., Bruhwiler, L., Tans, P.P.: A geostatistical approach to surface flux estimation of atmospheric trace gases. J. Geophys. Res. Atmos. **109**(D14), 1984–2012 (2004)

Mirghani, B.Y., Zechman, E.M., Ranjithan, R.S., Mahinthakumar, G.: Enhanced simulation-optimization approach using surrogate modeling for solving inverse problems. Environ. Forensics. **13**(4), 348–363 (2012). doi:10.1080/15275922.2012.702333

Mishra, S., Parker, J.: Parameter estimation for coupled unsaturated flow and transport. Water Resour. Res. **25**(3), 385–396 (1989)

Mitchell, M.: An introduction to genetic algorithms. Complex adaptive systems. MIT Press, Cambridge (1996)

Mondal, A., Efendiev, Y., Mallick, B., Datta-Gupta, A.: Bayesian uncertainty quantification for flows in heterogeneous porous media using reversible jump Markov chain Monte Carlo methods. Adv. Water Resour. **33**(3), 241–256 (2010)

Moore, R.: The probability-distributed principle and runoff production at point and basin scales. Hydrol. Sci. J. **30**(2), 273–297 (1985)

Moore, R.: The PDM rainfall-runoff model. Hydrol. Earth Syst. Sci. **11**(1), 483–499 (2007)

Moore, C., Doherty, J.: Role of the calibration process in reducing model predictive error. Water Resour. Res. **41**, W05020 (2005)

Moradkhani, H., Hsu, K.-l., Gupta, H.V., Sorooshian, S.: Improved streamflow forecasting using self-organizing radial basis function artificial neural networks. J. Hydrol. **295**(1–4), 246–262 (2004). doi:10.1016/j.jhydrol.2004.03.027

Moradkhani, H., Sorooshian, S., Gupta, H.V., Houser, P.R.: Dual state-parameter estimation of hydrological models using ensemble Kalman filter. Adv. Water Resour. **28**(2), 135–147 (2005)

Moradkhani, H., DeChant, C.M., Sorooshian, S.: Evolution of ensemble data assimilation for uncertainty quantification using the particle filter–Markov chain Monte Carlo method. Water Resour. Res. **48**, W12520 (2012)

Morris, M.D.: Factorial sampling plans for preliminary computational experiments. Technometrics **33**(2), 161–174 (1991)

Morshed, J., Kaluarachchi, J.J.: Parameter estimation using artificial neural network and genetic algorithm for free-product migration and recovery. Water Resour. Res. **34**(5), 1101–1113 (1998)

Mosegaard, K., Sambridge, M.: Monte Carlo analysis of inverse problems. Inverse Prob. **18**(3), R29 (2002)

Muller, W.G.: Collecting spatial data: optimum design of experiments for random fields, p. 242, Springer (2007)

Myers, D.E.: Matrix formulation of co-kriging. J. Int. Assoc. Math. Geol. **14**(3), 249–257 (1982)

Neal, R.M.: Bayesian learning for neural networks. Lecture notes in statistics, vol. 118. Springer, New York (1996)

Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions—I. Math. Program. **14**(1), 265–294 (1978)

Neuman, S.P.: A statistical approach to the inverse problem of aquifer hydrology: 3. Improved solution method and added perspective. Water Resour. Res. **16**(2), 331–346 (1980)

Neuman, S.: Maximum likelihood Bayesian averaging of uncertain model predictions. Stoch. Env. Res. Risk A. **17**(5), 291–305 (2003)

Neupauer, R.M., Borchers, B.: A MATLAB implementation of the minimum relative entropy method for linear inverse problems. Comput. Geosci. **27**(7), 757–762 (2001)

Neupauer, R.M., Wilson, J.L.: Adjoint–derived location and travel time probabilities for a multidimensional groundwater system. Water Resour. Res. **37**(6), 1657–1668 (2001)

Nichols, N.: Mathematical concepts of data assimilation. In: Al Wle. (ed.) Data assimilation. pp. 13–39. Springer, Berlin (2010)

Nishikawa, T., Yeh, W.W.G.: Optimal pumping test design for the parameter identification of groundwater systems. Water Resour. Res. **25**(7), 1737–1747 (1989)

Nobile, F., Tempone, R., Webster, C.: A sparse grid stochastic collocation method for partial differential equations with random input data. SIAM J. Numer. Anal. **46**(5), 2309–2345 (2008). doi:10.1137/060663660

Nocedal, J., Wright, S.J.: Numerical optimization, 2nd edn. (Springer series in operations research). Springer, New York (2006)

Nordqvist, R.: Comparison of parameter estimation design criteria using a solute transport model with matrix diffusion. Groundwater **38**(6), 827–835 (2000)

Nowak, W., de Barros, F., Rubin, Y.: Bayesian geostatistical design: task–driven optimal site investigation when the geostatistical model is uncertain. Water Resour. Res. **46**, W03535 (2010)

Oden, J.T., Vemaganti, K.S.: Estimation of local modeling error and goal-oriented adaptive modeling of heterogeneous materials: I. Error estimates and adaptive algorithms. J. Comput. Phys. **164**(1), 22–47 (2000)

Oden, J.T., Zohdi, T.I.: Analysis and adaptive modeling of highly heterogeneous elastic structures. Comput. Methods Appl. Mech. Eng. **148**(3), 367–391 (1997)

Odeh, I., McBratney, A., Chittleborough, D.: Further results on prediction of soil properties from terrain attributes: heterotopic cokriging and regression-kriging. Geoderma **67**(3), 215–226 (1995)

Okabe, A.: Spatial tessellations: concepts and applications of Voronoi diagrams, 2nd ed. (Wiley series in probability and statistics). Wiley, Chichester (2000)

Oladyshkin, S., Nowak, W.: Data-driven uncertainty quantification using the arbitrary polynomial chaos expansion. Reliab. Eng. Syst. Saf. **106,** 179–190 (2012)

Olden, J.D., Joy, M.K., Death, R.G.: An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. Ecol. Model. **178**(3–4), 389–397 (2004). doi:10.1016/j.ecolmodel.2004.03.013

Oliver, D., Chen, Y.: Recent progress on reservoir history matching: a review. Comput. Geosci. **15**(1), 185–221 (2011). doi:10.1007/s10596-010-9194-2

Oliver, D.S., Cunha, L.B., Reynolds, A.C.: Markov chain Monte Carlo methods for conditioning a permeability field to pressure data. Math. Geol. **29**(1), 61–91 (1997)

Oliver, D.S., Reynolds, A.C., Liu, N.: Inverse theory for petroleum reservoir characterization and history matching. Cambridge University Press, Cambridge (2008)

Olofsson, P., Andersson, M.: Probability, statistics, and stochastic processes. 2nd edn. Wiley, Hoboken (2012)

Osher, S., Fedkiw, R.P.: Level set methods and dynamic implicit surfaces. Applied mathematical sciences, vol. 153. Springer, New York (2003)

Papalambros, P.Y., Wilde, D.J.: Principles of optimal design: modeling and computation, 2nd edn. Cambridge University Press, Cambridge (2000)

Park, T., Casella, G.: The bayesian lasso. J. Am. Stat. Assoc. **103**(482), 681–686 (2008)

Park, J., Sandberg, I.W.: Universal approximation using radial-basis-function networks. Neural. Comput. **3**(2), 246–257 (1991). doi:10.1162/neco.1991.3.2.246

Pasetto, D., Putti, M., Yeh, W.W.G.: A reduced–order model for groundwater flow equation with random hydraulic conductivity: application to Monte Carlo methods. Water Resour. Res. **49**(6), 3215–3228 (2013)

Pau, G.S.H., Zhang, Y., Finsterle, S.: Reduced order models for many-query subsurface flow applications. Comput. Geosci. **17**, 705–721 (2013)

Pawitan, Y.: In all likelihood: statistical modelling and inference using likelihood. Oxford University Press, Oxford (2001)

Pázman, A.: Foundations of optimum experimental design. Mathematics and its applications East European series. D. Reidel; Distributors for the U.S.A. and Canada, Kluwer, Dordrecht (1986)

Pebesma, E.J.: Multivariable geostatistics in S: the gstat package. Comput. Geosci. **30**(7), 683–691 (2004)

Petersen, K.B., Pedersen, M.S.: The matrix cookbook. Technical University of Denmark, Lyngby. http://matrixcookbook.com (2012)

Pham, D.T.: Stochastic methods for sequential data assimilation in strongly nonlinear systems. Mon. Weather Rev. **129**(5), 1194–1207 (2001)

Phillips, D.L.: A technique for the numerical solution of certain integral equations of the first kind. J. ACM **9**(1), 84–97 (1962). doi:10.1145/321105.321114

Phillips, G.M.: Interpolation and approximation by polynomials. CMS books in mathematics, vol. 14. Springer, New York (2003)

Phillips, C., Rugg, M.D., Friston, K.J.: Systematic regularization of linear inverse solutions of the EEG source localization problem. Neuroimage. **17**(1), 287–301 (2002)

Pintér, J.: Global optimization in action: continuous and lipschitz optimization-algorithms, implementations, and applications. Nonconvex optimization and its applications, vol. 6. Kluwer, Dordrecht (1996)

Plackett, R.L.: Some theorems in least squares. Biometrika. **37,** 149–157 (1950)

Plessix, R.-E.: A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. Geophys. J. Int. **167**(2), 495–503 (2006)

Poeter, E., Anderson, D.: Multimodel ranking and inference in ground water modeling. Ground Water **43**(4), 597–605 (2005)

Poeter, E.P., Hill, M.C.: MMA, a computer code for multi-model analysis. In: U. S. Geological Survey, Techniques and Methods, 6-E3, 113 p (2007)

Polak, E., Ribiere, G.: Note on the convergence of methods of conjugate directions. Revue Francaise d'Informatique et de Recherche Operationnelle **3**(16), 35–43 (1969)

Pollock, D.W.: Semianalytical computation of path lines for finite–difference models. Ground Water. **26**(6), 743–750 (1988)

Pollock, D., Cirpka, O.A.: Fully coupled hydrogeophysical inversion of a laboratory salt tracer experiment monitored by electrical resistivity tomography. Water Resour. Res. **48**, W01505 (2012)

Polson, N.G., Stroud, J.R., Müller, P.: Practical filtering with sequential parameter learning. J. Roy. Stat. Soc. B. **70**(2), 413–428 (2008). doi:10.1111/j.1467-9868.2007.00642.x

Polyanin, A.D., Manzhirov, A.V.: Handbook of mathematics for engineers and scientists. Chapman & Hall/CRC, Boca Raton (2007)

Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes: the art of scientific computing, 3rd edn. Cambridge University Press, Cambridge (2007)

Pronzato, L.: Optimal experimental design and some related control problems. Automatica **44**(2), 303–325 (2008)

Pronzato, L., Walter, E.: Robust experiment design via maximin optimization. Math. Biosci. **89**(2), 161–176 (1988)

Prudhomme, S., Oden, J.: Preface to the special issue on modeling error estimation and adaptive modeling in computational mechanics. Comput. Methods Appl. Mech. Eng. **200**(37), 2625 (2011)

Qian, S.-E., Chen, G.: A new nonlinear dimensionality reduction method with application to hyperspectral image analysis, pp. 270–273. IEEE, Barcelona (2007) (Geoscience and Remote Sensing Symposium, 2007. IGARSS 2007. IEEE International)

Quiñonero-Candela, J., Rasmussen, C.: Analysis of some methods for reduced rank Gaussian process regression. In: Murray-Smith, R., Shorten, R. (eds.) Switching and learning in feedback systems, vol. 3355. Lecture notes in computer science, pp. 98–127. Springer, Berlin (2005)

Raftery, A.E.: Approximate Bayes factors and accounting for model uncertainty in generalised linear models. Biometrika **83**(2), 251–266 (1996)

Raftery, A.E., Gneiting, T., Balabdaoui, F., Polakowski, M.: Using Bayesian model averaging to calibrate forecast ensembles. Monthly Weather Rev. **133**(5), 1155–1174 (2005)

Rall, L.B.: Automatic differentiation: techniques and applications. (Lecture Notes in Computer Science vol. 1 20). Springer, Berlin (1981)

RamaRao, B.S., LaVenue, A.M., De Marsily, G., Marietta, M.G.: Pilot Point Methodology for Automated Calibration of an Ensemble of conditionally Simulated Transmissivity Fields: 1. Theory and Computational Experiments. Water Resour. Res. **31**(3), 475–493 (1995). doi:10.1029/94wr02258

Ranieri, J., Chebira, A., Vetterli, M.: Near-optimal sensor placement for linear inverse problems, in IEEE Transaction on Signal Processing, vol. 62, mum. Issue 5, pp. 1135–1146 (2014)

Rasmussen, C.E., Nickisch, H.: Gaussian processes for machine learning (GPML) Toolbox. J. Mach. Learn. Res. **11**, 3011–3015 (2010)

Rasmussen, C.E., Williams, C.K.I.: Gaussian processes for machine learning. Adaptive computation and machine learning. MIT Press, Cambridge (2006)

Raykov, T., Marcoulides, G.A.: Introduction to applied multivariate analysis. Routledge, New York (2008)

Reagan, M.T., Najm, H.N., Ghanem, R.G., Knio, O.M.: Uncertainty quantification in reacting-flow simulations through non-intrusive spectral projection. Combustion Flame **132**(3), 545–555 (2003). doi:10.1016/s0010-2180(02)00503-5

Reed, P.M., Minsker, B.S.: Striking the balance: long-term groundwater monitoring design for conflicting objectives. J. Water Resour. Plan. Manage. **130**(2), 140–149 (2004)

Reed, P., Minsker, B., Valocchi, A.J.: Cost–effective long–term groundwater monitoring design using a genetic algorithm and global mass interpolation. Water Resour. Res. **36**(12), 3731–3741 (2000)

Reed, P., Minsker, B.S., Goldberg, D.E.: Simplifying multiobjective optimization: an automated design methodology for the nondominated sorted genetic algorithm-II. Water Resour. Res. **39**(7), 1196 (2003). doi:10.1029/2002wr001483

Reed, P.M., Hadka, D., Herman, J.D., Kasprzyk, J.R., Kollat, J.B.: Evolutionary multiobjective optimization in water resources: the past, present, and future. Adv. Water Resour. **51**, 438–456 (2013)

Refsgaard, J.C., Van der Sluijs, J.P., Brown, J., Van der Keur, P.: A framework for dealing with uncertainty due to model structure error. Adv. Water Resour. 29(11), 1586–1597 (2006)

Refsgaard, J.C., van der Sluijs, J.P., Højberg, A.L., Vanrolleghem, P.A.: Uncertainty in the environmental modelling process—A framework and guidance. Environ. Model. Amp. Softw. **22**(11), 1543–1556 (2007). doi:10.1016/j.envsoft.2007.02.004

Refsgaard, J.C., Christensen, S., Sonnenborg, T.O., Seifert, D., Højberg, A.L., Troldborg, L.: Review of strategies for handling geological uncertainty in groundwater flow and transport modeling. Adv. Water Resour. **36,** 36–50 (2012)

Reichle, R.H., Kumar, S.V., Mahanama, S.P., Koster, R.D., Liu, Q.: Assimilation of satellite-derived skin temperature observations into land surface models. J. Hydrometeorol. **11**(5), 1103–1122 (2010)

Remy, N., Boucher, A., Wu, J.: Applied geostatistics with SGeMS: a user's guide. Cambridge University Press, Cambridge (2009)

Renard, P., De Marsily, G.: Calculating equivalent permeability: a review. Adv. Water Resour. **20**(5), 253–278 (1997)

Renard, B., Kavetski, D., Kuczera, G., Thyer, M., Franks, S.W.: Understanding predictive uncertainty in hydrologic modeling: the challenge of identifying input and structural errors. Water Resour. Res. **46**(W05521) (2010). doi:10.1029/2009WR008328

Renka, R.J.: Algorithm 660: QSHEP2D: quadratic Shepard method for bivariate interpolation of scattered data. ACM Trans. Math. Softw. (TOMS). **14**(2), 149–150 (1988)

Resmerita, E., Otmar, S.: Error estimates for non-quadratic regularization and the relation to enhancement. Inverse Prob. **22**(3), 801 (2006)

Rings, J., Vrugt, J.A., Schoups, G., Huisman, J.A., Vereecken, H.: Bayesian model averaging using particle filtering and Gaussian mixture modeling: theory, concepts, and simulation experiments. Water Resour. Res. **48**(5), W05520 (2012). doi:10.1029/2011wr011607

Ripley, B.D.: Spatial statistics. Wiley series in probability and statistics. Wiley-Interscience, Hoboken (2004)

Ristic, B., Arulampalm, S., Gordon, N.: Beyond the kalman filter: particle filters for tracking applications. Artech House Publishers, USA (2004)

Riva, M., Guadagnini, A., De Gaspari, F., Alcolea, A.: Exact sensitivity matrix and influence of the number of pilot points in the geostatistical inversion of moment equations of groundwater flow. Water Resour. Res. **46**, W11513, doi:10.1029/2009WR008476 (2010)

Robert, C.P.: The Bayesian choice: from decision-theoretic foundations to computational implementation, 2nd edn. (Springer texts in statistics). Springer, New York (2007)

Robert, C.P., Casella, G.: Monte Carlo statistical methods, 2nd edn. (Springer texts in statistics). Springer, New York (2004)

Roberts, G.O., Rosenthal, J.S.: Markov-chain monte carlo: some practical implications of theoretical results. Can. J. Statist. **26**(1), 5–20 (1998). doi:10.2307/3315667

Rodell, M., Houser, P., Jambor, U., Gottschalck, J., Mitchell, K., Meng, C., Arsenault, K., Cosgrove, B., Radakovich, J., Bosilovich, M.: The global land data assimilation system. Bull. Am. Meteorol. Soc. **85**(3), 381–394 (2004)

Rojas, C.R., Welsh, J.S., Goodwin, G.C., Feuer, A.: Robust optimal experiment design for system identification. Automatica **43**(6), 993–1008 (2007)

Rojas, R., Feyen, L., Dassargues, A.: Conceptual model uncertainty in groundwater modeling: combining generalized likelihood uncertainty estimation and Bayesian model averaging. Water Resour. Res. **44**(12), W12418 (2008). doi:10.1029/2008wr006908

Rojas, C.R., Barenthin, M.B., Welsh, J.S., Hjalmarsson, H.: The cost of complexity in system identification: frequency function estimation of finite impulse response systems. Autom. Control IEEE Trans. **55**(10), 2298–2309 (2010). doi:10.1109/tac.2010.2063470

Rojas, C.R., Barenthin, M., Welsh, J.S., Hjalmarsson, H.: The cost of complexity in system identification: the Output Error case. Automatica **47**(9), 1938–1948 (2011)

Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)

Rubin, Y.: Applied stochastic hydrogeology. Oxford University Press, Oxford (2003)

Rubin, Y., Dagan, G.: Conditional estimation of solute travel time in heterogeneous formations: impact of transmissivity measurements. Water Resour. Res. **28**(4), 1033–1040 (1992)

Rue, H., Held, L.: Gaussian Markov random fields: theory and applications. Monographs on statistics and applied probability 104. Chapman & Hall/CRC, Boca Raton (2005)

Sakov, P., Oke, P.R.: A deterministic formulation of the ensemble Kalman filter: an alternative to ensemble square root filters. Tellus A. **60**(2), 361–371 (2008a)

Sakov, P., Oke, P.R.: Implications of the form of the ensemble transformation in the ensemble square root filters. Mon. Weather Rev. **136**(3), 1042–1053 (2008b)

Salamon, P., Feyen, L.: Assessing parameter, precipitation, and predictive uncertainty in a distributed hydrological model using sequential data assimilation with the particle filter. J. Hydrol. **376**(3), 428–442 (2009)

Saltelli, A.: Making best use of model evaluations to compute sensitivity indices. Comput. Phys. Commun. **145**(2), 280–297 (2002)

Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., Tarantola, S.: Global sensitivity analysis: the primer. Wiley, Chichester (2008)

Sambridge, M.: Finding acceptable models in nonlinear inverse problems using a neighbourhood algorithm. Inverse Probl. **17**(3), 387 (2001)

Sambridge, M., Braun, J., McQueen, H.: Geophysical parametrization and interpolation of irregular data using natural neighbours. Geophys. J. Int. **122**(3), 837–857 (1995). doi:10.1111/j.1365-246X.1995.tb06841.x

Sambridge, M., Rickwood, P., Rawlinson, N., Sommacal, S.: Automatic differentiation in geophysical inverse problems. Geophys. J. Int. **170**(1), 1–8 (2007)

Santosa, F.: A level-set approach for inverse problems involving obstacles. ESAIM Control Optim. Calc. Var. **1,** 17–33 (1996)

Sarma, P., Durlofsky, L., Aziz, K.: Kernel principal component analysis for efficient, differentiable parameterization of multipoint geostatistics. Math. Geosci. **40**(1), 3–32 (2008). doi:10.1007/s11004-007-9131-7

Sawaragi, Y., Nakayama, H., Tanino, T.: Theory of multiobjective optimization. Mathematics in science and engineering, vol. 176. Academic, Orlando (1985)

Schaffer, J.D.: Some experiments in machine learning using vector evaluated genetic algorithms, Thesis, Vanderbilt University (1984)

Schapire, R.E.: The strength of weak learnability. Mach. Learn. **5**, 197–227 (1990)

Scherzer, O.: The use of Morozov's discrepancy principle for Tikhonov regularization for solving nonlinear ill-posed problems. Computing **51**(1), 45–60 (1993). doi:10.1007/bf02243828

Schnoor, J.L.: Environmental modeling: fate and transport of pollutants in water, air, and soil. Environmental science and technology. Wiley, New York (1996)

Schölkopf, B., Smola, A., Müller, K.-R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural. Comput. **10**(5), 1299–1319 (1998)

Schöneberger, J.C., Arellano-Garcia, H., Wozny, G.n.: Local optima in model-based optimal experimental design. Ind. Eng. Chem. Res. **49**(20), 10059–10073 (2010)

Schoof, J.T., Pryor, S.: Downscaling temperature and precipitation: a comparison of regression–based methods and artificial neural networks. Int. J. Climatol. **21**(7), 773–790 (2001)

Schoups, G., Van de Giesen, N., Savenije, H.: Model complexity control for hydrologic prediction. Water Resour. Res. **44**(12), W00B03 (2008)

Schwarz, G.: Estimating the dimension of a model. Ann. Stat. **6**(2), 461–464 (1978)

Seidou, O., Asselin, J.J., Ouarda, T.B.M.J.: Bayesian multivariate linear regression with application to change point models in hydrometeorological variables. Water Resour. Res. **43,** W08401 (2007). doi:10.1029/2005WR004835

Seinfeld, J., Chen, W.: Identification of petroleum reservoir properties. In: Ray, W.H., Lainiotis, D.G. (eds.) Distributed parameter systems, identification, estimation, and control. Marcel Dekker, New York (1978)

Sethian, J.A.: Level Set Methods: evolving interfaces in geometry, fluid mechanics, computer vision, and materials science. Cambridge monographs on applied and computational mathematics 3. Cambridge University Press, Cambridge (1996)

Settles, B.: Active learning literature survey. In: Computer Science Technical Report 1648. University of Wisconsin, Madison, Midison, WI, (2010)

Shalev-Shwartz, S., Singer, Y., Srebro, N., Cotter, A.: Pegasos: primal estimated sub-gradient solver for svm. Math. Program. **127**(1), 3–30 (2011)

Shamaiah, M., Banerjee, S., Vikalo, H.: Greedy sensor selection: leveraging submodularity. In: 2010 49th IEEE Conference on Decision and Control (CDC), Atlanta, GA, pp. 2572–2577. IEEE (2010)

Shan, S., Wang, G.G.: Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. Struct. Multidisc. Optim. **41**(2), 219–241 (2010). doi:10.1007/s00158–009-0420–2

Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**(3), 379–423 (1948)

Sharkey, A.J.C. (ed.) Combining artificial neural nets: ensemble and modular multi-net systems. Springer, New York (1999)

Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 23rd ACM national conference, pp. 517–524. ACM (1968)

Shimizu, K., Aiyoshi, E.: Necessary conditions for min-max problems and algorithms by a relaxation procedure. Autom. Control IEEE Trans. **25**(1), 62–66 (1980)

Shu, C., Burn, D.H.: Artificial neural network ensembles and their application in pooled flood frequency analysis. Water. Resour. Res. **40**(9), W09301 (2004)

Sibson, R.: A brief description of natural neighbor interpolation. In: Barnett, V. (ed.) Interpreting multivariate data, pp. 21–36. Wiley, New York (1981)

Silvey, S.D.: Optimal design: an introduction to the theory for parameter estimation. Monographs on applied probability and statistics. Chapman and Hall, London (1980)

Simeonov, V., Stratis, J.A., Samara, C., Zachariadis, G., Voutsa, D., Anthemidis, A., Sofoniou, M., Kouimtzis, T.: Assessment of the surface water quality in Northern Greece. Water Res. **37**(17), 4119–4124 (2003). doi:10.1016/S0043–1354(03)00398–1

Simon, E., Bertino, L.: Gaussian anamorphosis extension of the DEnKF for combined state parameter estimation: application to a 1D ocean ecosystem model. J. Marine. Syst. **89**(1), 1–18 (2012). doi:http://dx.doi.org/10.1016/j.jmarsys.2011.07.007

Singh, V.P.: Computer models of watershed hydrology, Revised edition. ed. Water Resources Publications, Highlands Ranch (2012)

Singh, T.S., Chakrabarty, D.: Multi-objective optimization for optimal groundwater remediation design and management systems. Geosci. J. **14**(1), 87–97 (2010)

Singh, A., Minsker, B.S.: Uncertainty-based multiobjective optimization of groundwater remediation design. Water Resour. Res. **44**(2), W02404 (2008)

Singh, A., Minsker, B.S., Valocchi, A.J.: An interactive multi-objective optimization framework for groundwater inverse modeling. Adv. Water Resour. **31**(10), 1269–1283 (2008)

Singh, A., Mishra, S., Ruskauff, G.: Model averaging techniques for quantifying conceptual model uncertainty. Ground Water **48**(5), 701–715 (2010)

Sivanandam, S.N., Deepa, S.N.: Introduction to genetic algorithms. Springer, Berlin (2007)

Smith, K.W.: Cluster ensemble Kalman filter. Tellus A. **59**(5), 749–757 (2007). doi:10.1111/j.1600-0870.2007.00246.x

Smith, R.A., Kummerow, C.D.: A comparison of in Situ, reanalysis, and satellite water budgets over the upper Colorado River basin. J. Hydrometeorol. **14**(3), 888–905 (2013). doi:10.1175/jhm-d-12-0119.1

Smithson, M.: Ignorance and uncertainty: emerging paradigms. Cognitive science. Springer, New York (1989)

Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Stat. Comput. **14**(3), 199–222 (2004)

Smolyak, S.: Quadrature and interpolation formulas for tensor products of certain classes of functions. Soviet Math Dokl (4), 240–243 (1963)

Sobol', I.M.: Sensitivity estimates for nonlinear mathematical models. Math. Model Comput. Exp. **1**(4), 407–417 (1993)

Sorenson, H.W., Alspach, D.L.: Recursive bayesian estimation using gaussian sums. Automatica **7**(4), 465–479 (1971)

Sorooshian, S.: Hydrological modelling and the water cycle: coupling the atmospheric and hydrological models. Water science and technology library, vol. 63. Springer, Berlin (2008)

Spearman, C.: General intelligence objectively determined and measured. Am. J. Psychol. **5**, 201–293 (1904)

Sprott, D.A.: Statistical inference in science. (Springer series in statistics). Springer, New York (2000)

Srinivas, N., Deb, K.: Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol. Comput. **2**(3), 221–248 (1994)

Stein, M.L.: Space–time covariance functions. J. Am. Stat. Assoc. **100**(469), 310–321 (2005)

Stein, E.M., Shakarchi, R.: Functional analysis: introduction to further topics in analysis. Princeton lectures in analysis, vol. 4. Princeton University Press, Princeton (2011)

Stein, A., Meer, F.v.d., Gorte, B.: Spatial statistics for remote sensing. Remote sensing and digital image processing, vol. 1. Kluwer, Dordrecht (1999)

Steinberg, D.M., Hunter, W.G.: Experimental design: review and comment. Technometrics **26**(2), 71–97 (1984)

Stern, P.C., Fineberg, H.V., National Research Council (U.S.). (Committee on Risk Characterization).: Understanding risk: informing decisions in a democratic society. National Academy Press, Washington, D.C. (1996)

Stoer, J., Bulirsch, R.: Introduction to numerical analysis, 3rd edn. Texts in applied mathematics, vol. 12. Springer, New York (2002)

Strang, G.: Linear algebra and its applications, 4th edn. Thomson, Brooks/Cole, Belmont (2006)

Strebelle, S.: Sequential simulation drawing structures from training images. Stanford University, California (2000)

Su, H., Yang, Z.L., Niu, G.Y., Dickinson, R.E.: Enhancing the estimation of continental–scale snow water equivalent by assimilating MODIS snow cover with the ensemble Kalman filter. J. Geophys. Res-Atmos. **113**(D8) Article No. D08120 (2008)

Sudret, B.: Global sensitivity analysis using polynomial chaos expansions. Reliab. Eng. Syst. Saf. **93**(7), 964–979 (2008). doi:10.1016/j.ress.2007.04.002

Sugihara, K.: Robust gift wrapping for the three-dimensional convex hull. J. Comput. Syst. Sci. **49**(2), 391–407 (1994)

Suman, B., Kumar, P.: A survey of simulated annealing as a tool for single and multiobjective optimization. J. Oper. Res. Soc. **57**(10), 1143–1160 (2005)

Sumengen, B.: Variational image segmentation and curve evolution on natural images. University of California, Santa Barbara (2004)

Sun, N.-Z.: Inverse problems in groundwater modeling. (Theory and applications of transport in porous media, vol. 6). Kluwer, Dordrecht (1994)

Sun, N.-Z.: Mathematical modeling of groundwater pollution: with 104 illustrations. Springer, New York (1996)

Sun, N.-Z.: Modeling biodegradation processes in porous media by the finite cell method. Water Resour. Res. **38**(3), 11–12 (2002). doi:10.1029/2000wr000198

Sun, N.-Z.: Structure reduction and robust experimental design for distributed parameter identification. Inverse Probl. **21**(2), 739 (2005)

Sun, A.: Enabling collaborative decision-making in watershed management using cloud-computing services. Environ. Model. Softw. **41,** 93–97 (2013a)

Sun, A.Y.: Predicting groundwater level changes using GRACE data. Water Resour. Res. **49**(9), 5900–5912 (2013b)

Sun, A.Y., Nicot, J.-P.: Inversion of pressure anomaly data for detecting leakage at geologic carbon sequestration sites. Adv. Water Resour. **44**, 20–29 (2012)

Sun, N.-Z., Sun, A.Y.: Parameter identification of environmental systems. In: Shen, H.H., Cheng, A.H.D., Wang, K.-H., Teng, M.H., Liu, C.C.K. (eds.) Environmental fluid mechanics: theories and applications, pp. 297–337. ASCE, Reston (2002)

Sun, N.-Z., Yeh, W.W.-G.: Identification of parameter structure in groundwater inverse problem. Water Resour. Res. **21**(6), 869–883 (1985)

Sun, N.-Z., Yeh, W.W.-G.: Coupled inverse problems in groundwater modeling: 2. Identifiability and experimental design. Water Resour. Res. **26**(10), 2527–2540 (1990a)

Sun, N.-Z., Yeh, W.W.-G.: Coupled inverse problems in groundwater modeling: 2. Identifiability and experimental design. Water Resour. Res. **26**(10), 2527–2540 (1990b)

Sun, N.-Z., Yeh, W.-G.: On the consistency of continuous and discrete approaches in deriving the adjoint state equations. In: Russell, T.F., Ewing, R.E., Brebbia, C.A., Gray, W.G., Pinder, G.F. (eds.) The 9th international conference on computational methods in water resources, pp. 187–194. Computational Mechanics Publications, Denver (1992a)

Sun, N.-Z., Yeh, W.W.G.: A stochastic inverse solution for transient groundwater flow: parameter identification and reliability analysis. Water Resour. Res. **28**(12), 3269–3280 (1992b). doi:10.1029/92wr00683

Sun, N.Z., Yeh, W.W.G.: Development of objective–oriented groundwater models: 1. Robust parameter identification. Water Resour. Res. **43**, W02420 (2007a)

Sun, N.-Z., Yeh, W.W.-G.: Development of objective-oriented groundwater models: 2. Robust experimental design. Water Resour. Res. **43**(2), W02421 (2007b)

Sun, N.Z., Jeng, M.C., Yeh, W.W.G.: A proposed geological parameterization method for parameter identification in three–dimensional groundwater modeling. Water Resour. Res. **31**(1), 89–102 (1995)

Sun, N.Z., Yang, S.l., Yeh, W.W.G.: A proposed stepwise regression method for model structure identification. Water Resour. Res. **34**(10), 2561–2572 (1998)

Sun, A.Y., Painter, S.L., Wittmeyer, G.W.: A constrained robust least squares approach for contaminant release history identification. Water Resour. Res. **42**(4), W04414 (2006a)

Sun, A.Y., Painter, S.L., Wittmeyer, G.W.: A robust approach for iterative contaminant source location and release history recovery. J. Contam. Hydrol. **88**(3), 181–196 (2006b)

Sun, A.Y., Ritzi, R.W., Sims, D.W.: Characterization and modeling of spatial variability in a complex alluvial aquifer: implications on solute transport. Water Resour. Res. **44**(4), W04402 (2008)

Sun, A.Y., Morris, A., Mohanty, S.: Comparison of deterministic ensemble Kalman filters for assimilating hydrogeological data. Adv. Water Resour. **32**(2), 280–292 (2009a)

Sun, A.Y., Morris, A.P., Mohanty, S.: Sequential updating of multimodal hydrogeologic parameter fields using localization and clustering techniques. Water Resour. Res. **45**(7), W07424 (2009b). doi:10.1029/2008wr007443

Sun, A.Y., Green, R., Rodell, M., Swenson, S.: Inferring aquifer storage parameters using satellite and in situ measurements: estimation under uncertainty. Geophys. Res. Lett. **37**(10), L10401 (2010)

Sun, A.Y., Green, R., Swenson, S., Rodell, M.: Toward calibration of regional groundwater models using GRACE data. J. Hydrol. **117,** 199–210 (2012). doi:10.1016/j.jhydrol.2011.10.025

Sun, A.Y., Nicot, J.-P., Zhang, X.: Optimal design of pressure-based, leakage detection monitoring networks for geologic carbon sequestration repositories. Int. J. Greenh. Gas Control **19,** 251–261 (2013)

Sun, A.Y., Miranda, R.M., Xu, X.: Development of multi-metamodels to support surface water quality management and decision making. Environ. Earth Sci. **73**(1), 423–434. doi:10.1007/s12665-014-3448-6 (2014a)

Sun, A.Y., Wang, D., Xu, X.: Monthly streamflow forecasting using Gaussian Process Regression. J. Hydrol. **511**, 72–81 (2014b). doi:http://dx.doi.org/10.1016/j.jhydrol.2014.01.023

Swenson, S., Wahr, J.: Post-processing removal of correlated errors in GRACE data. Geophys. Res. Lett. **33**(8), L08402 (2006)

Szunyogh, I., Kostelich, E.J., Gyarmati, G., Kalnay, E., Hunt, B.R., Ott, E., Satterfield, E., Yorke, J.A.: A local ensemble transform Kalman filter data assimilation system for the NCEP global model. Tellus A. **60**(1), 113–130 (2008)

Szymkiewicz, R.: Numerical modeling in open channel hydraulics. Water science and technology library, vol. 83. Springer, Dordrecht (2010)

Tai, X.-C., Chan, T.F.: A survey on multiple level set methods with applications for identifying piecewise constant functions. Int. J. Numer. Anal. Model. **1**(1), 25–47 (2004)

Tan, K.C., Khor, E.F., Lee, T.H.: Multiobjective evolutionary algorithms and applications. Advanced information and knowledge processing. Springer, London (2005)

Tan, C.-C., Tung, C.-P., Chen, C.-H., Yeh, W.W.-G.: An integrated optimization algorithm for parameter structure identification in groundwater modeling. Adv. Water Res. **31**(3), 545–560 (2008)

Tang, Y., Reed, P., van Werkhoven, K., Wagener, T.: Advancing the identification and evaluation of distributed rainfall-runoff models using global sensitivity analysis. Water Resour. Res. **43**(6), W06415 (2007a). doi:10.1029/2006wr005813

Tang, Y., Reed, P.M., Kollat, J.B.: Parallelization strategies for rapid and robust evolutionary multiobjective optimization in water resources applications. Adv Water Resour. **30**(3), 335–353 (2007b). doi:10.1016/j.advwatres.2006.06.006

Tapley, B.D., Bettadpur, S., Watkins, M., Reigber, C.: The gravity recovery and climate experiment: mission overview and early results. Geophys. Res. Lett. (31), L09607 (2004). doi:10.1029/2004GL019920

Tarantola, A.: Inverse problem theory and methods for model parameter estimation. Society for Industrial and Applied Mathematics, Philadelphia (2005)

Tareghian, R., Rasmussen, P.F.: Statistical downscaling of precipitation using quantile regression. J. Hydrol. **487**, 122–135 (2013). doi:http://dx.doi.org/10.1016/j.jhydrol.2013.02.029

Tartakovsky, A.M., Meakin, P., Scheibe, T.D., Eichler West, R.M.: Simulations of reactive transport and precipitation with smoothed particle hydrodynamics. J. Comput. Phys. **222**(2), 654–672 (2007)

Telen, D., Logist, F., Van Derlinden, E., Tack, I., Van Impe, J.: Optimal experiment design for dynamic bioprocesses: a multi-objective approach. Chem. Eng. Sci. **78**(0), 82–97 (2012). doi:http://dx.doi.org/10.1016/j.ces.2012.05.002

Tibshirani, R.: Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Ser. B. Methodol. **58,** 267–288 (1996)

Tiedeman, C.R., Hill, M.C., D'Agnese, F.A., Faunt, C.C.: Methods for using groundwater model predictions to guide hydrogeologic data collection, with application to the death valley regional groundwater flow system. Water Resour. Res. **39**(1), 1010 (2003)

Tikhonov, A.N., Arsenin, V.Y.: Solutions of ill-posed problems. Wiley, New York (1977)

Tippett, M.K., Anderson, J.L., Bishop, C.H., Hamill, T.M., Whitaker, J.S.: Ensemble square root filters. Mon. Weather Rev. **131**(7), 1485–1490 (2003)

Tipping, M.E.: Sparse Bayesian learning and the relevance vector machine. J. Mach. Learn. Res. **1**, 211–244 (2001)

Tipping, M.E.: Sparse Bayesian 2.0 User's Manual. http://www.relevancevector.com (2009)

Tisseuil, C., Vrac, M., Lek, S., Wade, A.J.: Statistical downscaling of river flows. J. Hydrol. **385**(1–4), 279–291 (2010). doi:http://dx.doi.org/10.1016/j.jhydrol.2010.02.030

Tonkin, M.J., Doherty, J.: A hybrid regularized inversion methodology for highly parameterized environmental models. Water Resour. Res. **41**, W10412 (2005)

Townley, L.R., Wilson, J.L.: Computationally efficient algorithms for parameter estimation and uncertainty propagation in numerical models of groundwater flow. Water Resour. Res. **21**(12), 1851–1860 (1985)

Tsai, F.T.C., Yeh, W.W.G.: Characterization and identification of aquifer heterogeneity with generalized parameterization and Bayesian estimation. Water. Resour. Res. **40**(10), W10102 (2004). doi:10.1029/2003wr002893

Tsai, F.T.-C., Sun, N.-Z., Yeh, W.W.-G.: Global-local optimization for parameter structure identification in three-dimensional groundwater modeling. Water Resour. Res. **39**(2), 1043 (2003)

Uciński, D.: Optimal measurement methods for distributed parameter system identification. CRC Press, Boca Raton (2005)

Van Genuchten, M.T.: A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. Soil Sci. Soc. Am. J. **44**(5), 892–898 (1980)

Van Lamsweerde, A.: Goal-oriented requirements engineering: a guided tour. In: Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on Requirements Engineering, pp. 249–262. IEEE (2001)

van de Merwe, R.: Sigma-point Kalman filters for probabilistic inference in dynamic state-space models. Oregon Health & Science University, Portland (2004)

van der Maaten, L.J.P.: An Introduction to Dimensionality Reduction Using Matlab. In., vol. Technical Report MICC 07-07. Maastricht University, Maastricht (2007)

van Genuchten, M.T., Alves, W.J.: Analytical solutions of the one-dimensional convective-dispersive solute transport equation. In: U.S. Department of Agriculture Technical Bulletin vol. 1661. p. 151. U.S. Salinity Laboratory, Riverside (1982)

Vapnik, V.N.: The nature of statistical learning theory. Springer, New York (1995)

Vapnik, V.N.: Statistical learning theory. Adaptive and learning systems for signal processing, communications, and control. Wiley, New York (1998)

Vapnik, V., Chervonenkis, A.: Theory of pattern recognition [in Russian]. Nauka, Moscow (1974)

Vargas-Guzmán, J.A., Yeh, T.C.: Sequential kriging and cokriging: two powerful geostatistical approaches. Stoch. Env. Res. Risk A. **13**(6), 416–435 (1999). doi:10.1007/s004770050047

Venditti, D.A., Darmofal, D.L.: Adjoint error estimation and grid adaptation for functional outputs: application to quasi-one-dimensional flow. J. Comput. Phys. **164**(1), 204–227 (2000)

Vese, L.A., Chan, T.F.: A multiphase level set framework for image segmentation using the mumford and shah model. Int. J. Comput. Vis. **50**(3), 271–293 (2002). doi:10.1023/a:1020874308076

Vrugt, J.A., Robinson, B.A.: Treatment of uncertainty using ensemble methods: comparison of sequential data assimilation and Bayesian model averaging. Water Resour. Res. **43**, W01411 (2007)

Vrugt, J.A., Gupta, H.V., Bouten, W., Sorooshian, S.: A Shuffled Complex Evolution Metropolis algorithm for optimization and uncertainty assessment of hydrologic model parameters. Water Resour. Res. **39**(8), 1201 (2003)

Vrugt, J.A., Diks, C.G., Gupta, H.V., Bouten, W., Verstraten, J.M.: Improved treatment of uncertainty in hydrologic modeling: combining the strengths of global optimization and data assimilation. Water Resour. Res. **41**(1), W01017 (2005)

Vrugt, J.A., Ter Braak, C.J., Clark, M.P., Hyman, J.M., Robinson, B.A.: Treatment of input uncertainty in hydrologic modeling: doing hydrology backward with Markov chain Monte Carlo simulation. Water Resour. Res. **44**, W00B09 (2008)

Vrugt, J.A., Ter Braak, C., Diks, C., Robinson, B.A., Hyman, J.M., Higdon, D.: Accelerating Markov chain Monte Carlo simulation by differential evolution with self-adaptive randomized subspace sampling. Int. J. Nonlinear Sci. Numer. Simul. **10**(3), 273–290 (2009)

Vrugt, J.A., ter Braak, C.J., Diks, C.G., Schoups, G.: Hydrologic data assimilation using particle Markov chain Monte Carlo simulation: theory, concepts and applications. Adv. Water Resour. **51**, 457–478 (2013)

Wagner, B.J.: Sampling design methods for groundwater modeling under uncertainty. Water Resour. Res. **31**(10), 2581–2591 (1995)

Wagner, B.J., Gorelick, S.M.: Optimal groundwater quality management under parameter uncertainty. Water Resour. Res. **23**(7), 1162–1174 (1987)

Wagener, T., Boyle, D.P., Lees, M.J., Wheater, H.S., Gupta, H.V., Sorooshian, S.: A framework for development and application of hydrological models. Hydrol. Earth Syst. Sci. **5**(1), 13–26 (2001). doi:10.5194/hess-5-13-2001

Wagener, T., Wheater, H., Gupta, H.V.: Rainfall-runoff modelling in gauged and ungauged catchments. Imperial College Press, London (2004)

Wainwright, H.M., Finsterle, S., Zhou, Q., Birkholzer, J.T.: Modeling the performance of large-scale CO2 storage systems: a comparison of different sensitivity analysis methods. Int. J. Greenh. Gas Control **17**(0), 189–205 (2013). doi:http://dx.doi.org/10.1016/j.ijggc.2013.05.007

Walker, W.E., Harremoës, P., Rotmans, J., van der Sluijs, J.P., van Asselt, M.B.A., Janssen, P., Krayer von Krauss, M.P.: Defining uncertainty: a conceptual basis for uncertainty management in model-based decision support. Integr. Assess. **4**(1), 5–17 (2003). doi:10.1076/iaij.4.1.5.16466

Walpole, R.E.: Probability & statistics for engineers & scientists, 8th edn. Pearson Prentice Hall, Upper Saddle River (2007)

Walpole, R.E.: Essentials of probability & statistics for engineers & scientists. Pearson, Boston (2013)

Walter, E., Pronzato, L.: Identification of parametric models from experimental data. Communications and control engineering. Springer, Berlin (1997)

Wan, X., Karniadakis, G.E.: Multi-element generalized polynomial chaos for arbitrary probability measures. SIAM J. Sci. Comput. **28**(3), 901–928 (2006)

Wan, E.A., Van Der Merwe, R.: The unscented Kalman filter for nonlinear estimation. In: Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC., Lake Louise, Alberta, Canada 2000, pp. 153–158. IEEE (2000)

Wan, J., Zabaras, N.: A Bayesian approach to multiscale inverse problems using the sequential Monte Carlo method. Inverse Probl. **27**(10), 105004 (2011)

Wang, G.G., Shan, S.: Review of metamodeling techniques in support of engineering design optimization. J. Mech. Design **129,** 370 (2007)

Wang, J., Zabaras, N.: Hierarchical Bayesian models for inverse problems in heat conduction. Inverse Prob. **21**(1), 183 (2005)

Wang, J., Zabaras, N.: A Markov random field model of contamination source identification in porous media flow. Int. J. Heat Mass. Transf. **49**(5–6), 939–950 (2006)

Wang, M.Y., Wang, X.M., Guo, D.M.: A level set method for structural topology optimization. Comput. Method. Appl. Mech. Eng. **192**, 227–246 (2003)

Wang, Y., Yagola, A.G., Yang, C.: Optimization and regularization for computational inverse problems and applications. Springer, Heidelberg, (2010)

Webster, R., Oliver, M.A.: Geostatistics for environmental scientists. Statistics in practice. Wiley, New York (2001)

Weinberger, K., Saul, L.: Unsupervised learning of image manifolds by semidefinite programming. Int. J. Comput. V. **70**(1), 77–90 (2006). doi:10.1007/s11263-005-4939-z

Weissmann, G.S., Zhang, Y., LaBolle, E.M., Fogg, G.E.: Dispersion of groundwater age in an alluvial aquifer system. Water Resour. Res. **38**(10), 1198 (2002)

Weisstein, E.W.: Normal Product Distribution, from MathWorld. http://mathworld.wolfram.com/NormalProductDistribution.html (2013)

Wen, X.-H., Capilla, J., Deutsch, C., Gómez-Hernández, J., Cullick, A.: A program to create permeability fields that honor single-phase flow rate and pressure data. Comput. Geosci. **25**(3), 217–230 (1999)

Whittle, P.: The analysis of multiple stationary time series. J. R. Stat. Soc. Series. B. Stat. Methodol. **15**(1), 125–139 (1953)

Wiener, N.: The homogeneous chaos. Am. J. Math. **60**(4), 897–936 (1938)

Wikle, C.K.: Hierarchical models in environmental science. Int. Stat. Rev. **71**(2), 181–199 (2003)

Wikle, C.K., Berliner, L.M., Cressie, N.: Hierarchical Bayesian space-time models. Environ. Ecol. Stat. **5**(2), 117–154 (1998)

Williams, C.K.: Computation with infinite neural networks. Neural Comput. **10**(5), 1203–1216 (1998)

Woodbury, A.D., Rubin, Y.: A full-Bayesian approach to parameter inference from tracer travel time moments and investigation of scale effects at the cape cod experimental site. Water Resour. Res. **36**(1), 159–171 (2000)

Woodbury, A.D., Ulrych, T.J.: Minimum relative entropy inversion: theory and application to recovering the release history of a groundwater contaminant. Water Resour. Res. **32**(9), 2671–2681 (1996)

Woodbury, A.D., Ulrych, T.J.: A full-Bayesian approach to the groundwater inverse problem for steady state flow. Water Resour. Res. **36**(8), 2081–2093 (2000)

Wu, J., Zhang, T., Journel, A.G.: Fast FILTERSIM simulation with score-based distance. Math. Geol. **40**, 773–788 (2008). doi:10.1007/s11004-008-9157–5

Xie, X., Zhang, D.: Data assimilation for distributed hydrological catchment modeling via ensemble Kalman filter. Adv. Water Resour. **33**(6), 678–690 (2010)

Xiu, D.: Numerical methods for stochastic computations: a spectral method approach. Princeton University Press, Princeton (2010)

Xiu, D.B., Hesthaven, J.S.: High-order collocation methods for differential equations with random inputs. SIAM J. Sci. Comput. **27**(3), 1118–1139 (2005). doi:10.1137/040615201

Xiu, D., Karniadakis, G.E.: The Wiener-Askey polynomial chaos for stochastic differential equations. SIAM J. Sci. Comput. **24**(2), 619–644 (2002)

Xu, L., Krzyzak, A., and Oja, E.: Rival penalized competitive learning for clustering analysis, RBF net, and curve detection. IEEE Trans. Neural Netw. **4**(4), 636–649 (1993)

Xu, X., Scanlon, B.R., Schilling, K., Sun, A.: Relative importance of climate and land surface changes on hydrologic changes in the US midwest since the 1930s. J. Hydrol. **47**, 110–120 (2013)

Yapo, P.O., Gupta, H.V., Sorooshian, S.: Multi-objective global optimization for hydrologic models. J. Hydrol. **204**(1–4), 83–97 (1998). doi:10.1016/s0022-1694(97)00107-8

Ye, M., Neuman, S.P., Meyer, P.D.: Maximum likelihood Bayesian averaging of spatial variability models in unsaturated fractured tuff. Water Resour. Res. **40**, W05113 (2004)

Ye, M., Meyer, P.D., Neuman, S.P.: On model selection criteria in multimodel analysis. Water Resour. Res. **44**(3) (2008). doi:10.1029/2008WR006803

Yeh, T.C.J., Liu, S., Glass, R.J., Baker, K., Brainard, J.R., Alumbaugh, D., LaBrecque, D.: A geostatistically based inverse model for electrical resistivity surveys and its applications to vadose zone hydrology. Water Resour. Res. **38**(12), 11–14 (2002)

Yoon, S., Malallah, A., Datta-Gupta, A., Vasco, D., Behrens, R.: A multiscale approach to production-data integration using streamline models. SPE J. **6**(2), 182–192 (2001)

Young, P.C.: Hypothetico-inductive data-based mechanistic modeling of hydrological systems. Water Resour. Res. **49**(2), 915–935 (2013). doi:10.1002/wrcr.20068

Yustres, Á., Asensio, L., Alonso, J., Navarro, V.: A review of Markov Chain Monte Carlo and information theory tools for inverse problems in subsurface flow. Comput. Geosci. **16**(1), 1–20 (2012)

Zafari, M., Reynolds, A.: Assessing the uncertainty in reservoir description and performance predictions with the ensemble Kalman filter. In: SPE Annual Technical Conference and Exhibition (2005)

Zannetti, P.: Air pollution modeling: theories, computational methods, and available software. Van Nostrand Reinhold, New York, (1990)

Zealand, C.M., Burn, D.H., Simonovic, S.P.: Short term streamflow forecasting using artificial neural networks. J. Hydrol. **214**(1), 32–48 (1999)

Zeng, L., Zhang, D.: A stochastic collocation based Kalman filter for data assimilation. Computat. Geosci. **14**(4), 721–744 (2010)

Zhan, L., Yortsos, Y.C.: A direct method for the identification of the permeability field of an anisotropic porous medium. Water Resour. Res. **37**(7), 1929–1938 (2001)

Zhang, D.: Stochastic methods for flow in porous media: coping with uncertainties. Academic, San Diego (2002)

Zhang, T.: Filter-based training pattern classification for spatial pattern simulation. Stanford University, Stanford (2006)

Zhang, D., Lu, Z.: An efficient, high-order perturbation approach for flow in random porous media via Karhunen–Loève and polynomial expansions. J. Comput. Phys. **194,** 773–794 (2004)

Zhang, D., Andricevic, R., Sun, A.Y., Hu, X., He, G.: Solute flux approach to transport through spatially nonstationary flow in porous media. Water Resour. Res. **36**(8), 2107–2120 (2000)

Zhang, Y., Pinder, G.F., Herrera, G.S.: Least cost design of groundwater quality monitoring networks. Water Resources Research **41**, W08412 (2005)

Zhang, D., Lu, Z., Chen, Y.: Dynamic reservoir data assimilation with an efficient, dimension-reduced Kalman filter. SPE J. **12**(1), 108–117 (2007)

Zhang, G., Lu, D., Ye, M., Gunzburger, M., Webster, C.: An adaptive sparse–grid high–order stochastic collocation method for Bayesian inference in groundwater reactive transport modeling. Water Resour. Res. **49**(10), 6871–6892 (2013)

Zhao, R.-J.: The Xinanjiang model applied in China. J. Hydrol. **135**(1–4), 371–381 (1992). doi:10.1016/0022-1694(92)90096-e

Zhao, Y., Reynolds, A., Li, G.: Generating facies maps by assimilating production data and seismic data with the ensemble Kalman filter. In: SPE/DOE Symposium on Improved Oil Recovery, 20–23 April, Tulsa, Oklahoma, USA (2008)

Zhao, R.J., Zhaung, Y.L., Fang, L.R., Lui, X.R., Zhang, Q.S.: The Xinanjiang model. Paper presented at the In Hydrological Forecasting, Proceedings of the Oxford Symposium, IAHS publ. No. 129

Zheng, C., Wang, P.: Parameter structure identification using tabu search and simulated annealing. Adv. Water Resour. **19**(4), 215–224 (1996)

Zheng, C., Wang, P.P.: An integrated global and local optimization approach for remediation system design. Water Resour. Res. **35**(1), 137–148 (1999)

Zhigljavsky, A.: Stochastic global optimization, 1st ed. (Springer optimization and its applications, vol. 9). Springer, New York (2007)

Zhou, Z.-H.: Ensemble methods: foundations and algorithms. Chapman & Hall/CRC machine learning & pattern recognition series. Taylor & Francis, Boca Raton, (2012)

Zhou, H., Gómez-Hernández, J.J., Hendricks Franssen, H.-J., Li, L.: An approach to handling non-Gaussianity of parameters and state variables in ensemble Kalman filtering. Adv. Water Resour. **34**(7), 844–864 (2011)

Zhou, Y., Obenour, D.R., Scavia, D., Johengen, T.H., Michalak, A.M.: Spatial and temporal trends in lake Erie Hypoxia, 1987–2007. Environ. Sci. Technol. **47**(2), 899–905 (2013a)

Zhou, Y., Ritzi, R.W., Soltanian, M.R., Dominic, D.F.: The influence of streambed heterogeneity on hyporheic flow in gravelly rivers. Ground. Water. **52,** 206–216 (2013b). doi:10.1111/gwat.12048

Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)

Zlatev, Z.: Computer treatment of large air pollution models. Environmental science and technology library, vol. 2. Kluwer, Dordrecht (1995)

Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. J. R. Stat. Soc. Ser. B. Stat. Methodol. **67**(2), 301–320 (2005)

Index