Ryan G. McClarren

Uncertainty Quantification and Predictive Computational Science

A Foundation for Physical Scientists and Engineers

EXTRAS ONLINE

Springer

Uncertainty Quantification and Predictive Computational Science

Ryan G. McClarren

Uncertainty Quantification and Predictive Computational Science

A Foundation for Physical Scientists and Engineers

Springer

Ryan G. McClarren
University of Notre Dame
Department of Aerospace
and Mechanical Engineering
Notre Dame, IN, USA

ISBN 978-3-319-99524-3 ISBN 978-3-319-99525-0 (eBook)
https://doi.org/10.1007/978-3-319-99525-0

Library of Congress Control Number: 2018961189

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To Beatrix, Flannery, Lowry, and Cormac for the joyous uncertainty they add to my life.

Preface

This book began as a collection of notes from a class on "predictive science" that I started teaching in 2009 at Texas A&M University. Initially, the course was in the statistics department and taught a group of engineers and statisticians a common body of knowledge around using simulation to make predictions about reality. That initial course had sections on code verification, model validation, and uncertainty quantification (UQ). Each time I taught the course, the UQ section expanded, and eventually the UQ portion became the entirety of the course. This was in response to student feedback and the fact that the research and practice of UQ expanded so much in the intervening years. The content in this work represents what I feel to be a range of topics that gives engineers and physical scientists the crucial knowledge of uncertainty quantification and predictive science. I have tried to include as many examples as possible to give the reader insight on how the methods in the book behave, as well as guidance in applying the methods to other problems.

This book is geared toward readers who are numerically solving mathematical models, often in the form of partial differential equations, that have uncertainties due to the distribution of the inputs, discretization and solver error, and model error. The topics that are covered give the reader the ability, and the motivating reasons, to analyze how uncertainties affect computer simulation and ultimately predictions. A thorough discussion of the landscape and overall setting of uncertainty quantification in the context of simulation-based prediction is given in Chap. 1.

Throughout most of the text, the advection-diffusion-reaction equation is used as a test bed for different UQ methods. This equation in one of its many forms can be found in most engineering and science disciplines so that, I hope, most readers will find examples based on this equation relatable to his or her work. The ideas behind uncertainty quantification can be applied to almost any problem, but having examples that can be directly connected to the reader's experience is more powerful.

In my experience many students do not have a deep enough probability and statistics background to digest all the techniques that are used in UQ. For this reason Part I of this work gives the reader the necessary background in probability and statistics. This goes beyond the basic definitions to include topics such as copulas, Karhunen-Loève expansions, tail dependence, and rejection sampling.

This book includes coverage, in Part II, of the topic of local sensitivity analysis because I feel that is a good place for a novice to begin to understand the overall topic of UQ, and local sensitivities can be useful in reducing the input parameter space. The coverage of local sensitivity goes beyond derivative approximations and estimation of output variance. Using regression techniques, including regularized regression, to estimate first- and second-order sensitivities is included. The topic of adjoint equations as a means to estimate sensitivities can be found in Chap. 6, wherein a concise procedure for deriving adjoints for nonlinear, time-dependent problems is presented.

Part III of this work covers what many would call conventional UQ, that is, the estimation of output uncertainty from parametric, or input, uncertainties. Therein the topics of Monte Carlo, reliability methods, and stochastic projection are covered. The chapter on Monte Carlo, Chap. 7, goes beyond simple random sampling to include Latin hypercube designs (and variants) as well as quasi-Monte Carlo techniques and comparing all the sampling-based methods discussed. Reliability methods are presented in Chap. 8 as an approach to estimate properties of the output using a small number of simulations.

The exposition of stochastic projection and collocation methods, sometimes called polynomial chaos techniques, in Chap. 9 is detailed and gives concrete examples of expansions in several different orthogonal polynomials, as well as details of the quadrature sets needed. In that chapter I take the liberty of defining beta and gamma random variables that are slightly different than the standard definitions to make the expansions much easier to calculate. Chapter 9 also includes discussions of sparse quadrature for multidimensional integration, the use of regularized regression to estimate expansion coefficients, and the stochastic finite element/projection method. The coverage in Chap. 9 is complete and addresses the common complaint from students that polynomial chaos is difficult to apply because of the different definitions of orthogonal polynomials and quadratures. One small downside to this completeness is that there are over 100 numbered equations in Chap. 9.

Part IV demonstrates how surrogate models (sometimes called emulators) can be used to fuse experimental and simulation data to make predictions. Chapter 10 introduces Gaussian process regression as a technique to construct surrogate models. The discussion of calibration and predictive models in Chap. 11 follows that of Kennedy and O'Hagan for the predictive model form but does include the extension to a hierarchy of model fidelities. Chapter 12 also provides the requisite background in Markov chain Monte Carlo and the Metropolis-Hastings algorithm to fit predictive models. The final chapter, Chap. 12, is devoted to handling uncertainties that do not have a distributional nature. This chapter shows how interval uncertainties can be treated and how they affect predictions.

The material in this book can be covered in a single course on uncertainty quantification. I assume knowledge of the standard mathematical content covered in the engineering/physical science undergraduate program. Some knowledge of partial differential equations is assumed, and any topics that I believe would be new

to the reader are introduced gently. The most challenging mathematics is probably in Chaps. 9, 10, and 11. I have attempted to make these topics as uncomplicated as possible without making the techniques seem like opaque, black boxes.

Finally, a note about style. I have tried to make the text of this work not be burdened by an overly pedantic style. I hope that the style does not veer into the realm of being too conversational. My intent is to make the reader feel as though we are discussing the material face to face. Of course in discussion, I often make allusions to topics that are far afield of science and engineering. I have tried to minimize the number of times the reader will be sent to the nearest search engine to look up something, but at the same time, I hope some readers learn about more than just UQ.

Many thanks are in order for making this book possible. I would like to thank Denise Penrose at Springer who managed a project that was long in the making and shepherded drafts of the manuscript through the review process. The feedback of the anonymous reviewers, as well as Martin Frank and Jonas Kusch from Karlsruhe Institute of Technology (KIT), helped improve the work. My engineering colleagues during my time at Texas A&M, Marvin Adams, Jim Morel, and Jean Ragusa, were especially influential in the development of this book. I am also grateful to Bani Mallick and Derek Bingham for many helpful discussions. I would like to thank KIT and RWTH Aachen University for hosting me at points during the preparation of this manuscript, including giving a short course based on Chap. 9 for the AICES EU Regional School in 2016. Finally, this work would not have been possible without the support and help of my wife Katie.

Notre Dame, IN, USA Ryan G. McClarren
July 2018

Contents

Part I
Fundamentals

Part I of this text gives the background in scientific computing, probability, and statistics that will be the baseline for the development of uncertainty quantification techniques. The first chapter deals with the question of how and why we need to understand the uncertainty in computer simulation results and sets the stage for the type of problems that we will solve. The second and third chapters in this part discuss how we will use probability and statistics, with the last chapter giving in-depth discussion of the more advanced statistical tools and concepts necessary to perform uncertainty analyses.

Chapter 1
Introduction to Uncertainty Quantification and Predictive Science

You shall know the truth, and the truth shall make you odd.

—Flannery O'Connor

Since I was a child, I was enthralled with the idea of using a computer to solve problems that I could not with pencil and paper. There is a good chance if you are reading this that you have had a similar experience with the augmented problem-solving ability that computers allow. Most people in computational science have, at one point or another, been frustrated with the limited applicability of the typical toolbox used to solve partial differential equations (e.g., integral transforms, eigenfunction expansions, etc.). The beauty of computer simulation is that any problem can be solved provided you can cast the continuum equations in terms of finite quantities and you have enough computer horsepower at your disposal.

Beyond the fact that computation allows the solution of problems that are intractable by other means, simulation also allows us to probe areas that experimental measurements cannot. No experiment could give you the temperature profile at every point on the surface of a space reentry vehicle or the distribution of neutrons in a nuclear reactor. Solving the equations on a computer gives you this information at the scale one desires and can give insight into the mechanism of a phenomenon in ways that experiments can only suggest.

The ability to show what is going on in an experiment can be extrapolated to make a prediction. It is reasonable to suggest that a computational simulation could tell a researcher what will happen in an experiment that has yet to be performed. Such a request occurs often in terms of design, asking how a new system will perform when it is built. Typically, this exercise uses computation to rule out certain designs, and the candidate designs that pass the computational test are then tested in small scale experiments, before production of the new system takes place. Eliminating some designs will cut down on the possible number of prototypes that need to be built and tested, leading to significant cost and time savings. Using computation in this way is entirely justifiable and reasonable, especially when there is operation history and previous experimental results for systems that are "nearby" the new candidate design. Take the example of an airplane. From my

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0_1

(admittedly) outsider's perspective, the commercial jets produced in the 2000s are not significantly different than those produced in the 1980s in terms of basic aeronautics. A new aircraft design probably shares a lot in common with previous designs and if computational models could adequately predict what happens with those systems, then there is hope that the new designs could have their performance adequately simulated. Of course, this begs the questions: "What does it mean to have a nearby design" and "How do we quantify adequate simulation performance?" We will revisit these topics later.

If simulation for evolutionary design makes sense, provided we define our goal clearly, we can go one step further: we can predict the behavior of a system in conditions where we cannot do a full-scale experiment either for cost, safety, or regulatory reasons. These are the questions that are often the thorniest to answer and have the highest impact. When the space shuttle was damaged by falling debris on a launch, how can we use computation to make statements about the reliability of the craft? The people making decisions about the mission want an answer, but we also need to quantify the uncertainty in our answer. Another case worth mentioning is the question of long-term reliability of a system. Consider a nuclear reactor that was initially designed to last 30 years of continuous operation. What can we say about the safety of the system if its license is extended another 50 years? We obviously cannot do an experiment where a reactor system undergoes 80 years of irradiation at operating conditions without actually operating the system for that long (and even if we did, that would be one sample from the distribution of possible outcomes). We can do small-scale experiments where certain components receive an equivalent dose to decades of radiation exposure, but how can we assemble the experimental data to say that the entire system is safe? How do we state our sense of the risks/uncertainty in any result?

Both of these scenarios are high-impact decisions that must be made without full knowledge of how the system will respond. Lives could be potentially on the line, and we need to make the best decision given the experimental, computational, and theoretical data at hand. We could be "safe" and always answer the question in the negative. In such a case, almost no new technologies would be fielded and arguably life as a whole would suffer. To a large extent, economic growth depends on the development of technology; this has been true since the advent of agriculture; to stop the progress of technology given the tools at our disposal would almost be criminal. A person earning a subsistence wage in a first-world country today has a life that would be envied by the monarchs even two centuries in the past (if only for antibiotics alone).

I contend that having the default "safe" option is not an option at all. I imagine you feel the same way: you have most likely made the decision to travel somewhere by automobile. It is not possible to guarantee that such a trip was safe, but you, perhaps unconsciously, decided the risks were worth the reward. The best answer is to balance the uncertainty in the outcome with the benefit of the risk.

This work *does not* deal with the process of making decisions under uncertainty. That is, we will not deal with policy matters of what is an acceptable risk in a situation. What we will discuss is how to assess the amount of uncertainty in a

prediction based on a simulation. We will call the process of making a credible prediction based on computer simulation and available experimental data *predictive science*.

1.1 The Limits of Prediction

Before embarking on a journey to make predictions with simulations, we will have a short scitament regarding the theoretical limits of how we make predictions and the path that scientific progress has taken to get to the current state of affairs where we believe we can make predictions with an understanding of the uncertainty in those predictions.

The height of predictability, i.e., determinism, can be expressed through a thought experiment. In 1814 Pierre Simon Laplace proposed his "demon". This entity, if it knew the position and momentum of every atom in the universe, could, using Newton's laws, determine the future state of the universe. In this sense, then everything in the future is known. Obviously, quantum mechanics makes the power of the demon impossible because the position and momentum of every atom is not knowable. Moreover, even in the classical sense, the demon's task is impossible, as has been recently demonstrated (Collins 2009; Wolpert 2008). Of course, Laplace's demon represents a rather strong form of determinism.

The thought that we could have strong predictions based on solid physics and mathematics was on strong footing until early in the twentieth century. Humanity believed that the solution to the problems of nature were at hand as evidenced by the triumph of classical physics to predict the behavior of the then-observable universe. Also, mathematicians and logicians embarked on a quest to provide a foundation for all of mathematics. An example of this is Russell and Whitehead's *Principia Mathematica*, which tried to derive mathematical truth from a set of basic axioms. This incredibly dense work takes 379 pages to prove that $1 + 1 = 2$, and this result is accompanied by the comment "The above proposition is occasionally useful."

Breakthroughs in the twentieth century did serve to dampen the ebullience of the era. Heisenberg's uncertainty principle, and the other strange results of quantum mechanics, indicated that, at some level, the best that physics could do was give probabilities of events. The program of Russell and Whitehead was also derailed by Gödel's incompleteness theorem. This theorem says that a complete and consistent set of axioms for all of mathematics is impossible. Gödel does this by using the axioms to derive a liar's paradox: "True proposition G cannot be proved." This theme of the reach of knowledge being circumscribed was also shown in computer science by the work of Turing and others.

Despite the fact that we know there are limits to our knowledge and what we can predict, all is not lost though. We engineer, build, and design systems and they generally work in the way we predict them to. Technology has marched on. We as scientists rely on some, perhaps ineffable, weak form of determinism. In this work, we will look at how we can use our models and calculations for making predictions while being cognizant of the limitations in the predictions.

1.2 Verification and Validation

Verification and validation (V&V) are two processes in computational science that give confidence in the results of a simulation. Successful V&V are essential to performing uncertainty quantification. Approaches to performing V&V are the subject of several books. Three useful examples are Roache (1998), Oberkampf and Roy (2010), and Knupp and Salari (2002).

1.2.1 Code and Solution Verification

Verification is the process of demonstrating that a simulation code solves the underlying mathematical model equations and the characterization of the numerical error. In simpler terms, verification answers the questions:

- Does the code solve the equations it claims to?
- How big is the error?
- How do I expect that error to change as the mesh, time step, etc. changes?

The verification exercise is often a computer science and mathematics exercise. The computer science aspect is evident in the fact that errors, i.e., bugs, in the code slightly alter the solutions and create errors in the simulation. In this regard, a code bug code makes the code solve a different set of equations than intended. The mathematics aspect of verification is showing that the code has a numerical error (i.e., the combination of discretization error, iterative solver errors, etc.) that behaves as expected, based on the knowledge of the accuracy, stability, and other properties of the underlying numerical methods. To demonstrate that the error behaves as expected, one often compares to the code to exact solutions to the underlying equations and shows that the error in the calculation goes to zero in an expected way as the resolution of the simulation is increased.

Also included in the verification process is the exercise known as solution verification. Solution verification attempts to bound and perhaps quantify the numerical error in a calculation. One might imagine that when simulating a large system, calculations may only be done at a handful of resolutions (due to, perhaps, the difficulties of mesh generation or the amount of available time on a machine). Quantifying the error in such a situation may be difficult. In these instances one may turn to convergence acceleration techniques or other estimates such as Richardson extrapolation or single-grid error estimators. Solution verification is an important component of studying uncertainty because the numerical error in a calculation is a source of uncertainty. One needs to know the magnitude of this error to account for its impact. There are technologies, such as goal-oriented adaptive refinement methods, that take care of some of this solution verification as part of the solution process.

1.2.2 Validation

Validation attempts to answer the question of whether the underlying mathematical model is appropriate for the system of interest. In the parlance of our times, validation can be said to answer the question: "Am I solving the correct equations?" Validation is thought to be an endeavor in physics and engineering. This is due to the fact that to perform validation one needs to compare numerical solutions to experiments, and, if the system of interest does not have experimental data, expert judgment to decide if the mathematical models are available is applicable. Beyond the physics/engineering questions, there is an element of the philosophy of science to answer the question of whether the mathematical model is predictive. Validation is necessarily situation dependent: a code that is valid for one system will not necessarily be valid for another. That being said there is no such thing as a "validated code," even if that term is commonly used. This is because one can always come up with a situation where the mathematical model will fail.[1] The best that validation can do is make concrete and narrow statements about the applicability of a mathematical model for a system under a particular circumstance. The range of scenarios where a model has been shown to be valid is known as its domain of validity.

Here is where the opt-quoted aphorism by George Box, "All models are wrong, but some are useful," could be mentioned. Validation answers the question of where a given mathematical model, that is, a simplification of reality, is useful describing physical phenomena.

In bears mentioning that one cannot perform validation without having done thorough verification of the code because it is not possible to make statements about a code's validity unless we know something about the numerical error. This also connects validation to uncertainty quantification because we cannot measure the agreement between simulation and experiment without knowledge of how uncertain the simulation result is.

Where verification is a mathematical and computer science exercise, validation, it can be said, is a scientific endeavor. We have a theory that a particular model or system of equations can explain the phenomena of a real-world situation; proving that the theory is applicable is by no means trivial. Also, because validation answers a scientific question, the methodology of validation differs significantly between the scientific branches. This contrasts with the fact that mathematics is a generic construct, a property that makes the process of verification the same across disciplines: only the equations change. In a validation exercise, one needs to leverage knowledge of the underlying scientific branch, be it physics, engineering, chemistry, biology, economics, or sociology.

[1] This is not just a handwaving argument. There is no unified theory of all the forces in the universe, i.e., we have not uncovered the equations that underlie the universe at all scales. Therefore, any single mathematical model will not be accurate for every problem.

Even a fledgling science student knows that lynchpin of the scientific process is the use of experiments to support hypotheses or to falsify them. Supporting the theory that a given mathematical model describes a phenomena is no different. The problem is the process of comparing experiments to numerical results is not as simple as computing a number and then seeing if it agrees with the experimental measurement.

Unfortunately, experimental data is often lacking or impossible to gather. This predicament is not uncommon. The problem of geologic disposal of nuclear waste is a prime example. We can model the behavior of a repository for nuclear waste using geology, hydrology, and nuclear engineering considerations in an attempt to say whether the waste will contaminate the groundwater, but we cannot do an experiment unless we want to wait 10,000 years (!) for the result of the experiment. In such a situation, often the best we can do is to forthrightly state the assumptions in our model and point by point justify each of these assumptions.

1.2.3 Experiments for Validation

Using experimental measurements to compare with simulation results is the bedrock of model validation. Nevertheless, comparing numerical results with experiment can be exceedingly difficult. Specifying the problem for the numerical code to solve is not a straightforward task. For example, the boundary and initial conditions that are needed to mock up the experimental setup may not be known to enough precision or may not fit into the framework of the code. Care must be taken and a large amount of detail is needed to simulate a given experiment.

The large amount of detail needed to properly specify the experimental configuration in order to simulate the experiment on a computer will often mean that "old" experiments are not suitable for the validation task. Generally, there is not enough detail archived about the experiments, especially in journal publications where economy of space is favored over detailed descriptions of the experiment and long lists of data. It is best to use experiments explicitly designed to validate computational models. That way care can be taken to precisely characterize the experimental setup, provide large amounts of data, and provide detailed estimates of the measurement error.

Another fact that should be considered when thinking about experiments is the fact that experiments rarely report raw data. Rather, experiments often use some conceptual model to process the raw data and produce a result. For example, consider an experiment measures the yield stress of a novel material. That experiment assumes that the materials properties are such that there is one particular value of the yield stress. Also, the yield stress is not measured; other parameters of the measured and then a yield stress is inferred.

1.2.4 Simulation Versus Experiment

For the most part, computer simulations are guilty until proven innocent, in that the burden of proving that a simulation represents reality lies in the hands of the one doing the simulation. On the other hand, an experimental result is often widely accepted as being an accurate picture of reality. Few will question whether the team who completed the experiment properly characterized and accounted for all sources of error. Paraphrasing Roache (1998), the state of play is such that nobody believes the result of a simulation, except the person who performed the simulation, and everybody believes the result of an experiment, except the person who ran the experiment.

The outlook of the experimenter is often proper, that is, it is naive to assume that the result of a single experiment is the final word on a specific phenomenon or system. This should also be the outlook of the computational scientist that is attempting to validate a particular model: one number from one experiment should not make or break a model.

1.2.5 Small-Scale Experiments

The small-scale experiments mentioned above test the simulation performance on a particular aspect of the system simulation. For example, if the system under consideration couples heat transfer, fluid flow, and chemical reactions, there are several small-scale experiments possible. One type is a single-physics experiment, where a particular physics phenomenon is observed and measured in isolation, for example, one could field a heat transfer experiment and a separate fluid-flow experiment. Then the simulation for that single "physics" is compared with the experiment. If each of the single-physics experiments can be reproduced with the simulation code, we at least have the hope that the simulation will perform in the coupled case.

The other type of small-scale experiments involve a system similar to the full system but modified to make the experiment possible. For instance, the system could be made smaller, the heating rate could be lower, and the materials used could be surrogates for the actual materials. These experiments test the simulation's ability to reproduce phenomena in a coupled system as similar to the full system as possible. One example of a small-scale experiment involves the heat transfer in a nuclear reactor. In a typical reactor, each fuel assembly could be generating megawatts of power. A small-scale experiment may involve a fuel assembly containing nonnuclear material (a surrogate material) and heated electrically with kilowatts of energy (a scaled-down system load). These choices are made for several reasons; building the experiment with nuclear material like uranium would require much more effort in terms of safety and regulatory approval and might limit the types

of diagnostics available. The lower-power level is required because megawatts of electricity are difficult to get except at specialized facilities.

1.3 What Is Uncertainty Quantification?

Uncertainty quantification (UQ) attempts to answer the question of how uncertain is the result of the computation (National Academy of Science 2012). Every simulation has inherent uncertainties in the input. These could be the dimensions of pieces of the system due to manufacturing tolerances, the constitutive properties of materials, or a lack of knowledge of the ambient conditions. Propagating these uncertainties to the result of the simulation is one aspect of UQ. Beyond such propagation of input uncertainties, also called parametric uncertainties, UQ attempts to include knowledge of numerical error (perhaps from solution verification) and the mathematical model error.

UQ is often considered an exercise in statistics because of the probabilistic nature of uncertainty and the typically large number of uncertainties in a simulation. It should not be a purely statistical exercise, however, because the results of statistical models should respect the physics of the problem. Including physical knowledge in the statistics of the problem provides better estimates of uncertainty due to the fact that the physics considerations can constrain the distribution of quantities of interest.

UQ is not just a single-step process; there are several stages that must be completed to defensibly, reliably, and accurately quantify the uncertainty in the simulation of a physical system. Each step could be the subject of a book on its own and is the subject of active research. The steps of uncertainty quantification are:

1. Identifying quantities of interest,
2. Identifying and modeling the uncertainties in the problem inputs,
3. A down-selection of the uncertain inputs,
4. The propagation of uncertain inputs through the simulation,
5. Determining how the uncertainties will affect predictions.

The first step in UQ, though often not thought of as a difficult task, is the selection of the quantities of interest (QoIs), that is, what are the metrics by which we will assess a given system or design. Typically, a QoI is a scalar number, such as, for example, the maximum temperature in the system, the failure stress of the structure, etc. These quantities are typically expressed in terms of a function of the solution of a set of model equations (e.g., partial differential equations, algebraic relations, etc.). For example, the maximum temperature in the system could be determined by applying a maximum function to the solution of the heat equation. Another common situation requires taking an integral over time and/or space of the solution of the model equations. This might be the case if the QoI was the average of the temperature in the system during a certain time.

To proceed with uncertainty quantification, we need to be able to make statements about the input uncertainties. We cannot simply say that an input x is uncertain, we need to say how it is uncertain. Can we give it a probability distribution? Is it correlated with another input? Do we know basic statistics of the input, e.g., mean and variance? Answering these questions can be difficult. For instance, if we have 1000 observations of an input parameter x that all fall in the range $[a, b]$, does that mean x can never take a value outside of this range? The answer to this question will have an impact on the resulting uncertainty in the QoI.

Uncovering and identifying the uncertainties in a simulation often asks questions that have not been asked before. Some of the data used in simulations is of unknown origin or based on approximate models. In these cases it may be difficult to characterize the uncertainty. Additionally, questions about numerical error and model error can also have an impact on a simulation and should be considered.

The task of identifying uncertainties will often uncover many uncertain parameters in the simulation. Typically, as there are more uncertain parameters, more simulation results are required to quantify the uncertainty due to each parameter. Furthermore, in situations where the simulations are expensive in terms of computer time, one cannot afford to run a large number of simulations to characterize the impact of all of the uncertain parameters. In this scenario it can be useful to judiciously remove some of the uncertain parameters if they will have a small impact on the QoIs. This can be done by estimating local sensitivities and using other approximations, such as active subspace projection (Constantine 2015). Of course, the selection of parameters to remove from the simulation is a source of uncertainty in the uncertainty quantification.

Given the uncertain parameters that one wants to analyze, the next step is to actually quantify the uncertainty in the QoIs from them. There are several approaches to do this that span the spectrum from simple methods that are quick and approximate or slow, but more robust, methods. The choice will often depend on what the analysis will be used for. If one wants to judge whether a standard safety factor is applicable to a system, reliability methods can quickly assess how close a system is to "failure." If the distribution and extreme values of the QoI are important, for instance, if there are potentially low-frequency but high-impact events possible in the system as in, for example, nuclear reactor safety, then more sophisticated methods may be required, such as polynomial chaos or Monte Carlo methods.

At this point, many would consider the uncertainty quantification process complete, yet the application of the knowledge learned about the system is an important consideration. Given that one knows how the QoI can vary with respect to inputs, what does that mean for making predictions? Additionally, we need to re-evaluate past experimental data in light of the uncertainties, as well as determine how likely the simulations are to be accurate for a prediction on a different system. Addressing these issues requires deeper understanding of the simulations and the input uncertainties. To answer these questions, when there is a limited budget of time or other resources for performing more simulations, it is often necessary to construct an approximation to the simulation, known as an emulator or a reduced-

order model. Furthermore, the types of input uncertainties affect the interpretation of the prediction.

The process and science of UQ is more than just putting error bars on the simulation. It requires inquisitiveness to ask questions about the impact of results, physical and engineering intuition to know how to interpret results, and the humility to understand that not all questions can be answered with certainty.

While this work focuses primarily on using probability theory to estimate uncertainties, there are other mathematical approaches that we do not cover. These include fuzzy logic and worst-case analysis. Halpern (2017) discusses these alternate approaches.

1.4 Selecting Quantities of Interest (QoIs)

As mentioned above, one of the necessary steps in the UQ process is selecting quantities of interest. These QoIs are necessary to perform many of the subsequent UQ tasks. The fact that the QoIs are a finite number of scalar values may be counterintuitive to many computational scientists. One of the benefits of computational simulation is that we can usually get the solution "everywhere" in the problem. That is, we can know the solutions to the underlying model equations throughout the problem domain in space, time, etc. Nevertheless, the problem of discussing the uncertainty of a function, or in more technical terms a distribution of functions, is a much more difficult proposition. In a sense, the uncertainty in a function is the same as having an infinite number of QoIs. As we shall see, handling a small number of QoIs is difficult enough.

To illustrate the selection of a QoI, we will introduce a model problem that we will use throughout our study. This is the advection-diffusion-reaction equation. Here we are interested in a function $u(\mathbf{r}, t)$ that is governed by the following partial differential equation on the spatial domain V

$$\frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u = \nabla \cdot \omega \nabla u + R(u), \qquad \mathbf{r} \in V, \quad t > 0. \tag{1.1}$$

with boundary and initial conditions given by

$$u(\mathbf{r}, t) = g(\mathbf{r}, t) \quad \mathbf{r} \in \partial V, \qquad u(\mathbf{r}, 0) = f(\mathbf{r}). \tag{1.2}$$

In this model, $\mathbf{v}$ is the speed of advection in each direction of u, ω is a diffusion coefficient, and $R(u)$ is a reaction function. We choose this model because it is a simplified model for many physical processes. For instance, if u is a temperature and $R(u) = 0$, then we have a heat equation that includes convection via the $\mathbf{v} \cdot \nabla$ term and heat conduction via the diffusion term. Other possible ways to use this model is to treat simplified problems in particle transport, contaminate dispersion, fluid-flow problems, and damped, mass-spring systems.

In the situation where Eq. (1.1) is an adequate model for our physical system, we might be interested in the following quantities:

- The maximum value of u inside a given time range $[a, b]$:

$$\max_{\mathbf{r},\, t\in[a,b]} u(\mathbf{r}, t),$$

- The average value over a particular region of space, D, and time range $[a, b]$:

$$\frac{1}{b-a}\frac{1}{|D|}\int_D d\mathbf{r}\int_a^b dt\, u(\mathbf{r}, t).$$

  Here, $|D|$ is the volume of the region D.
- The total reaction rate in the system over a given time range $[a, b]$:

$$\int_V d\mathbf{r}\int_a^b dt\, R(u(\mathbf{r}, t)).$$

- The outflow of u from the system over a given time range $[a, b]$:

$$\int_{\partial V} dA\int_a^b dt\ (\mathbf{v}\cdot\mathbf{n} - \mathbf{n}\cdot\omega\nabla)\, u(\mathbf{r}, t),$$

  where $\mathbf{n}(\mathbf{r})$ is the outward normal on ∂V.

These examples can be readily generalized for many problems and scenarios. As a generic way of writing a QoI, it is often possible to write a QoI, Q, as

$$Q = s(u) + \int_V d\mathbf{r}\int_0^T dt\, w(u, \mathbf{r}, t). \tag{1.3}$$

Here, $s(u)$ is a function that maps the output of $u(\mathbf{r}, t)$ to a scalar, such as the max function we saw above, and $w(u, \mathbf{r}, t)$ is a weight function. As an example, if the QoI is the reaction rate over a range of time, then $s(u) = 0$ and

$$w(u, \mathbf{r}, t) = \begin{cases} R(u) & t \in [a, b] \\ 0 & \text{otherwise} \end{cases}.$$

Alternatively, we were interested in the maximum value of u; then $s(u)$ would be a maximum function.

The upshot of this discussion is that QoIs can be very general and can typically written in the form given in Eq. (1.3). If the domain of u includes more than just space and time, then the integrals will include those added dimensions.

1.5 Types of Uncertainties

There are two main classes of uncertainties in a problem. These are not necessarily two distinct classes as some uncertainties could be classified into either category. The nature of the uncertainty does impact how we want to treat the results of the analysis, as we will demonstrate. Further discussion of these concepts can be found in Der Kiureghian and Ditlevsen (2009).

1.5.1 Aleatory Uncertainties

Aleatory uncertainties come from the inherent randomness of a system. The term derives from the Latin *aleator* for dice player, and this provides a good mental model for these uncertainties. If we consider every replicate of an experiment or system fielded, there will be slight differences due to issues such as manufacturing tolerances, ambient conditions (e.g., weather), and other randomness.

A property of aleatory uncertainty is that the randomness can be described by a distribution. For example, given a process that manufactures a part of the system of interest, there will be a distribution of sizes of the component. By looking at realizations from the manufacturing process, one could fit a distribution for the size of the component.

Another example of an aleatory uncertainty would be the position of aggregate (i.e., the rocks) in concrete. The distribution of the position and the size of the rocks in the concrete will change from concrete sample to concrete sample, and one could obtain a distribution for the position, shape, size, etc. for the aggregate.

1.5.2 Epistemic Uncertainties

Epistemic uncertainties arise from the lack of knowledge about a system. Oftentimes, these uncertainties are due to an approximate model for the system, but they can also arise from numerical error. In both cases, it is likely there are errors made from the approximations, and we do not know how large those errors are *and* those errors are not described by a probability distribution. In many cases the best we can do is bound the uncertainty, but then we are dealing with intervals and not probabilities.

The epistemic uncertainty could arise from approximations in the analysis of a system. For instance, when an analyst prescribes a distribution for an aleatoric uncertainty based on a set of samples from the distribution, an error likely arises. This uncertainty arises from a lack of knowledge from the true distribution of that input.

Other sources of epistemic uncertainties are the unknown uncertainties or the uncertainties that we do not treat as uncertain. These are sometimes called the unknown unknowns, indicating those uncertainties in a system that have not been identified and are therefore excluded from the analysis.

As an illustration of epistemic uncertainty, consider a car braking system where there is a "0.1% chance of failure." The implications of this 0.1% number depend on the type of uncertainties in the estimate. In one case, the uncertainties are due to aleatory uncertainty: the system performance is determined by variability in manufactured parts. The analysis indicates that 0.1% of manufactured parts made will fall outside the tolerable range and will fail due to inherent uncertainties in the manufacturing process. The result is the 0.1% of the systems will fail.

However, there are uncertainties in the failure temperature in the brake system. Based on the possible range of the failure temperature, about 0.1% of that range will lead to system failure. What this means is that if the failure temperature for the system is in that 0.1% range, then all the brakes will fail. In other words, the 0.1% chance of failure means that there is a 0.1% chance that all the brakes fail.

As this example makes clear, the types of uncertainties affect the interpretation of the output. Also, because epistemic uncertainties do not have an associated probability distribution, there are fewer mathematical tools to deal with the lack of knowledge. We will show, however, that we can take them into account in real systems using specialized techniques.

As mentioned above, there are instances where a uncertainty could be considered epistemic or aleatory depending on the context. For example, when we speak of the uncertainty in a physical quantity, there may be a model implied in the idea of that quantity to begin with. One example is an equation of state model that assumes a gamma-law gas. The parameter γ may have a distribution depending on some ambient conditions, but there is no correct γ because it is derived from a simplified model of how gas molecules behave. In this case we may say that part of the uncertainty is aleatory (the value of γ that we use based on experimental data for the gas) and part is epistemic (the uncertainty due to using the simplified model).

1.6 Physics-Based Uncertainty Quantification

An important consideration in uncertainty quantification is where uncertain data comes from. This will be a part of the process of identifying uncertainties, but it is important to think about the origin of data. For example, in some simulations an input quantity is the equation of state of the material (e.g., a relation between pressure, temperature, and internal energy of the material). As used in codes, the equation of state can be represented by a large look-up table that could have thousands of entries. That does not mean there are thousands of uncertain parameters, however. The equation of state table is likely to be generated by a combination of experimental measurements, theoretical models, or simulations. Each of these components of the equation of state table will have its own uncertainties. For

example, the experiments will have uncertainties in the measurement, the theoretical models have parameters that will be uncertain, such as a gas constant, and the simulations will also have uncertainties. The sum total of the uncertainties in these components is likely to be much smaller than the number of parameters in the table. Therefore, the true dimension of the uncertainty is not based on the equation of state table, but on the physics behind the table.

This is an example of physics-based uncertainty quantification, and it is an important illustration of the power that knowledge of the simulation and the processes behind it are useful to the uncertainty quantification practitioner. There are also many other ways that domain expertise can inform a UQ study. With knowledge of the properties of the inputs and QoIs, the UQ process can be tailored to the situation and be more efficient and more accurate. For instance, if a parameter is known to be strictly positive, that will influence the type of distribution it can be. Also, if a QoI cannot be larger than a given amount, the UQ procedure should respect that.

These examples of physics-based uncertainty quantification indicate that the expertise that the scientist has cannot be forgotten when performing a UQ study, or, to put it the other way, the UQ expert is most effective when expert knowledge is combined with domain expertise from a scientist or engineer. Furthermore, this type of domain knowledge is not limited to physics; one could easily speak of chemistry-based or biology-based UQ or any number of other technical fields.

1.7 From Simulation to Prediction

Given the results of a UQ study, that is, knowledge of the QoI and its uncertainty, the next question is what does one do with that information. There are several scenarios that serve as the bridge between understanding of parametric uncertainties and the application of that knowledge to making a prediction. As a way to highlight how this might be used, we will detail some examples of predictive science in action.

1.7.1 Best Estimate Plus Uncertainty

The term best estimate plus uncertainty is used by regulators in nuclear reactor certification around the world. The term refers to the use of simulation codes and models that have been demonstrated to be applicable to the system and conditions under question. The values of the QoI (typically the probability of failure) are quoted at the most likely values of the uncertain parameters, this is the best estimate part of the equation, and then a confidence interval around that estimate, the uncertainty part. This confidence interval is estimated by sampling uncertain parameters, running a simulation, looking at the distribution of the outputs, or building an approximate model based on the outputs.

1.7.2 Quantification of Margins and Uncertainties

Quantification of Margins and Uncertainty (QMU) is concerned with making decisions on whether a system will perform as expected given the performance margin built into the system and the uncertainties in a simulation. In the simplest illustration, we have a system where a performance metric can be in the range $[q, q + M]$, i.e., there is a margin of M. Then from a best estimate plus uncertainty study, we have a range of simulated system outputs, U. Then, based on the ratio M/U, as well as subject matter expertise (i.e., expert judgment), and small-scale experiments that test system components, the decision is made whether the probability of failure of the system is tolerable.

1.7.3 Optimization Under Uncertainty

Designing a system while considering the uncertainties in simulations is known as optimization under uncertainty. In this exercise, one wants to tune a system's performance by adjusting inputs, but taking into consideration the fact that one does not know precisely what the system performance will be. Unique problems arise in this type of optimization. It is possible that the global maximum (i.e., nominally the best design) has a larger uncertainty than a lesser maximum that has a smaller uncertainty. In other words, if two designs give quantities of interest of q_1 and q_2 with $q_1 > q_2$, but the range of performance when considering uncertainty is $\pm 10\%$ for the design giving q_1 and $\pm 1\%$ for the other design, the second design may be a better choice if the worst-case performance is more important than the optimal performance.

1.7.4 Data-Driven Experimental Design

One of the outcomes of an uncertainty quantification study could be which uncertainties in the inputs or models lead to the largest fraction of the uncertainty in the quantities of interest. With this knowledge the investment in additional experiments, higher fidelity models, additional computer resources, etc. can be prioritized. Making quantifiable statements about where uncertainty in our QoIs comes from and how to reduce them is a powerful result. If an engineer can say 80% of the uncertainty in our system's performance is due to uncertainty in the melting point of a component, then we know that improving the knowledge of that melting point will result in a large decrease in uncertainty in the system performance. This is an important, but often overlooked, benefit of a rigorous uncertainty study.

Chapter 2
Probability and Statistics Preliminaries

Stars were falling across the sky myriad and random, speeding along brief vectors from their origins in night to their destinies in dust and nothingness.

—Cormac McCarthy, *Blood Meridian, or the Evening Redness in the West*

We will need some definitions from probability theory as well as a smattering of statistics nomenclature and definitions. This chapter can be safely skipped by those with familiarity with those subjects. We will also set the stage for some of our notation in this chapter as well.

In this chapter and the next, we will use the convention of denoting a random variable by a capital letter, e.g., X, and a *realization or sample* from that random variable using the lower case of the same letter, in this case x. In other words, x is a realization of random variable X. Later we will at times drop this convention when it is convenient and does not lead to confusion.

2.1 Random Variables

2.1.1 Probability Density and Cumulative Distribution Functions

The probability density and cumulative distribution functions are key pieces of information about a random variable. Sometimes we know these functions, for instance, when we say an input to a code has a normal distribution, and other times,

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-319-99525-0_2) contains supplementary material, which is available to authorized users.

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0_2

for example, a QoI, we would like to determine these functions. In either case, we will need to know how the two are related and the key properties of each.

For a given real random variable $X \in \mathbb{R}$, the cumulative distribution function (CDF) is defined as

$$\begin{aligned} F_X(x) &= P(X \leq x) \\ &= \text{The probability that the random variable } X \text{ is less than or equal to } x. \end{aligned} \tag{2.1}$$

Oftentimes, we will leave out the subscript on F when it is clear what random variable we are referring to. One of the uses of the CDF is to find the probability that a random variable is between two numbers. From the above definition, it is straightforward to see that we can find the probability that X is between a and b via subtraction:

$$F_X(b) - F_X(a) = P(a < X \leq b). \tag{2.2}$$

In this equation we note that the probability is strictly greater than a and less than or equal to b. This comes from the definition of the CDF that we used. Based on the fact that a probability must be in the closed interval [0, 1] we assert that

$$F_X(x) \in [0, 1].$$

Also, since X is a real number we know that

$$\lim_{x \to \infty} F_X(x) = 1, \qquad \lim_{x \to -\infty} F_X(x) = 0.$$

These relations are equivalent to saying that X will take some value between negative and positive infinity. There is one more property of the CDF that we need, namely, that the CDF is nondecreasing. One way to state this is to say

$$F_X(x + \epsilon) \geq F_X(x) \qquad \text{for } \epsilon > 0.$$

In other words as x increases, the probability that X is less than or equal to x cannot go down. We will show some examples of CDFs later.

If X is a continuous random variable, that is, X can take any value on the real line or on some interval of the real line, we define the probability density function (PDF) as

$$f(x) = \frac{dF_X}{dx}. \tag{2.3}$$

Because $f(x)$ is a density, if we multiply it by a differential volume element dx, we get

$$f(x)dx = \text{ probability that } X \text{ is within } dx \text{ of } x.$$

We can "invert" the definition of the PDF to get the CDF in terms of the PDF:

$$F_X(x) = \int_{-\infty}^{x} f(x')\,dx'.$$

Following this line of thinking further, we deduce that the probability X is between a and b is given by

$$P(a < X \leq b) = \int_a^b f(x)\,dx = F_X(b) - F_X(a).$$

Also, using the limits of $F_X(x)$,

$$\int_{-\infty}^{\infty} f(x)\,dx = 1.$$

We note here that it is possible for the density to be undefined for a given random variable. This can occur, for example, when the CDF is not differentiable.

As an example of the PDF and CDF of a distribution, consider the normal distribution (also known as a Gaussian distribution). This distribution has two parameters, $\mu \in \mathbb{R}$ and $\sigma > 0$. As we will see later, these correspond to the mean and standard deviation of the distribution. A random variable X that is normally distributed has a PDF given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{2.4}$$

One can show that the CDF is given by

$$F(x) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right) \tag{2.5}$$

where

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\,dt$$

is the error-function. When X is a normally distributed random variable with parameters μ and σ, we denote this as $X \sim \mathcal{N}(\mu, \sigma)$.

In Figs. 2.1 and 2.2, we show the PDF and CDF for a normal distribution with different values of μ and σ. Notice that the PDF is highest at $x = \mu$ and the PDF is symmetric about μ. Also, the parameter σ controls the width of the PDF with higher values making a wider PDF. From the CDF we see that $F(x)$ is equal to 0.5 at $x = \mu$, and the smaller values of σ have a steeper increase in $F(x)$. All of these features could be deduced from the definitions of the PDF and CDF.

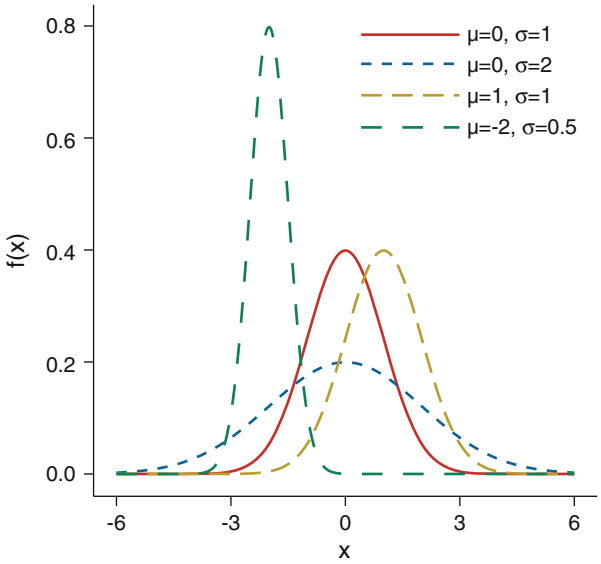

Fig. 2.1 Probability density functions for a normally distributed random variable with different values of μ and σ

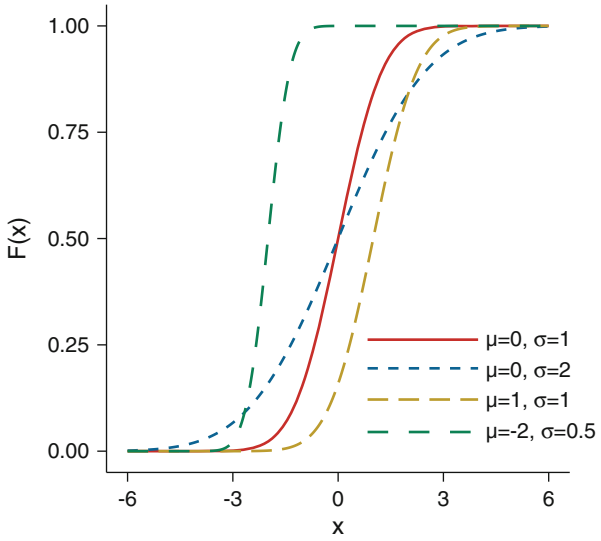

Fig. 2.2 Cumulative distribution functions for a normally distributed random variable with different values of μ and σ

A normal distribution with $\mu = 0$ and $\sigma = 1$ is called the *standard* normal distribution and the PDF of this case is given by $\phi(x)$ and the CDF is written as $\Phi(x)$. Also, any normal random variable can be written in terms of a standard normal. If $X \sim \mathcal{N}(\mu, \sigma)$, then

$$z = \frac{x - \mu}{\sigma}, \tag{2.6}$$

will create a random variable $Z \sim \mathcal{N}(0, 1)$.

2.1.2 Discrete Random Variables

For discrete random variables, that is, a random variable that only takes on a countable number of values, we cannot use a probability density function because it does not make sense to talk about a differential volume element. Instead we define the probability mass function (PMF) for a discrete random variable as

$$f(x) = P(X = x) = \text{ the probability that } X \text{ is exactly equal to } x. \tag{2.7}$$

The notation is being somewhat abused by having both the PDF and probability mass function use f. Nevertheless, by the context it should be clear which we mean, and in practice this is a distinction without a difference if we think of the probability mass function as a sum of Dirac delta functions, that is, a function that is nonzero only at a single point and has a well-defined definite integral. For the CDF of a discrete random variable, instead of an integral, we have a sum:

$$F_X(x) = \sum_{s \in S} f(s), \tag{2.8}$$

where S is the set of all possible values of X less than or equal to x.

An important example of a discrete random variable is the Bernoulli distribution, named after Jacob Bernoulli who developed it in his work *Ars Conjectandi* (Bernoulli 1713). This distribution is simple but useful. It involves a random variable X that can take on two values, 0 and 1, with the probability of $x = 1$ being p. That is the PMF

$$f(x) = \begin{cases} p & x = 1 \\ 1 - p & x = 0 \end{cases}. \tag{2.9}$$

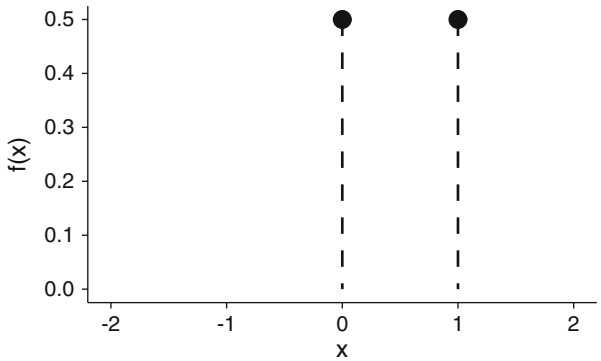

Fig. 2.3 Probability mass function for a Bernoulli distributed random variable $p = 0.5$

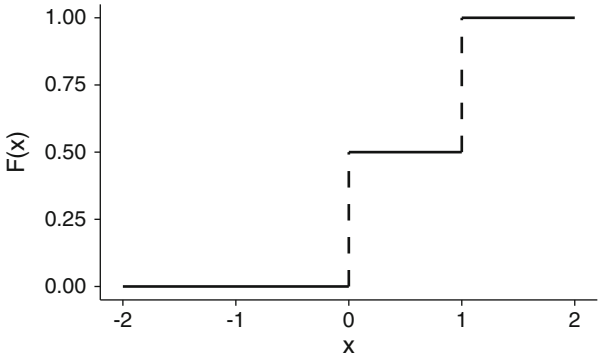

Fig. 2.4 Cumulative distribution function for a Bernoulli distributed random variable $p = 0.5$

The CDF can be easily shown to be

$$F_X(x) = \begin{cases} 0 & x < 0 \\ 1 - p & 0 \leq x < 1 \\ 1 & x \geq 1 \end{cases}. \tag{2.10}$$

If the random variable is a fair coin, then p is 0.5 and we can (arbitrarily) choose a flip that lands on heads as $x = 1$ and a flip that lands on tails as $x = 0$. The PMF and CDF for a Bernoulli distributed X with $p = 0.5$ is shown in Figs. 2.3 and 2.4. Notice the "stair-step" shape of the CDF because the probability that x is less than or equal to a given number "jumps" when crossing 0 and 1.

2.2 Expected Value

It is common to express properties of a random variable in terms of particular moments of its PDF or PMF called expected values. The expected value (or expectation) of a function $g(x)$ is denoted as $E[g(X)]$ given by

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f(x)\, dx \tag{2.11}$$

The expected value is a weighted average of $g(x)$ where the weighting function is the PDF (or PMF).

An important special case of the expected value is the mean which is the expected value of x. It is often denoted as μ

$$\mu = E[X] = \int_{-\infty}^{\infty} x f(x)\, dx. \tag{2.12}$$

In common parlance, the mean is the value of X one would "expect" when drawing a random variable. In many cases this is true. For example, if $X \sim \mathcal{N}(\mu, \sigma)$, then X is normally distributed. The mean of X is then

$$E[X] = \int_{-\infty}^{\infty} \frac{x}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx = \mu. \tag{2.13}$$

The above relation can be shown by making the substitution $u = x^2$ in the integral. Equation (2.13) says that μ is the mean of the distribution. It is also true that μ is the maximum value of $f(x)$ and therefore the most likely value of X.

The mean is not always the most likely value of a random variable, in fact it may not even be a possible value of X. Consider the Bernoulli distribution, the mean of this distribution is

$$E[X] = \int_{-\infty}^{\infty} x f(x)\, dx = 0 \cdot (1-p) + 1 \cdot p = p. \tag{2.14}$$

Therefore, mean (or expected value of X) is p when X can only take the values of 0 or 1. The mean is still useful in this case; we just cannot interpret it as the most likely value.

An old saying about judging a random variable by its mean goes something like this: if I put my head in the oven and my feet in ice water, my mean temperature is just right. In other words, the mean does not tell us everything about the random variable: do not try to walk across a river that has an average depth of 1 m.

2.2.1 Median and Mode

There are two useful properties of the distributions that are not related to the expected value: the median and mode. The median is the point at which the CDF is equal to one-half, i.e., $F(x) = \frac{1}{2}$. This is a useful quantity because it indicates the point that splits the random variable into two equal parts: in the limit of an infinite number of realizations, half will be above the median and half will be below the median. This is not true of the mean. Also, the median is less influenced by outliers.

The mode is the point which the PDF takes its maximum value. Therefore it is the most likely value of the distribution. A distribution with a single mode is said to be unimodal.

2.2.2 Variance

The expected value of $(x-\mu)^2$ is called the variance, and often written in shorthand as σ^2. It is worth noting that the variance can be expressed in terms of the mean and $E[X^2]$ via

$$E[(X-\mu)^2] = E[X^2] - 2E[\mu X] + E[\mu^2] = E[X^2] - \mu^2.$$

In this relation, we used the fact that $E[X] = \mu$, $E[\mu X] = \mu^2$, and $E[\mu^2] = \mu^2$. One can interpret the variance as the average squared difference between a random variable and its mean. The larger the value of the variance, the more likely values are away from the mean. The square root of the variance is called the standard deviation, σ. The standard deviation is useful because it will have the same units as X, whereas the variance has the units of X^2.

For a normally distributed random variable $X \sim \mathcal{N}(\mu, \sigma)$, the variance of X is σ^2. The fact that larger values of σ^2 correspond to values away from the mean being more likely can be seen in Fig. 2.1. In that figure those curves with larger values of σ have much wider curves. For the Bernoulli distribution, the variance can be shown to be $p(1-p)$. Therefore, the maximum value of σ^2 for the Bernoulli distribution is $0.5^2 = 0.25$ and occurs when $p = 0.5$.

2.2.3 Skewness

The mean and the variance are related to the expectation of X and X^2, respectively. The skewness, γ_1, is related to the third moment of $f(x)$, that is, the expected value of X^3:

$$\gamma_1 = \frac{E[(X-\mu)^3]}{\mathrm{Var}(X)^{3/2}}. \tag{2.15}$$

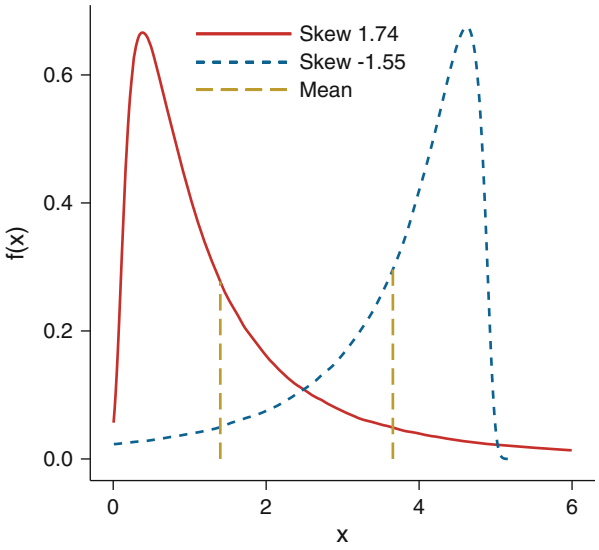

Fig. 2.5 The PDFs of two distributions demonstrating positive and negative skewness. Notice that for positive skewness in this case the peak of the distribution is to the left of mean and for negative skewness it is to the right

The skewness tells us something about the symmetry of the distribution about the mean. The skewness can be counterintuitive because a distribution with positive skew may look as though it is leaning to the left or negative direction.

As illustrated Fig. 2.5, the skewness tells us how the distribution goes to zero away from the mean when the distribution has a single maximum (a unimodal distribution). For this type of distribution, a negative skewness tells us the distribution goes to zero more slowly to the left of the mean, whereas a positive skewness says the opposite. The normal distribution has a skewness of 0 because it is symmetric about the mean.

2.2.4 Kurtosis

Next on the list of properties of a distribution is the excess kurtosis (usually just referred to as the kurtosis) which is a measure of "tail fatness" for a distribution. The kurtosis, $\text{Kurt}(X)$, is related to the fourth moment of a random variable's PDF and is defined as:

$$\text{Kurt}(X) = \frac{E[(X-\mu)^4]}{\sigma^4} - 3. \tag{2.16}$$

The minus three is included so that a normal distribution has a kurtosis of 0. The definition of the kurtosis is such that for a unimodal distribution, the slower the

PDF approaches zero as one moves away from the mode, the higher the kurtosis will be. Another way of thinking about it is that the sign of the kurtosis tells you if the distribution has heavier tails than a normal distribution (positive kurtosis) or if it has thinner tails than a normal distribution (negative kurtosis). There are also fancier names for these cases. A distribution that has negative kurtosis is said to be platykurtic from the Greek *platy*[1] for "flat", whereas a positive kurtosis indicates a leptokurtic distribution from the Greek word *leptós* meaning narrow.[2] A distribution with zero kurtosis is mesokurtic.

As an example we look at a uniform distribution, a normal distribution, and the logistic distribution in terms of kurtosis. A uniform distribution has a PDF that is uniform over a finite range:

$$f_{\text{uni}}(x) = \begin{cases} \frac{1}{b-a} & x \in [a, b] \\ 0 & \text{otherwise} \end{cases}. \tag{2.17}$$

A uniform distribution over the range $[a, b]$ is written as $X \sim \mathcal{U}(a, b)$. The kurtosis of a uniform distribution is $-\frac{6}{5}$ and a variance of $\frac{1}{12}(b-a)^2$. We already noted that the definition of kurtosis we are using defines a normal distribution as having a kurtosis of zero. The logistic distribution's PDF is given by

$$f_{\text{logistic}}(x) = \frac{1}{4s}\text{sech}^2\left(\frac{x-\mu}{2s}\right), \tag{2.18}$$

where s is a parameter that acts in a similar way to the standard deviation in a normal distribution. The variance of a logistic distribution is $\frac{1}{3}s^2\pi^2$ and its kurtosis is 6/5. To compare these distributions, we will look at each with a variance of 1. We show the three on the same plot in Figs. 2.6 and 2.7. The uniform distribution has a kurtosis of $-\frac{6}{5}$ (playtkurtic) and demonstrates this with its flat shape that approaches zero very quickly once we look far enough away from the mean. The logistic distribution is more peaked than the normal and has a positive kurtosis of $\frac{6}{5}$ (leptokurtic). In Fig. 2.6 we see the relative flatness and peakedness of these distributions. When we zoom in on the tails above $x = 3$, that is, more than 3 standard deviations from the mean, we see that the leptokurtic distribution has a higher probability density than the normal distribution. This means that for the logistic distribution, one is more likely to have the random variable take on "extreme values" far outside the mean.

[1] To remember this one can think of a duck-billed platypus having a flat bill, or, for the animal taxonomy aficionado, the name of the phylum of flat worms, *Platyhelminthes*.

[2] There does not exist a great mnemonic for *leptós*, unfortunately. Leptons are small (i.e., narrow) mass subatomic particles, but one does not typically think of them as narrow. Interestingly, the word *leptós* can be found in Mycenaean Greek documents written in Linear B, one of the oldest recorded forms of Greek.

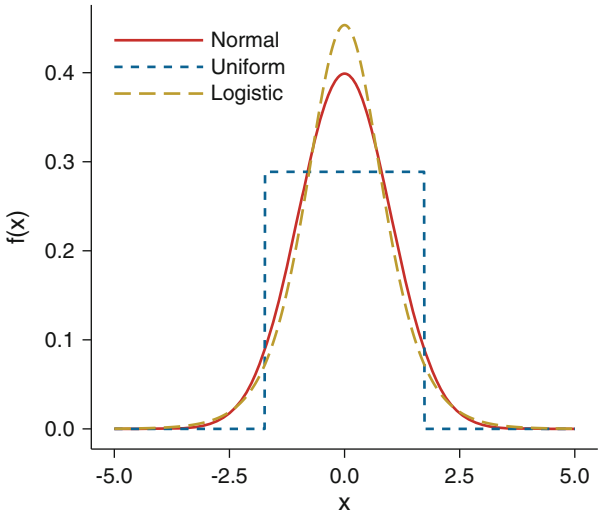

Fig. 2.6 PDFs for a uniform, normal, and logistic distribution all with mean 0 and variance 1

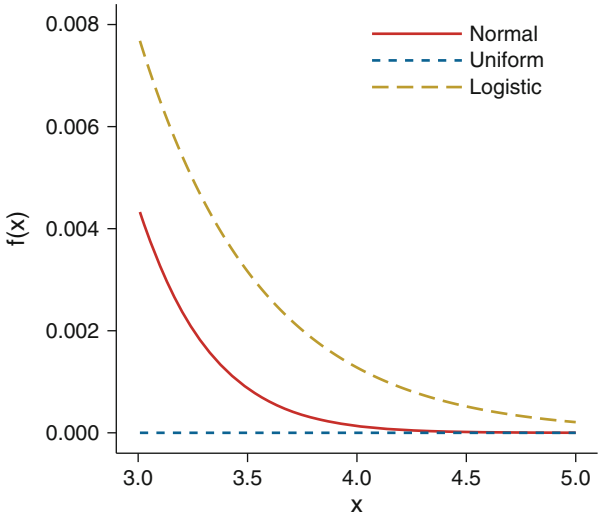

Fig. 2.7 Detail of Fig. 2.6 where we see that for the logistic distribution, one is more likely to have extreme values (greater than 3 standard deviations from the mean) than a normal distribution

2.2.5 Estimating Moments from Samples

Given a number of samples, or realizations, of a random variable, it is useful to estimate what the moments and other quantities of the underlying distribution are. Using this knowledge one can then approximate the probability distribution of the

random variable. The moments are integrals over the probability distribution. To estimate these quantities, we rely on the näive estimator:

$$E[g(X)] = \int_{-\infty}^{\infty} dx\, g(x) f(x) \approx \frac{1}{N} \sum_{i=1}^{N} g(x_i), \tag{2.19}$$

where x_i is a sample from the PDF $f(x)$ and N is the number of samples. In other words, the expected value of $g(x)$ is approximated by the average value of $g(x_i)$. Therefore, an estimate of the mean of the PDF can be estimated via the approximation:

$$\mu \approx \frac{1}{N} \sum_{i=1}^{N} x_i \equiv \bar{x}. \tag{2.20}$$

The notation $\bar{x}$ is used for the estimate of the mean and is known as the sample mean or sample average. This estimate of the mean will have an error based on the randomness of the samples involved. One can show, via the central limit theorem, that the error is the estimate of the mean is proportional to $1/\sqrt{N}$ as $N \to \infty$.

The variance estimate is similar in that we are trying to estimate an integral. There is a slight wrinkle, however, because to estimate the variance, we use our estimate of the mean. The formula for the estimate of the variance based on a sample of random variables is written as s^2 given by:

$$\mathrm{Var}(X) = \int_{-\infty}^{\infty} (x-\mu)^2 f(x)\, dx \approx \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2 \approx \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2 \equiv s^2. \tag{2.21}$$

The factor $1/(N-1)$ comes from the fact that the we have to use the estimate of the mean, $\bar{x}$, instead of the true mean. This factor is called Bessel's correction and comes from the fact that the quantity $(x_i - \bar{x})$ has N values but only $N-1$ independent values because the sum of $(x_i - \bar{x})$ must equal zero. Nevertheless, if N is large, the correction has a small effect.

The skewness has a similar formula for an estimator; it is a combination of the sample mean, $\bar{x}$, and the sample variance, s^2, along with an additional integral estimate. The skewness estimate for a sample is written as b_1 and given by

$$b_1 = \frac{\frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^3}{(s^2)^{3/2}}. \tag{2.22}$$

The excess kurtosis for a sample is written as g_2

$$g_2 = \frac{\frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^4}{(s^2)^2} - 3. \tag{2.23}$$

There is no simple formula for computing the median of a sample. In principle, one needs to either sort the list of samples and find the middle element, if there are an odd number of samples, or in the case of an even number of elements to take the average of the two elements adjacent to the middle of the list. There are more sophisticated algorithms that can find the smallest $N/2$ items in a list.

2.3 Multivariate Distributions

Consider a vector of random p variables: $\mathbf{X} = (X_1, X_2, \ldots, X_p)^t$. We can discuss properties of this collection of random variables in a similar way to a single random variable. First, we define the *joint cumulative distribution function* (joint CDF) as

$$F(\mathbf{a}) = F(a_1, a_2, \ldots, a_p) = P(X_1 \leq a_1, X_2 \leq a_2, \ldots, X_p \leq a_p). \tag{2.24}$$

This function is the probability that each random variable is smaller than a given number. As before, this definition allows the difference of the joint CDFs to give you the probability that each random variable is within a range:

$$F(\mathbf{b}) - F(\mathbf{a}) = P(a_1 < X_1 \leq b_1, a_2 < X_2 \leq b_2, \ldots, a_p < X_p \leq b_p).$$

As before, the derivative of the joint CDF is the *joint probability density function* (joint PDF):

$$f(\mathbf{x}) = f(x_1, x_2, \ldots, x_p) = \left. \frac{\partial^p F(\mathbf{x})}{\partial x_1 \partial x_2 \ldots \partial x_p} \right|_{\mathbf{x}}. \tag{2.25}$$

The joint CDF is then the integral of the joint PDF in a similar fashion to the single variable:

$$F(\mathbf{x}) = \int_{-\infty}^{x_1} dx_1' \int_{-\infty}^{x_2} dx_2' \ldots \int_{-\infty}^{x_p} dx_p' \, f(\mathbf{x}'). \tag{2.26}$$

Using the joint PDF, we can get the PDF of a single variable. For instance, $f(x_1)$ can be computed by integrating over the other $p - 1$ variables:

$$f(x_1) = \int_{-\infty}^{\infty} dx_2 \ldots \int_{-\infty}^{\infty} dx_p \, f(\mathbf{x}'). \tag{2.27}$$

That is, if we integrate over the second through pth variables, we will have a function of just x_1 that is equal to it PDF. In this case we would call $f(x_1)$ the *marginal* probability density function for random variable X_1. Additionally, we can define a marginal cumulative distribution function for X_1 as

$$F(x_1) = \int_{-\infty}^{x_1} dx_1' \int_{-\infty}^{\infty} dx_2' \ldots \int_{-\infty}^{\infty} dx_p' \, f(\mathbf{x}'). \tag{2.28}$$

Clearly, the marginal PDF and CDF could be defined for any of the p variables in the multivariate distribution.

We can generalize the idea of the marginal PDF into the joint marginal PDF of any subset of the p variables. Say for $l < p$ variables, the joint PDF for these l variables is

$$f(x_1, x_2, \ldots, x_l) = \int_{-\infty}^{\infty} dx_{l+1} \ldots \int_{-\infty}^{\infty} dx_p \, f(\mathbf{x}'). \tag{2.29}$$

These definitions then allow us to define a *conditional probability distribution function* (conditional PDF). The conditional PDF gives the distribution of a collection of random variables provided that another collection of random variables takes particular values. For an example, imagine a collection of two random variables, X and Y. We can define the probability distribution of Y provided $X = x$ as

$$f(y|X = x) = \frac{f(x, y)}{\int_{-\infty}^{\infty} f(x, y)\, dy} = \frac{\text{Prob. density of } x \text{ and } y}{\text{Prob. density of } x \text{ for any } y}. \tag{2.30}$$

Using the definition of Eq. (2.27), we can simplify this, for $f_X(x) \neq 0$.

$$f(y|X = x) = \frac{f(x, y)}{f_X(x)},$$

where we have used the subscript X to indicate that f_X is the PDF of the random variable X. Going back to the more general case, the conditional probability of l random variables given $p - l$ other variables is

$$\begin{aligned} &f(x_1, \ldots, x_l | X_{l+1} = x_{l+1}, \ldots, X_p = x_p) \\ &\quad = \frac{f(\mathbf{x})}{f(x_{l+1}, \ldots, x_p)} \qquad f(x_{l+1}, \ldots, x_p) \neq 0, \end{aligned} \tag{2.31}$$

where

$$f(x_{l+1}, \ldots, x_p) = \int_{-\infty}^{\infty} dx_1 \ldots \int_{-\infty}^{\infty} dx_l \, f(\mathbf{x}).$$

The mean of a collection of random variables is just the vector, $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_p)$, of the expected values for each element in the collection:

$$\mu_i = \int_{-\infty}^{\infty} dx_1 \int_{-\infty}^{\infty} dx_2 \ldots \int_{-\infty}^{\infty} dx_p \, x_i \, f(\mathbf{x}). \tag{2.32}$$

The variance for a collection of random variables is more complicated than that for a single variable because we can look at how the random variables change together. The measure of this is called the covariance and the covariance between X_i and X_j is written as σ_{ij}:

$$\begin{aligned}\sigma_{ij} &= E[(X_i - \mu_i)(X_j - \mu_j)] \\ &= \int_{-\infty}^{\infty} dx_1 \int_{-\infty}^{\infty} dx_2 \ldots \int_{-\infty}^{\infty} dx_p \, (x_i - \mu_i)(x_j - \mu_j) f(\mathbf{x}),\end{aligned} \tag{2.33}$$

note $\sigma_{ij} = \sigma_{ji}$. The covariance of X_i with itself is the variance of X_i:

$$\sigma_{ii} = \sigma_i^2 = \int_{-\infty}^{\infty} dx_1 \int_{-\infty}^{\infty} dx_2 \ldots \int_{-\infty}^{\infty} dx_p \, (x_i - \mu_i)^2 f(\mathbf{x}). \tag{2.34}$$

The covariances form a $p \times p$ symmetric matrix with the diagonal being the variance of each random variable. The covariance matrix is typically denoted by $\Sigma(\mathbf{x})$ so that

$$\Sigma_{ij}(\mathbf{x}) = \sigma_{ij}. \tag{2.35}$$

There is a special case for a collection of random variables where the joint PDF can be factored into the product of individual PDFs as

$$f(\mathbf{x}) = \prod_{i=1}^{p} f(x_i).$$

This type of multivariate distribution is said to be independent: the value of one random variable does not depend on the value another random variable takes. An independent collection of random variables will have a covariance matrix with no nonzero off-diagonal elements. However, the opposite is not true. It is possible to have zero covariance between variables but for them to not be independent. In other words, independence is sufficient for two variables to have a zero covariance between them, but not necessary to have zero covariance.

Example: Multivariate Normal Distribution

The multivariate normal distribution is a higher-dimension version of the normal random variable. In this case a collection of variables is jointly distributed according to a mean value for each, and a covariance matrix for the relation between the variables. The probability density function for a multivariate normal PDF of k variables is given by

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right). \tag{2.36}$$

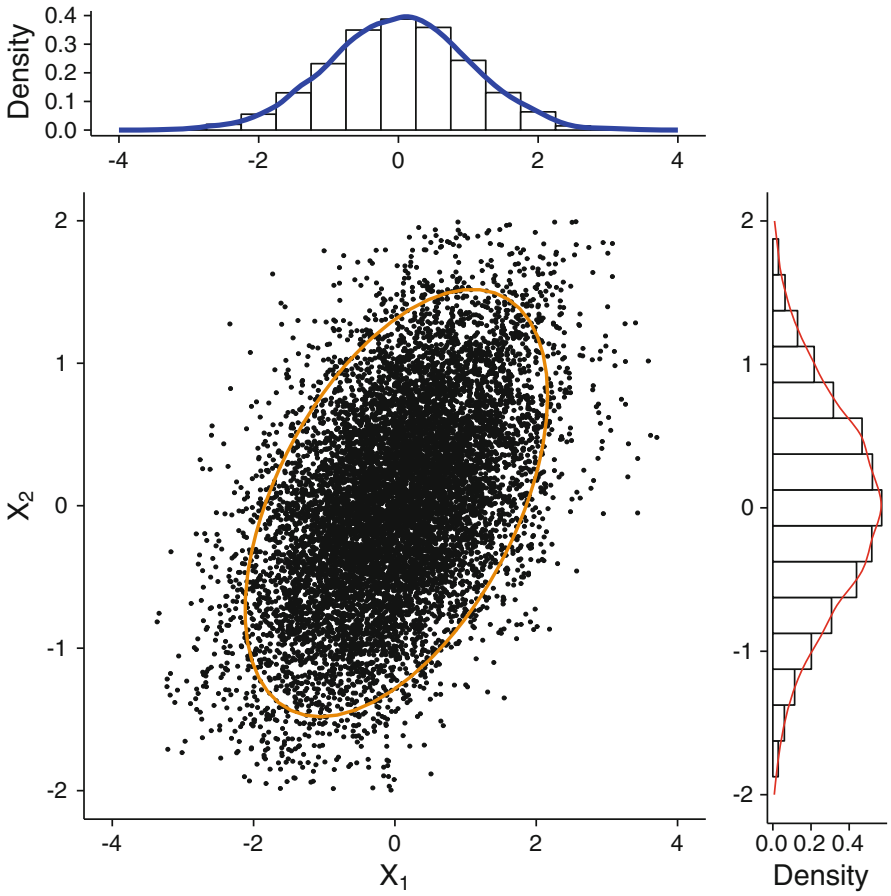

Fig. 2.8 Ten-thousand samples from a 2-D multivariate normal random variable with $\text{Var}(x_1) = 1$, $\text{Var}(x_2) = 0.5$ and covariance $\sigma_{12} = 0.35$. The histograms show the marginal distribution of the samples and the ellipse is the 95% probability interval for the distribution (i.e., the integral of the joint PDF over that ellipse will be 0.95)

Here $\mathbf{x}$ is a k-dimensional vector, $\mathbf{x} = (x_1, x_2, \ldots, x_k)^{\mathrm{T}}$, $\boldsymbol{\mu}$ is a vector of the expected value, or mean of each of the random variables X_i:

$$\boldsymbol{\mu} = (E[X_1], E[X_2], \ldots, E[X_k])^{\mathrm{T}} = (\mu_1, \mu_2, \ldots, \mu_k)^{\mathrm{T}},$$

and the covariance matrix Σ was defined in Eq. (2.35), with the determinant of the matrix written as $|\Sigma|$. The notation for a random variable $\mathbf{X}$ to be a multivariate normal with mean vector, $\boldsymbol{\mu}$, and covariance matrix Σ is $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ (see Fig. 2.8).

2.4 Stochastic Processes

We can generalize the idea of a finite collection of random variables by defining a stochastic process that is a collection of a continuum of random variables. In this sense, it is an infinite-dimensional collection of random variables that has an index analogous to the input to a function. The stochastic processes we will use will be defined over a finite domain. In the case of a stochastic process, the mean and covariances will be functions.

A stochastic process u over the domain $x \in [a, b]$ will be written as $u(x; \xi)$ where ξ is used to denote a particular realization of the stochastic process. Defining CDFs and PDFs for a stochastic process is not as simple as a finite collection of random variables because now the CDF is a function of a function, rather than a function of a vector. As we will see, there are particular stochastic process where we can define these functions.

We can define a mean function $\mu(x)$ which is mean value of the stochastic process as a function of x. Additionally, we can write $k(x_1, x_2)$ as the covariance between the value of the stochastic process at points x_1 and x_2. From the covariance function, we can define a variance as $\sigma^2(x) = k(x, x)$.

Simple Example of a Stochastic Process

Consider the stochastic process

$$u(x; \xi) = \cos(x + A), \qquad x \in [0, 2\pi],$$

where A is a random variable given by $A \sim N(0, 1)$. In this case we can write the mean function as

$$\mu(x) = \int_{-\infty}^{\infty} \frac{\cos(x + a)}{\sqrt{2\pi}} e^{-a^2/2}\, da = \frac{\cos(x)}{\sqrt{e}}.$$

The covariance function in this case will be

$$\begin{aligned} k(x_1, x_2) &= \int_{-\infty}^{\infty} \frac{(\cos(x_1 + a) - \mu(x_1))(\cos(x_2 + a) - \mu(x_2))}{\sqrt{2\pi}} e^{-a^2/2}\, da \\ &= \frac{(e - 1)\,(e \cos(x_1 - x_2) - \cos(x_1 + x_2))}{2e^2}. \end{aligned}$$

From the covariance function, we can get the variance at any point x as

$$\sigma^2(x) = \frac{(e - 1)(e - \cos(2x))}{2e^2}.$$

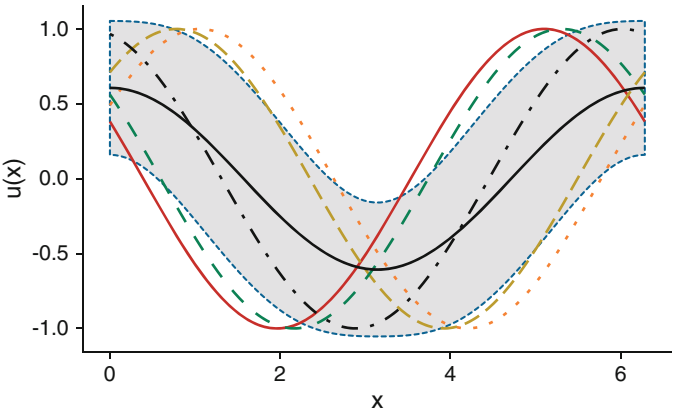

Fig. 2.9 Five realizations of a simple stochastic process $u(x;\xi) = \cos(x+A)$, $x \in [0, 2\pi]$, where A is a random variable given by $A \sim N(0,1)$. The black line is the mean function of the process, $\mu(x)$, and the gray band represents $\mu(x) \pm \sigma(x)$

Five realizations of this process are shown in Fig. 2.9. This is a particularly simple stochastic process because all of the randomness is contained in a single parameter, and this makes the mean and covariance functions computable.

2.4.1 Gaussian Processes

A particular type of stochastic process is the Gaussian process. A Gaussian process is a stochastic process where any finite collection of points $\mathbf{x}$ are described by a multivariate normal distribution with mean $\mu(x)$ and covariance matrix with elements given by

$$\Sigma_{ij} = k(x_i, x_j).$$

The function $k(x_i, x_j)$ is sometimes called a kernel function, and it needs to be defined so that it yields a valid covariance matrix. This means, for example, that the function must be symmetric in its arguments, $k(x_i, x_j) = k(x_j, x_i)$, and $k(x_i, x_j) \geq 0$.

The Gaussian process is useful because, like a multivariate normal, it is completely described by its mean and covariance. That means if we only know the mean and covariance of a stochastic process, we can define a Gaussian process. Also, at any point in the system, we know the variance in the stochastic process because $\sigma^2(x) = k(x, x)$.

Example Gaussian processes are shown in Figs. 2.10, 2.11, and 2.12. In each of these, we change the covariance and mean functions to modify the behavior of the process. Notice that the behavior over space appears more random (i.e., less spatially

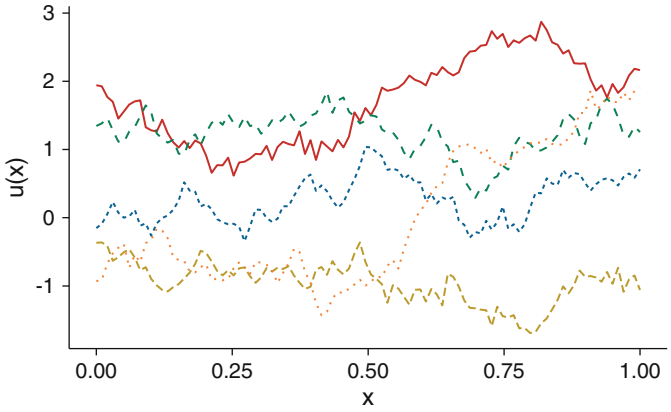

Fig. 2.10 Five realizations of a Gaussian process defined on $x \in [0, 1]$ with $\mu(x) = 0$ and $k(x_1, x_2) = \exp(-|x_1 - x_2|)$

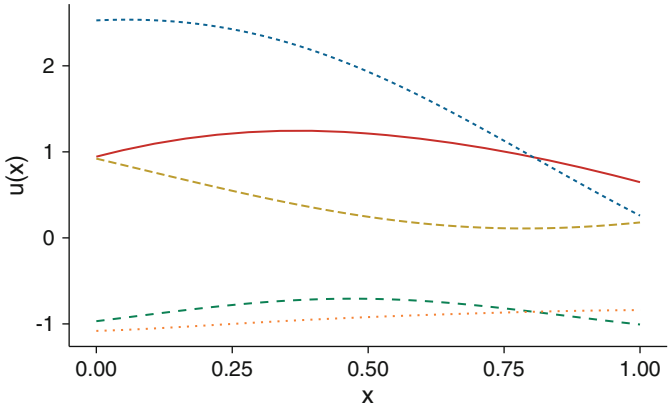

Fig. 2.11 Five realizations of a Gaussian process defined on $x \in [0, 1]$ with $\mu(x) = 0$ and $k(x_1, x_2) = \exp(-(x_1 - x_2)^2)$

correlated) when the covariance function is a simple exponential compared with a squared exponential covariance function.

2.5 Sampling a Random Variable

In general it is easy to get a random variable that is uniformly distributed between 0 and 1. In fact, almost all programming languages have a function for generating such random numbers. We would like the ability to generate a random sample of any type of random variable. This is can be done if the CDF of the distribution is known by inverting the CDF of the random variable. As mentioned above, the CDF is a

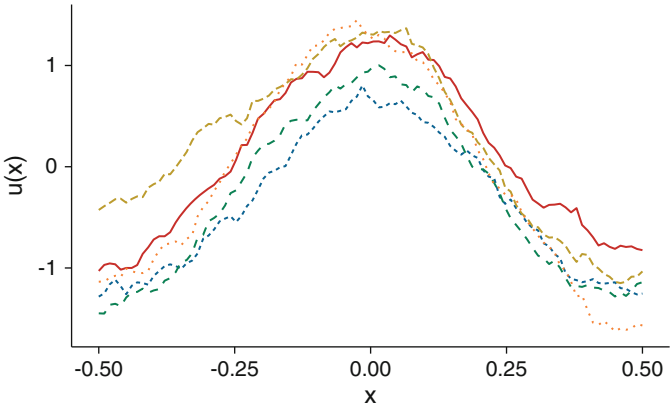

Fig. 2.12 Five realizations of a Gaussian process defined on $x \in [-0.5, 0.5]$ with $\mu(x) = \cos(2\pi x)$ and $k(x_1, x_2) = 0.1 \exp(-|x_1 - x_2|)$

function that has a range from [0, 1] and is a monotonic, non-decreasing function. Therefore, the CDF is invertible. With this result we can take a uniformly distributed random variable between 0 and 1 and invert the CDF to get a sample of the random variable associated with that CDF. That is,

$$x = F^{-1}(\xi), \qquad \xi \sim \mathscr{U}(0, 1), \tag{2.37}$$

will give a sample x that is distributed according to the CDF $F(x)$. Note that if the CDF has jumps, then the inverse CDF is defined so that it gives the smallest value x such that $F(x) = \xi$.

An illustration of this procedure is shown in Fig. 2.13 for a standard normal random variable. Here we show samples of a uniformly distributed variable between 0 and 1 and the corresponding samples from the distribution after inverting the CDF. Notice where the CDF is changing more rapidly, there is a higher density of samples.

Example: Sampling from an Exponential PDF

An exponential random variable has PDF

$$f(x) = \lambda e^{-\lambda x}, \quad x \geq 0,$$

where $\lambda > 0$ is a parameter. The CDF of an exponential random variable is

$$F(x) = \int_0^x \lambda e^{-\lambda x'} dx' = 1 - e^{-\lambda x}.$$

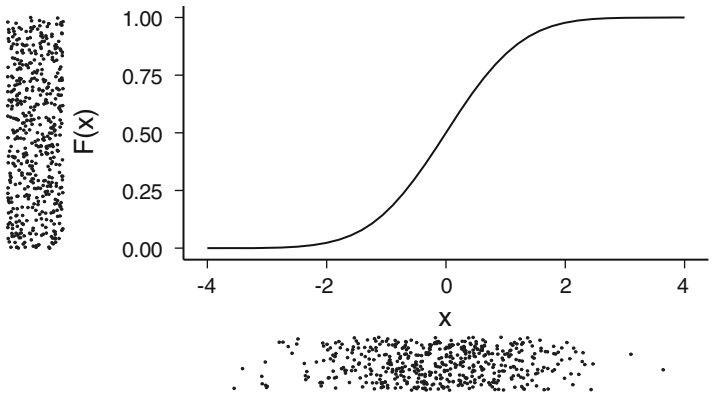

Fig. 2.13 In this figure we show a set of points on the y-axis that are randomly chosen between 0 and 1. Then on the x axis, we show the corresponding sample points from inverting the CD (in this case the standard normal CDF). Notice that the uniform samples in y are nonuniformly clustered around 0, as we would expect for samples from a standard normal random variable

To sample an exponential random variable, choose $\xi \sim \mathscr{U}(0, 1)$ randomly and set

$$F(x) = 1 - e^{-\lambda x} = \xi \quad \Rightarrow \quad 1 - \xi = e^{-\lambda x}.$$

Therefore,

$$x = \frac{-\log(1-\xi)}{\lambda},$$

and x will be distributed according to $f(x)$.

Example: Normal Random Variable

In this example we will explain a clever way of inverting the CDF for a standard normal random variable. A sample from a standard normal random variable can be transformed to a general normal random variable through the relation:

$$x = \mu + z\sigma, \qquad Z \sim \mathscr{N}(0, 1).$$

Consider a normal random variable with mean 0 and standard deviation 1. The associated PDF will be

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}.$$

The Box-Muller transform gives a way to get two samples at a time. Consider the product of two PDFs:

$$f(x)\,dx f(y)\,dy = \frac{e^{\frac{-(x^2+y^2)}{2}}}{2\pi}\,dx\,dy.$$

If we change coordinates into polar coordinates so that

$$dx\,dy = r\,dr\,d\theta,$$

for $r = \sqrt{x^2 + y^2}$, and $\theta = \tan^{-1}(y/x)$, we can write

$$f(x)f(y)\,dy\,dx = e^{\frac{-r^2}{2}}\,r\,dr\frac{d\theta}{2\pi} \quad r \in [0, \infty),\ \theta \in [0, 2\pi].$$

We can separate this expression into two functions,

$$g(r) = e^{\frac{-r^2}{2}}r,$$

and

$$h(\theta) = \frac{1}{2\pi}.$$

These functions are both properly normalized PDFs:

$$\int_0^\infty g(r)\,dr = \int_0^{2\pi} d\theta\,h(\theta) = 1.$$

One can easily sample a θ:

$$\theta = 2\pi\xi_1, \qquad \xi_1 \in [0, 1].$$

To sample an r from $g(r)$, we can use the result from the previous example if we define $u = r^2$ and $du = 2rdr$ to get

$$r = \sqrt{-2\log(1 - \xi_2)}, \qquad \xi_2 \in [0, 1].$$

As a result, drawing two random numbers, ξ_1 and ξ_2, gives two samples from the Gaussian:

$$x = r\cos\theta, \qquad y = r\sin\theta.$$

This compares with the brute force approach of inverting the CDF for a normal random variable, with inverting two simple CDFs. The trade-off is that one needs to generate two samples at a time.

2.5.1 Sampling a Multivariate Normal

Consider a collection of p random variables that are multivariate normal, $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$. To sample from this distribution, we will first take the Cholesky decomposition of the covariance matrix:

$$\Sigma = \mathbf{L}\mathbf{L}^{\mathrm{T}},$$

where $\mathbf{L}$ is a lower triangular matrix. The Cholesky decomposition exists for any symmetric matrix of real values that is positive definite. The covariance matrix satisfies these properties. The Cholesky decomposition requires $O(p^3)$ floating point operations to compute and is therefore expensive when p is large.

With the Cholesky decomposition, we then generate p independent samples from a standard normal random variable:

$$\mathbf{z} = (z_1, \ldots, z_p)^{\mathrm{T}}, \qquad Z_i \sim \mathcal{N}(0, 1).$$

To get a sample from $\mathbf{X}$, we then compute

$$\mathbf{x} = \boldsymbol{\mu} + \mathbf{L}\mathbf{Z}.$$

To demonstrate how this procedure works, we look at the covariance matrix of the vector $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. In terms of the expected value,

$$\Sigma(\mathbf{Z}) = E[\mathbf{Z}\mathbf{Z}^{\mathrm{T}}] = \mathbf{I}.$$

Now consider a vector $\mathbf{X} = \mathbf{L}\mathbf{Z}$. The covariance matrix for this collection of random variables is

$$E[\mathbf{X}\mathbf{X}^{\mathrm{T}}] = E[\mathbf{L}\mathbf{Z}(\mathbf{L}\mathbf{Z})^{\mathrm{T}}] = E[\mathbf{L}\mathbf{Z}\mathbf{Z}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}]$$

From this we can move the $\mathbf{L}$'s from outside the expectation operator to get

$$E[\mathbf{X}\mathbf{X}^{\mathrm{T}}] = \mathbf{L}E[\mathbf{Z}\mathbf{Z}^{\mathrm{T}}]\mathbf{L}^{\mathrm{T}} = \mathbf{L}\mathbf{L}^{\mathrm{T}} = \Sigma(\mathbf{X}).$$

To shift this result to a variable with a nonzero mean, we add in the desired mean $\boldsymbol{\mu}$.

2.5.2 Sampling a Gaussian Processes

Previously, we discussed a Gaussian stochastic process, which is a stochastic process where any finite number of points are jointly Gaussian with a known mean function, $\mu(x)$, and covariance function $k(x_1, x_2)$. To generate realizations of a Gaussian process, we need to specify, I, the number of points in space we want to evaluate the process at and the value of x at those points: $x_i, i = 1, \ldots, I$.

We then sample from a multivariate normal with mean vector given by

$$\boldsymbol{\mu} = (\mu(x_1), \ldots, \mu(x_I))^{\mathrm{T}},$$

and a covariance matrix given by

$$\Sigma_{ij} = k(x_i, x_j), \qquad i, j = 1, \ldots, I.$$

The vectors that we sample can be interpreted as the Gaussian process evaluated at each point:

$$U(x_1), \ldots, U(x_I) \sim N(\boldsymbol{\mu}, \Sigma).$$

This method was used to produce the realizations of Gaussian processes in Figs. 2.10, 2.11, and 2.12.

Producing realizations of Gaussian processes at a large number of points can be costly because we need to compute the Cholesky factorization of the covariance matrix. This can limit the number of points at which one generates realizations of Gaussian processes.

2.6 Rejection Sampling

In some cases it is difficult to create the CDF from the PDF or the CDF may not be known in closed form or may not be invertible except by expensive numerical solution. In this case, it can be easier to use rejection sampling. To illustrate how this works, we will take the PDF for a random variable X, where the random variable takes values only inside a given range $[a, b]$. We then draw a rectangle around the function. The base of the rectangle extends from a to b and the height of the rectangle is the maximum value of the PDF, called h here. An example of this is shown in Fig. 2.14. Then we pick points at random in the box, i.e., $X \sim \mathscr{U}(a, b)$, and $Y \sim \mathscr{U}(0, h)$. If the point is below the PDF, i.e., $y \leq f(x)$, then we accept it, and if not is rejected. The accepted values of X are our samples from the random variable. Figure 2.15 shows how rejection sampling proceeds as more points are tried.

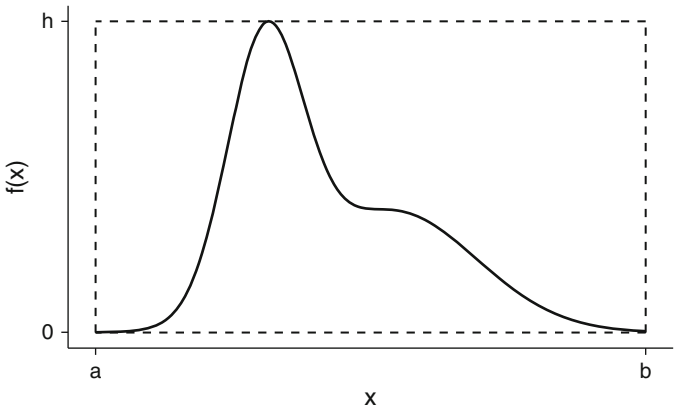

Fig. 2.14 Illustration of drawing a box around the PDF for a random variable for the purpose of rejection sampling

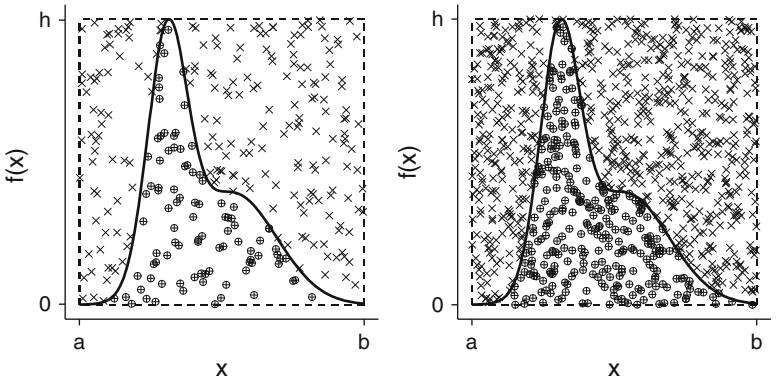

Fig. 2.15 Rejection sampling at two different number of attempted samples, 300 (left) and 1000 (right). The points with an "times" symbol were rejected and the "circled plus" points were accepted

The rejection rate is an important measure of the effectiveness of the rejection sampling procedure. If the function is highly peaked, and goes to zero slowly, many of the sampled points can be rejected. When the rejection rate is high, it can make generating samples difficult, especially if evaluating the PDF is expensive.

One can see this in a highly peaked distribution. In such a case, it can be more efficient to draw a different shape around the function than a rectangle. In Fig. 2.16, this is demonstrated using a triangle that circumscribes the PDF. The rejection rate for this function would be much higher if we used a rectangle to bound the PDF.

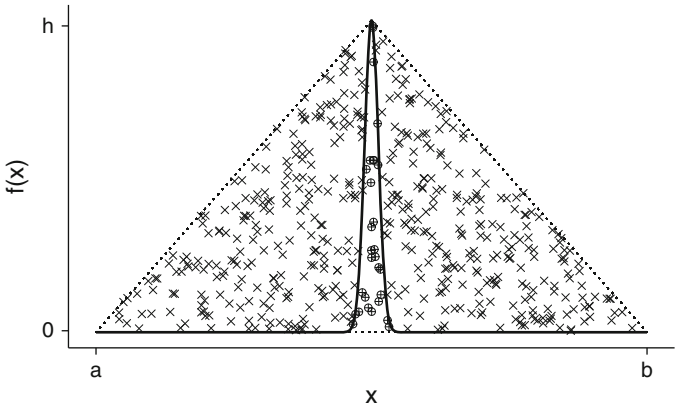

Fig. 2.16 Rejection sampling using a triangle to bound the PDF

2.7 Bayesian Statistics

Previously, we defined the conditional probability $f(x|Y = y)$ as the probability density that $X = x$ conditional on $Y = y$. We will often drop the "$Y =$" part from the expression. Note that we could define parameters in a distribution as random variables. For example, the mean and variance of normal random variable could be random variables. In this sense we could write the conditional probability of X given $\mu = 0$ and $\sigma^2 = 1$ as

$$f(x|\mu = 0, \sigma^2 = 1) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$

Additionally, in Eq. (2.31) we wrote that the conditional probability was the joint probability density function divided by the marginal probability density function, viz.,

$$\begin{aligned} &f(x_1, \ldots, x_l | X_{l+1} = x_{l+1}, \ldots, X_p = x_p) \\ &= \frac{f(\mathbf{x})}{f(x_{l+1}, \ldots, x_p)} \qquad f(x_{l+1}, \ldots, x_p) \neq 0. \end{aligned}$$

Therefore, for random variables X and Y, the conditional probability of X given Y can be written as

$$f(x|y) f_Y(y) = f(x, y). \tag{2.38}$$

Additionally, the conditional probability of Y given X can be written as

$$f(y|x) f_X(x) = f(x, y). \tag{2.39}$$

Equating these two expressions and rearranging, we can write Bayes' law (or Bayes' theorem or Bayes' rule)

$$f(x|y) = \frac{f(y|x)f_X(x)}{f_Y(y)}. \tag{2.40}$$

A more common way to write Bayes' rule is to use the relation

$$f_Y(y) = \int_{-\infty}^{\infty} f(x, y)\,dx = \int_{-\infty}^{\infty} f(y|x)f_X(x)\,dx.$$

Then Bayes' law is

$$f(x|y) = \frac{f(y|x)f_X(x)}{\int_{-\infty}^{\infty} f(y|x)f_X(x)\,dx}. \tag{2.41}$$

Oftentimes, we write Bayes' law using special notation that indicates the interpretation of its implications. We define $\pi(x)$ as the *prior* probability density function for X, and $\pi(x|y)$ as the *posterior* conditional probability density function for X given $Y = y$, and $f(y|x)$ as the conditional likelihood, or just likelihood, of y given $X = x$. Using this notation we write

$$\pi(x|y) = \frac{f(y|x)\pi(x)}{\int_{-\infty}^{\infty} f(y|x)\pi(x)\,dx}. \tag{2.42}$$

The interpretation of Bayes' law is that we have a prior density function for x that we update given the observation that $Y = y$ to get $\pi(x|y)$.

Example: False Positives

Assume a drug test is 99% accurate in the sense that the test will produce 99% true positive and 99% true negative results. Say 0.5% of the population use the drug. An individual tests positive. What is the probability they are a user?

$$\begin{aligned} P(\text{user}|+\ \text{test}) &= \frac{P(+|\text{user})P(\text{user})}{P(+|\text{user})P(\text{user}) + P(+|\text{non-user})P(\text{non-user})} \\ &= \frac{0.99 \cdot 0.005}{0.99 \cdot 0.005 + 0.01 \cdot 0.995} = 0.332, \end{aligned}$$

or 33.2%.

Example: Fairness of a Coin

Say we want to know the fairness of a coin (i.e., is the probability of heads $\frac{1}{2}$?). If I flip the coin 10 times and get 3 heads, what is my estimate of the probability of getting heads on any toss? Using Bayes' rule we write the probability of heads as p and write

$$f(p|y) = \frac{f(y|p)\pi(p)}{\int_{-\infty}^{\infty} dp f(y|p)\pi(p)}.$$

In this equation

- $f(y|p)$ = probability density of getting y given a value of p,
- $\pi(p)$ = prior distribution on p (what I believe given no data), and
- $f(p|y)$ = posterior distribution for p given data y.

For the coin example, we claim to have no idea if the coin is fair, i.e., p could be anywhere between 0 and 1 with equal likelihood. We express this as

$$\pi(p) = \begin{cases} 1 & p \in [0, 1] \\ 0 & \text{otherwise} \end{cases}.$$

The probability of getting 3 heads in 10 tosses or trials is a binomial random variable where each flip has probability p of getting heads (see Appendix A), with probability mass function:

$$f(3|p) = \binom{10}{3} p^3(1-p)^7 = 120p^3(1-p)^7.$$

The denominator for Bayes' theorem is

$$\int_0^1 120p^3(1-p)^7\, dp = \frac{1}{11}.$$

Putting this all together, we get the posterior

$$f(p|3) = 1320p^3(1-p)^7.$$

In Fig. 2.17, we show the results of this trial. We see in the posterior the maximum is at $p = 0.3$, but it does not rule out the coin being fair. The posterior does rule out, however, $p = 0$ or $p = 1$, because those are not possible given the observation of only 3 heads.

A useful feature of Bayes' theorem is that we can update the posterior if new data comes along in the same way as before. That is, we use the current posterior as

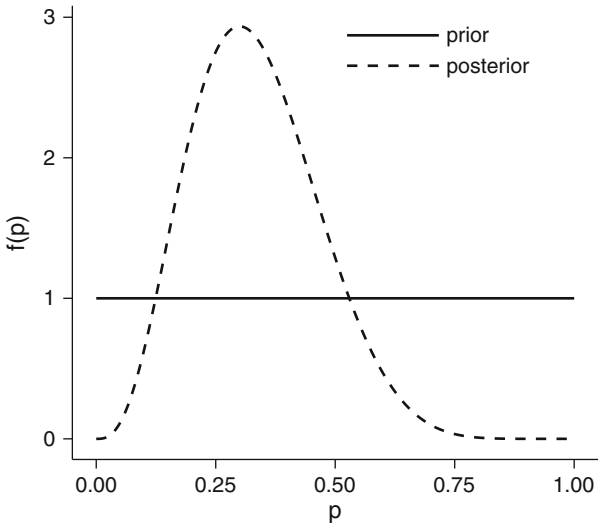

Fig. 2.17 Posterior and prior distributions of the probability of getting 3 heads in 10 tosses for a coin of unknown fairness

the prior in another calculation. If we make 990 more flips of the coin and get 430 heads, this makes the likelihood in the numerator

$$f(430|p) = \binom{990}{430} p^{430}(1-p)^{560} = 5.127419 \times 10^{292} p^{430}(1-p)^{560},$$

and the denominator is

$$\int_0^1 1320 \binom{990}{430} p^{433}(1-p)^{567}\, dp = \frac{2016464117980615134777}{9987612500849703903228 50}.$$

Then, using $\pi(p) = f(p|30)$, Bayes' theorem gives

$$\begin{aligned} f(p|460) &= \frac{1}{0.0020190} \binom{990}{430} p^{430}(1-p)^{560}(1320p^3(1-p)^7) \\ &= \frac{1320}{0.0020190} \binom{990}{430} p^{433}(1-p)^{567}. \end{aligned}$$

The new posterior distribution is highly peaked around $p = 0.433$, as seen in Fig. 2.18, indicating that it is likely that this coin is not quite fair. The maximum of the posterior moved considerably from the result based on ten trials. In other words, Bayes' theorem does what we would want: a large number of trials, in this case 990, have a larger impact on the posterior than a smaller number.

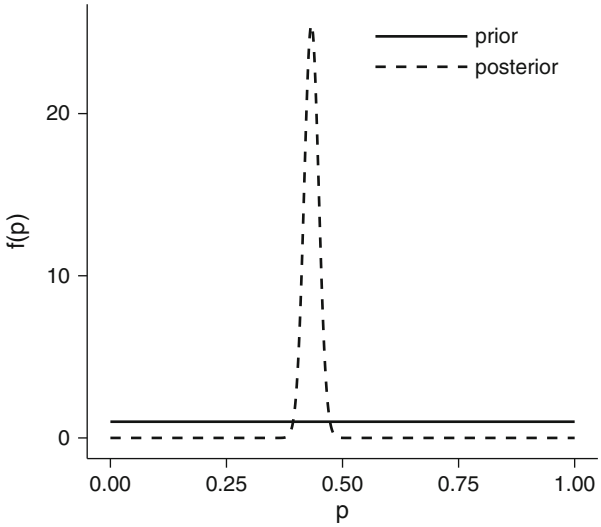

Fig. 2.18 Posterior and prior distributions of the probability of getting heads for the coin tossing example

In the example above, we initially used what is known as an uninformed prior. That is, we put no information into the prior on p, other than saying p could be anywhere in [0, 1]. An uninformed prior is a conservative choice when we have no other information. Had we chosen a different prior, say something peaked around $p = 0.5$, we may have gotten a different result after 10 trials.

A criticism of Bayesian calculations is that the choice of prior can matter and affect the results. Consider the hypothetical example of a shipping method of radioactive waste. If the expected value for the chance of a catastrophic accident is 10^{-3} per year of shipping, and shipping has been going on for 25 years without an accident, does that mean we can adjust the probability of a catastrophic accident? How we answer this question would depend on our choice of prior. If the prior was a delta function centered at 10^{-3}, then the distribution and the expected failure rate would not change. However, if the distribution on the failure rate had a large variance, then the operation history would affect the posterior distribution of the failure rate.

A problem with Bayes' theorem is that it can be hard to estimate the integral in the denominator, except for some particular cases called conjugate priors. If the likelihood and the prior are chosen correctly, then the integral can be done analytically. For example, if the likelihood and the prior are both normal, then the posterior will also be normal.

Without conjugate priors, it may be difficult to estimate the integral in the denominator in Bayes' theorem. We could use quadrature approximations if we can easily evaluate the likelihood and prior. This type of approximation can be too expensive if the integral is over a high-dimensional space (i.e., the x in Bayes'

theorem is a vector of many variables). Later, we will discuss an approach, called Markov chain Monte Carlo, to generate samples from a posterior distribution without needing to compute the denominator or needing a closed form for the numerator.

2.8 Exercises

1. Show that the transformation in Eq. (2.6) results in a standard normal random variable by computing the mean and variance of Z.
2. Consider the random variables $X \sim U(-1, 1)$ and $Y \sim X^2$. Are these independent random variables? What is their covariance?
3. Show that a general covariance matrix must be positive definite, i.e. $\mathbf{x}^{\mathrm{T}} \boldsymbol{\Sigma} \mathbf{x} > 0$ for any vector $\mathbf{x}$ that is not all zeros.
4. Use rejection sampling to sample from a Gamma random variable $X \sim \mathcal{G}(\alpha, \beta)$ where

$$f(x) = \frac{x^{\alpha} e^{-\beta x}}{\Gamma(\alpha + 1)\beta^{-\alpha - 1}}, \qquad \alpha > -1, \beta > 0.$$

   Let $\alpha = 0$ and $\beta = 0.5$. From rejection sampling with a $N = 10^4$, compute a rejection rate for the sampling procedure. Now draw a triangle around the function and do rejection sampling. Compare the rejection rate from the triangle versus the rectangle. You may consider that the PDF is zero if $f(x) < 10^{-6}$.
5. Consider a random variable, $X > 0$, that has its logarithm distributed by a normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$. Such a distribution is called a log-normal distribution. Compute this distribution's (a) mean, (b) variance, (c) median, (d) mode, (e) skew, and (f) kurtosis.
6. (Monty Hall Problem) You are on a game show and are presented with three doors from which to choose. One of the doors contains a prize and the other two have nothing. You pick a door (say door 1), and then the host opens another door (say door 3), and asks if you want to switch to door number 2. What should you do?

   (a) Using Bayes' theorem give the probability of winning if you switch.
   (b) Write a simulation code to show this by randomly assigning a prize to a door, then opening either door 2 or 3 depending on which has the prize, and then either switching or not. Compute the likelihood of winning if you switch.

7. Consider a variable Y distributed by a normal distribution with mean given by θ:

$$f(y|\theta) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y-\theta)^2}{2\sigma^2}\right).$$

Now consider θ to be a random variable as well, and σ to be a known constant. Then say θ is normally distributed, with mean μ and variance τ^2 to give

$$\pi(\theta) = \frac{1}{\tau\sqrt{2\pi}} \exp\left(-\frac{(\theta-\mu)^2}{2\tau^2}\right).$$

The parameters μ and τ are called hyperparameters. Using Bayes' theorem find $p(\theta|y)$, and show that it is a normal distribution.

8. Suppose that X is the number of people arriving at a particular tavern during a given hour. This type of arrival process is naturally described by a Poisson process:

$$f(x|\theta) = \frac{e^{-\theta}\theta^x}{x!}, \qquad x \in \{0, 1, 2, \dots\}, \quad \theta > 0.$$

We then say that our prior distribution of θ is a Gamma distribution

$$\pi(\theta) = \frac{\theta^{\alpha-1}e^{-\beta\theta}}{\Gamma(\alpha)\beta^{-\alpha}}, \qquad \alpha, \beta > 0.$$

Therefore, we say that $\theta \sim G(\alpha, \beta)$.

- Show using Bayes' theorem that the posterior distribution for θ given x is proportional to a Gamma distribution.
- Suppose you observe 42 people arriving in 1 h, and the prior distribution has $\alpha = 5$, and $\beta = 6$. Generate samples from the posterior distribution and show graphically how the prior as changed given the observation.

9. Generate N samples from a standard normal random variable and estimate the mean, variance, skewness, and kurtosis from the samples. Use $N = 10, 10^2, \dots, 10^4$, and discuss how the errors in the approximations behave as a function of N.
10. Consider the joint PDF

$$f(x, y) = e^{-x/y}, \qquad x \in [0, \infty) \quad y \in (0, \sqrt{2}].$$

Compute and plot the marginal PDFs for X and Y. Additionally, compute the conditional probability distributions, and make plots of $f(y|X = \mu_x)$ and $f(x|Y = \mu_y)$.
11. Consider a covariance function between points in 2-D space:

$$k(x_1, y_1, x_2, y_2) = \exp\left[-|x_1 - x_2| - |y_1 - y_2|\right].$$

Generate four realizations of a Gaussian stochastic process with zero mean, $\mu(x, y) = 0$, and this covariance function defined on the unit square, $x, y \in [0, 1]$. For the realizations, evaluate the process at 50 points in each direction. Plot the realizations.

Chapter 3
Input Parameter Distributions

Yes, I'm paranoid—but am I paranoid enough?

—David Foster Wallace, *Infinite Jest*

In this chapter we will explore how we can use the principles of statistics and probability to model input parameters to simulation models. This discussion will require that we understand how random variables depend on each other, how we can model this dependence when we have limited information, and how we can approximate a collection of random variables, or even a stochastic process, based on some underlying structure.

In a computer simulation, there will be typically several random variables as inputs. For a collection of random variables, it is common to not have an expression for the joint distribution functions (CDF or PDF) for the collection. Rather, the best one can do is hope to have some measure of the dependence between the pairs of variables. As we will see, the dependence measures we use are not enough to uniquely determine the relationship between random variables.

Additionally, later when we try to model the distribution of output quantities of interest based on input uncertainties, we will see that the number of random variables we have as input determine the accuracy we can achieve with our uncertainty quantification given a fixed computational budget. Therefore, we would like to determine if we can eliminate input random variables if there is an underlying correlation or approximation. Methods for this type of reduction will be discussed in this chapter as well.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-319-99525-0_3) contains supplementary material, which is available to authorized users.

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0_3

3.1 Dependence Between Variables

So far we have discussed probability distributions and multivariate distributions in some detail. For collections of random variables, we are often interested in how they vary together. We already have a measure for this: the covariance. One issue with the covariance between two random variables, X and Y,

$$\Sigma(X, Y) = E[XY] - E[X]E[Y], \tag{3.1}$$

is that it has units that are the product of the units of X and Y. This can make it difficult to compare covariances. For instance, $\Sigma(X, Y) > \Sigma(X, Z)$ does not imply that there is a stronger relationship between X and Z than X and Y because of the units.

3.1.1 Pearson Correlation

A normalized measure of the relation between two random variables is the Pearson correlation coefficient, ρ. Oftentimes, this is simply called the correlation coefficient or correlation. Considering two random variables, X, and Y, the correlation coefficient is

$$\rho(X, Y) = \frac{E[XY] - E[X]E[Y]}{\sigma_X \sigma_Y}. \tag{3.2}$$

That is, the Pearson correlation is the covariance normalized by the standard deviation of each variable. On this normalized scale, we can say things about how two variables change together. If the variables are independent, then $\rho(X, Y) = 0$. As with covariance, a correlation of zero between variables does not imply that the variables are independent.

One property of the correlation coefficient is that if X and Y are linearly related, i.e., there exist an a and b such that $Y = aX + b$, then $\rho(X, Y) = \text{sign}(a)$. As a corollary, if we define a new random variable $X' = aX + b$, we have the relation

$$\rho(X', Y) = \text{sign}(a)\rho(aX + b, Y),$$

which can be shown found using the properties of the expected value.

When we have a collection of random variables, $\mathbf{X} = (X_1, X_2, \ldots, X_p)^{\mathrm{T}}$, we can define a correlation matrix $\mathbf{R}$ in terms of the covariance matrix as

$$R_{ij} = \frac{\Sigma_{ij}}{\sigma_{X_i} \sigma_{X_j}}, \tag{3.3}$$

where $\sigma_{X_i}^2 = \Sigma(X_i, X_i)$ is the variance in X_i.

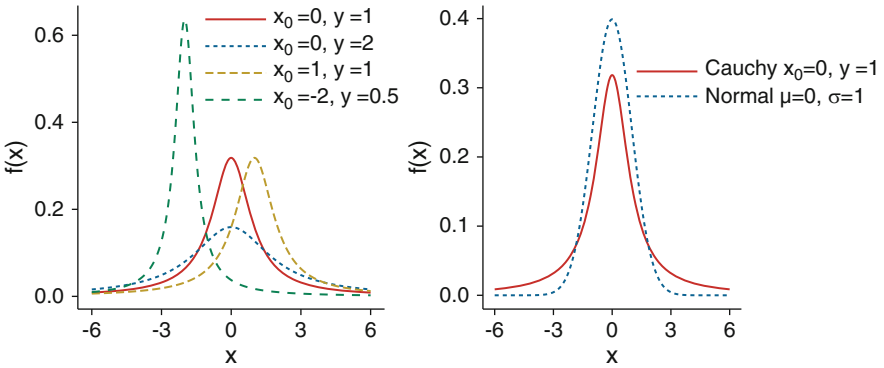

Fig. 3.1 The Cauchy distribution with various parameters and compared with the standard normal

The benefit of the Pearson correlation coefficient is that it is easy to calculate, as simple as the covariance matrix. However, there are some downsides. One is that it is not defined if the expected value of XY is not defined (just as the covariance is not defined in this case). The Pearson correlation is not defined for a Cauchy random variables . This type of random variable is given by a PDF with parameters x_0, γ:

$$f(x) = \frac{1}{\pi\gamma}\left[1 + \left(\frac{x - x_0}{\gamma}\right)^2\right]^{-1}. \tag{3.4}$$

The mean and variance of the distribution are undefined because the distribution goes to zero too slowly, but the median and mode are x_0. The PDF for a Cauchy distribution and it's comparison to the standard normal are given in Fig. 3.1.

Another, potentially more important, downside of the Pearson correlation coefficient is that if X is transformed by a nonlinear, strictly increasing function, $g(X)$, the correlation $\rho(X, Y)$ will be different than $\rho(g(X), Y)$. This means that if there is a nonlinear relation between X and Y, the Pearson correlation coefficient may under- or overestimate the relation between the two variables.

3.1.2 Spearman Rank Correlation

An alternative to the Pearson correlation is the Spearman rank correlation, or Spearman correlation. In this measure we look for general, monotonic relationships between two variables. This is defined by looking at the correlation between the marginal CDF of each variable:

$$\rho_S(X, Y) = \rho(F_X(x), F_Y(y)). \tag{3.5}$$

If we do not know the marginal CDF, but we have samples of the random variables, we can still estimate the Spearman correlation. Given N samples of X and Y, we create a function that takes sample x_i or y_i and gives the rank of that sample among the N samples:

$$\text{rank}(x_i) = \text{rank of } x_i \text{ in sample population.}$$

Using this function we then define the Spearman correlation coefficient for the samples:

$$\rho_S(X, Y) = \frac{\sum_{i=1}^{N}(\text{rank}(x_i) - \bar{r}_X)(\text{rank}(y_i) - \bar{r}_Y)}{\sqrt{\sum_{i=1}^{N}(\text{rank}(x_i) - \bar{r}_X)^2}\sqrt{\sum_{i=1}^{N}(\text{rank}(y_i) - \bar{r}_Y)^2}}, \tag{3.6}$$

where

$$\bar{r}_X = \frac{1}{N}\sum_{i=1}^{N}\text{rank}(x_i).$$

When computing ρ_S any ties in the data are assigned the average rank of the tied scores.

One of the important properties of the Spearman correlation is that if there exists a strictly increasing function $g(X)$ that relates X to Y as $Y = g(X)$, then $\rho_S(X, Y) = 1$. Furthermore, a strictly monotonic transformation of X or Y will not affect the Spearman correlation

As with the Pearson correlation, we can compute a Spearman correlation matrix for a collection of random variables $\mathbf{X} = (X_1, \ldots, X_p)^T$. We will call this matrix $\mathbf{R}_S$, and it is given by

$$R_{S,ij} = \rho_S(X_i, X_j).$$

3.1.3 Kendall's Tau

The final measure of correlation that we will use is Kendall's tau or the Kendall rank correlation coefficient. Similar to the Spearman correlation, it tries to measure the relation between two variables in terms of the ranks. It is best for looking at a sample population of random variables because it requires looking at pairs of samples of random variables. To define Kendall's tau, consider N samples of random variables x and y. We examine all the pairs of samples (x_i, y_i) for $i \neq j$. There are $\frac{1}{2}N(N-1)$ such pairs. We look at each pair and say that a pair ij is concordant if $x_i > x_j$ and $y_i > y_j$ or if $x_i < x_j$ and $y_i < y_j$. A pair is discordant if $x_i > x_j$ and $y_i < y_j$ or if $x_i < x_j$ and $y_i > y_j$. If either $x_i = x_j$ or $y_i = y_j$, then the pair is a tie.

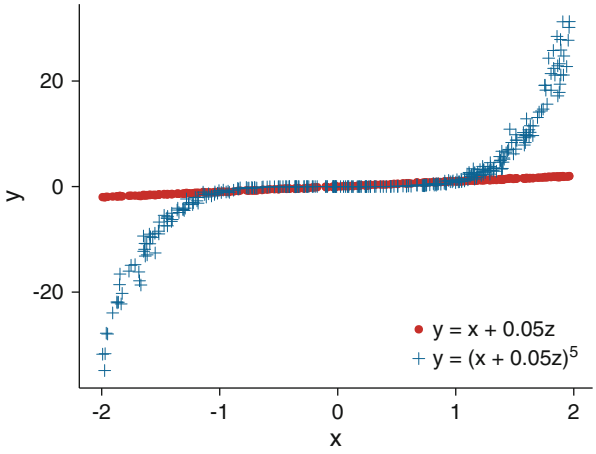

Fig. 3.2 The comparison of Pearson, Spearman, and Kendall's tau correlation measures on 300 samples of two pairs of random variables, $(x, x+0.05z)$ and $(x, (x+0.05z)^5)$, where z is a standard normal random variable. The three measures give a correlation of $\rho = 0.999$, $\rho_S = 0.999$, and $\tau = 0.973$ for the correlation of $(x, x + 0.05z)$. For the correlation of $(x, (x + 0.05z)^5)$, the Spearman correlation and Kendall's tau values do not change, but $\rho = 0.843$ for this data

Using this comparison of pairs, we define Kendall's tau as

$$\tau = \frac{(\#\text{ of concordant pairs}) - (\#\text{ of discordant pairs})}{\frac{1}{2}N(N-1)}. \tag{3.7}$$

The range of τ is $[-1, 1]$. Kendall's tau has the property that it is not affected by performing a nonlinear, increasing transformation on either random variable: this is the same property Spearman correlation has. We can relate τ to the Pearson correlation coefficient if the variables X and Y are jointly normally distributed through the equation

$$\tau(X, Y) = \frac{2}{\pi} \arcsin \rho(X, Y).$$

We will use Kendall's tau when we want to relate two random variables through copulas.

As comparison of the correlation measures, Fig. 3.2 shows how a strictly increasing transformation of a variable changes the Pearson correlation, but not the Spearman correlation or Kendall's tau. In the figure the correlation between random variables $(x, x + 0.05z)$ and the correlation between $(x, (x + 0.05z)^5)$, where z is a standard normal random variable, are computed. The Spearman and Kendall measures do not change, whereas the Pearson correlation drops by 15%.

3.1.4 Tail Dependence

Another important characterization of how two variables vary together is tail dependence. This is a measure of the correlation between variables as their lower and upper bounds are approached. The lower tail dependence, λ_l is

$$\lambda_l(X, Y) = \lim_{q \to 0} \mathrm{P}(Y \leq F_Y^{-1}(q) \mid X \leq F_X^{-1}(q)). \tag{3.8}$$

This is the probability that Y goes to its lower bound as X goes to its lower bound. The upper tail dependence is

$$\lambda_u(X, Y) = \lim_{q \to 1} \mathrm{P}(Y > F_Y^{-1}(q) \mid X > F_X^{-1}(q)), \tag{3.9}$$

and measures the probability that X and Y go to their upper bound together.

Tail dependence is different than typical correlation measures in that it is only interested in extreme values. For example, two variables could have a Pearson correlation of 0.5, but a tail dependence is much larger, say 0.9. This has been observed, for example, in the returns of stocks. Many stocks that had low correlation in typical times had very high lower tail dependence during the financial crises (they all went down a lot).

The lower tail dependence can be written in terms of the joint CDF for two variables. Using the definition of the CDF and law of total probability, we get that

$$\mathrm{P}(Y \leq F_Y^{-1}(q) \mid X \leq F_X^{-1}(q)) = \frac{F_{XY}(F_X^{-1}(q), F_Y^{-1}(q))}{F_X(F_X^{-1}(q)))} = \frac{F_{XY}(F_X^{-1}(q), F_Y^{-1}(q))}{q}, \tag{3.10}$$

where F_{XY} is the joint CDF for X and Y. Similarly, the upper tail dependence can be written in terms of the joint CDF as

$$\begin{aligned} \mathrm{P}(Y > F_Y^{-1}(q) \mid X > F_X^{-1}(q)) &= \frac{\mathrm{P}(Y > F_Y^{-1}(q), X > F_X^{-1}(q))}{P(Y > F_Y^{-1}(q))} \\ &= \frac{1 - P(X \leq F_X^{-1}(q)) - P(Y \leq F_Y^{-1}(q)) + F_{XY}(F_X^{-1}(q), F_Y^{-1}(q))}{1 - F_X(F_X^{-1}(q))} \\ &= \frac{1 - 2q + F_{XY}(F_X^{-1}(q), F_Y^{-1}(q))}{1 - q}. \end{aligned} \tag{3.11}$$

These equations give us formulas for the tail dependences in terms of the joint and marginal CDFs for each of these variables.

3.2 Copulas

A common occurrence when evaluating collections of random variables is that it is often much easier to determine the marginal CDF or PDF of each variable rather than determine the joint distribution functions. Moreover, given a sample of data, it is possible to estimate correlations between the random variables (in either of the three flavors we mentioned in the previous section). The question is, given the scenario where one has

- An estimate of the marginal CDF of each random variable,
- An estimate of the correlation between the random variables,

can one generate a joint distribution between the variables and generate samples from the joint distribution? Clearly, there is not a unique way of creating this joint distribution because many functions could replicate the marginal distributions and have a defined correlation.

To answer this question, we turn to copulas (or copulæ if one is a fan of Latinisms). We will begin with discussing bivariate copulas before generalizing the idea to general collections of random variables. The word copula comes from a Latin for linking together; in our context it will link marginal distributions to a joint distribution.

A copula, $C(u, v)$, joins random variables X and Y if the joint CDF can be written as

$$F_{XY}(x, y) = C(F_X(x), F_Y(y)). \tag{3.12}$$

This definition takes the marginal CDF for each variable and creates a joint CDF. A result known as Sklar's theorem tells us that such a copula will exist for any joint CDF, and it is unique if the marginal CDFs are continuous. A copula has the domain $u, v \in [0, 1]$ and a range of $[0, 1]$. For a given copula, we can define the joint PDF as

$$f(x, y) = c(F_X(x), F_Y(y)) f_X(x) f_Y(y), \tag{3.13}$$

where the copula density, $c(u, v)$, is given by

$$c(u, v) = \frac{\partial^2}{\partial u \partial v} C(u, v). \tag{3.14}$$

This definition is a special case of Eq. (2.25). Additionally, the conditional CDF $C(v|u)$ is

$$C(v|u) = \frac{\partial}{\partial u} C(u, v). \tag{3.15}$$

The tail dependence for a copula can be obtained by plugging Eq. (3.12) into the definitions for tail dependence, Eqs. (3.8) and (3.9), to get

$$\lambda_l = \lim_{q \to 0} \frac{C(q, q)}{q}, \tag{3.16}$$

and

$$\lambda_u = \lim_{q \to 1} \frac{1 - 2q + C(q, q)}{1 - q}. \tag{3.17}$$

The simplest copula is the independent copula:

$$C_{\mathrm{I}}(u, v) = uv.$$

Copulas are widely used in the finance and insurance industries to model the joint distributions of risks. Because the fact that mapping marginal distributions to joint distributions is not unique, the way we use copulas requires choices by the user. The considerations of ease of use, matching observed correlation, and tail dependence have to be weighed when choosing a copula.

3.2.1 Normal Copula

A simple but useful copula is the normal (or Gaussian) copula:

$$C_N(u, v) = \Phi_{\mathbf{R}}(\Phi^{-1}(u), \Phi^{-1}(v)), \tag{3.18}$$

where $\mathbf{R}$ is a correlation matrix for the intended joint distribution. The normal copula is simple to sample. Given two random variables X and Y with marginal CDFs $F_X(x)$ and $F_Y(y)$, we can generate a sample from $C_N(F_X(x), F_Y(y))$ using the following procedure:

1. Sample from the collection of two random variables $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ using the Cholesky factorization approach in the previous chapter.
2. Compute $u = \Phi(z_1)$ and $v = \Phi(z_2)$.
3. The samples are $x = F_X^{-1}(u)$ and $y = F_Y^{-1}(v)$.

Therefore, via the normal copula, we can create a joint distribution that has a prescribed Pearson correlation where the underlying marginal distributions do not have to be normal. This is different than saying that the two variables are a multivariate normal with a known correlation. Note that the matrix $\mathbf{R}$ has only 1

degree of freedom because the diagonal is 1 and it is symmetric; we can call this degree of freedom ρ. It can be shown that for a normal copula, the value of Kendall's tau is

$$\tau(X, Y) = \frac{2}{\pi} \arcsin \rho, \tag{3.19}$$

Therefore, given a desired value of Kendall's tau for the joint distribution, one can produce it using the normal copula.

The normal copula has zero tail dependence: as one variable approaches $\pm\infty$, the probability that the other variable does the same goes to zero. Therefore, if we are modeling a system where tail dependence could matter greatly, e.g., analyzing how the system behaves under input variables near their extremes, the normal copula may not be appropriate.

The normal has been blamed for the financial crisis of 2008 (Jones 2009) because it does not account for the fact that mortgage defaults, while not being correlated under normal circumstances, have strong lower tail dependence because if everyone in a neighborhood is foreclosed, then housing prices fall, and more mortgages then default: a fact that risk assessors never understood, or to be more charitable they did not account for it. The lack of tail dependence needs to be carefully analyzed when quantifying uncertainty in a physical system. In many cases tail dependence could be present, and we need to understand how this may affect our predictions.

In Fig. 3.3, two uniform distributions joined by a normal copula with $\rho = 0.8$ are shown. Notice how there is a clear correlation between the two random variables and, as a result, a clustering in the corners of the distributions. An important property of these samples is that they are not normal; we have just used a normal copula to join them.

3.2.2 *t*-Copula

A distribution similar to the normal is the t-distribution: it is unimodal but has more kurtosis than a normal random variable. This distribution can be used to define a t-copula with a scale parameter $\nu > 0$ and a positive definite, symmetric scale matrix $\mathbf{S}$ with a diagonal of ones as

$$C_t(u, v) = F_{\mathbf{t}}(F_t^{-1}(u), F_t^{-1}(v)), \tag{3.20}$$

where $F_{\mathbf{t}}$ is the joint CDF for a t-distribution with parameters $\boldsymbol{\mu} = \mathbf{0}$, $\mathbf{S}$, and ν. The CDF $F_t(x)$ is the CDF of the t-distribution with parameter ν. The number of degrees of freedom in the $\mathbf{S}$ matrix will be written as r.

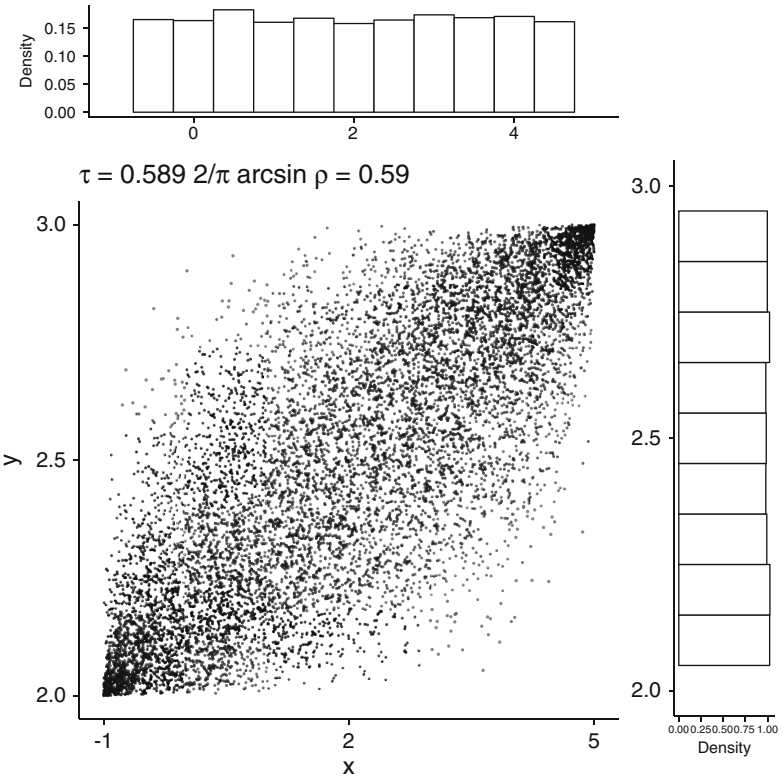

Fig. 3.3 Samples from uniform random variables $X \sim \mathscr{U}(-1, 5)$ and $Y \sim \mathscr{U}(2, 3)$ joined by a normal copula with $\rho = 0.8$. From these 10^4 samples, the empirical value of τ and the predicted value from Eq. (3.19) are shown also

To sample from random variables joined by the t-copula, we use a similar procedure to that for the normal copula:

1. Sample from the collection of two random variables $\mathbf{Z} \sim \mathscr{N}(\mathbf{0}, \mathbf{S})$ using the Cholesky factorization approach in the previous chapter.
2. Compute $\hat{Z} = \sqrt{w}Z$, where w is a sample from the inverse gamma distribution, $W \sim \text{IG}(\nu/2, \nu/2)$.
3. Compute $u = F_t(z_1)$ and $v = F_t(\hat{z}_2)$.
4. The samples are $x = F_X^{-1}(u)$ and $y = F_Y^{-1}(v)$.

The t-copula has the same form for Kendall's tau as the normal copula. In particular if we replace $r \to \rho$ in Eq. (3.19), we can relate Kendall's tau to the matrix $\mathbf{S}$.

In Fig. 3.4, two uniform distributions joined by a t-copula with $r = 0.8$ are shown. Notice how there is a clear correlation between the two random variables and, as a result, a clustering in the corners of the distributions. Also, there are more samples farther off the diagonal than in the normal case. This is due to the

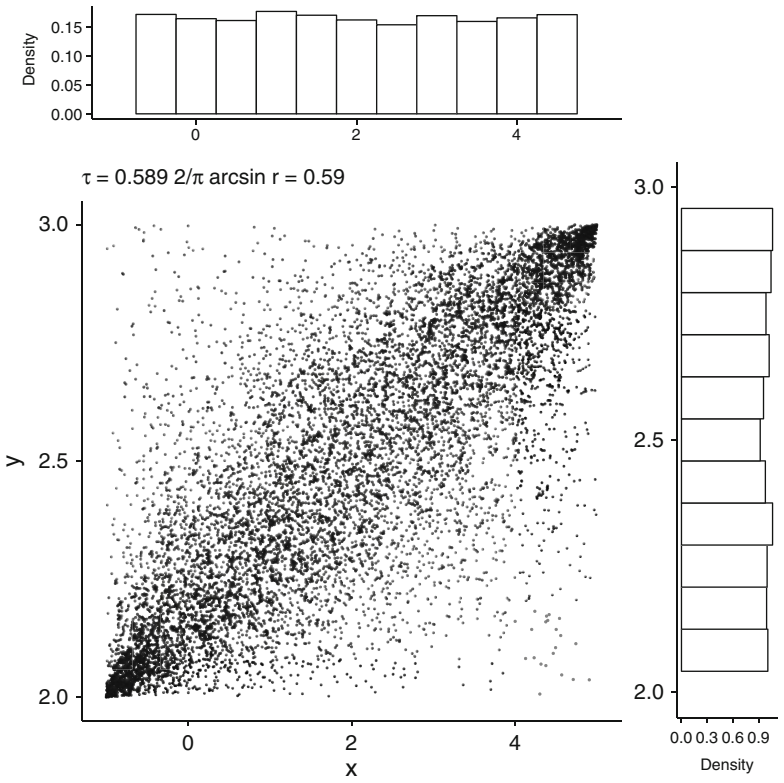

Fig. 3.4 Samples from uniform random variables $X \sim \mathscr{U}(-1, 5)$ and $Y \sim \mathscr{U}(2, 3)$ joined by a t-copula with $r = 0.8$ and $\nu = 4$. From these 10^4 samples, the empirical value of τ and the predicted value from Eq. (3.19) are shown also

fact that the t-distribution with a small value of ν has more kurtosis than a normal distribution. Therefore, it is more likely to get anticorrelated values as samples. The fact that the t-copula has tail dependence can also be observed in this figure in the concentration of points near the lower left and upper right corners.

The tail dependence can be seen even more clearly if we use a t-copula to couple two normal random variables. In Fig. 3.5 the t-copula and normal copulas are compared. Here, we see that the tail dependence appears as the area that the samples occupy narrowing as the upper right and lower left corners are approached in the t-copula, but this not present in the normal copula. This discrepancy in the tails exists even though both distributions have the same value for τ and the same marginal distributions for X and Y. The change in the underlying distribution as a function of r and ν is shown in Fig. 3.6. In this figure two standard normals are joined by a t-copula. As τ increases the tail dependence between the distributions increases.

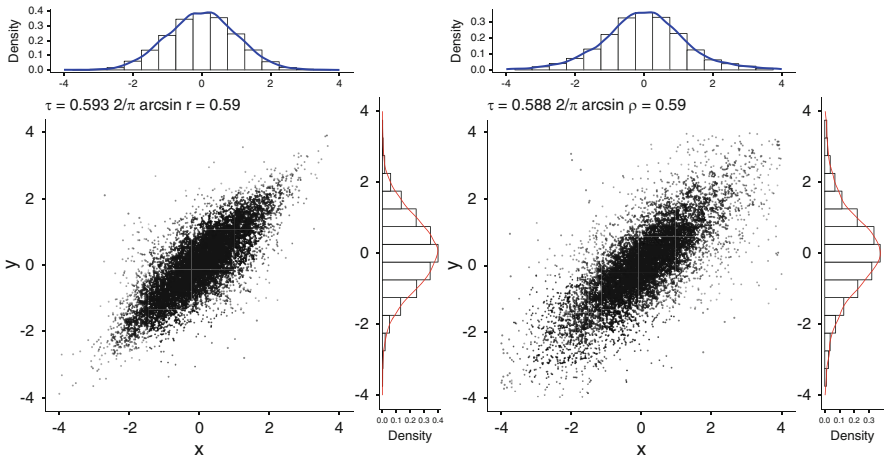

Fig. 3.5 Samples from standard normal random variables $X \sim \mathcal{N}(0, 1)$ and $Y \sim \mathcal{N}(0, 1)$ joined by a t-copula with $r = 0.8$ and $\nu = 4$ (left) and the normal copula with $\rho = 0.8$ (right). From these 10^4 samples, the empirical value of τ and the predicted value from Eq. (3.19) are shown also. Note the tail dependence in the t-copula that is lacking in the normal copula: when one variable is close to ± 4, the other variable is also likely to be close to ± 4

3.2.3 *Fréchet Copulas*

The Fréchet copulas C_L and C_U are simple copulas that join random variables with Spearman correlation ± 1. Furthermore, any other copula is bounded by the relation $C_L \leq C \leq C_U$. The Fréchet copulas are

$$C_L(u, v) = \max(u + v - 1, 0), \qquad C_U(u, v) = \min(u, v). \tag{3.21}$$

C_L will give perfect negative dependence between variables and C_U will give perfect positive correlation between variables. We can then combine Fréchet copulas to describe something with a Spearman correlation between $[-1, 1]$:

$$C_A(u, v) = (1 - A)C_L(u, v) + AC_U(u, v), \qquad A \in [0, 1]. \tag{3.22}$$

These are a simple combination and can give a Spearman correlation given by $2A - 1$.

3.2.4 *Archimedean Copulas*

There is another class of copulas that easily generalize to an arbitrary number of dimensions and have an explicit formula. These copulas, called Archimedean

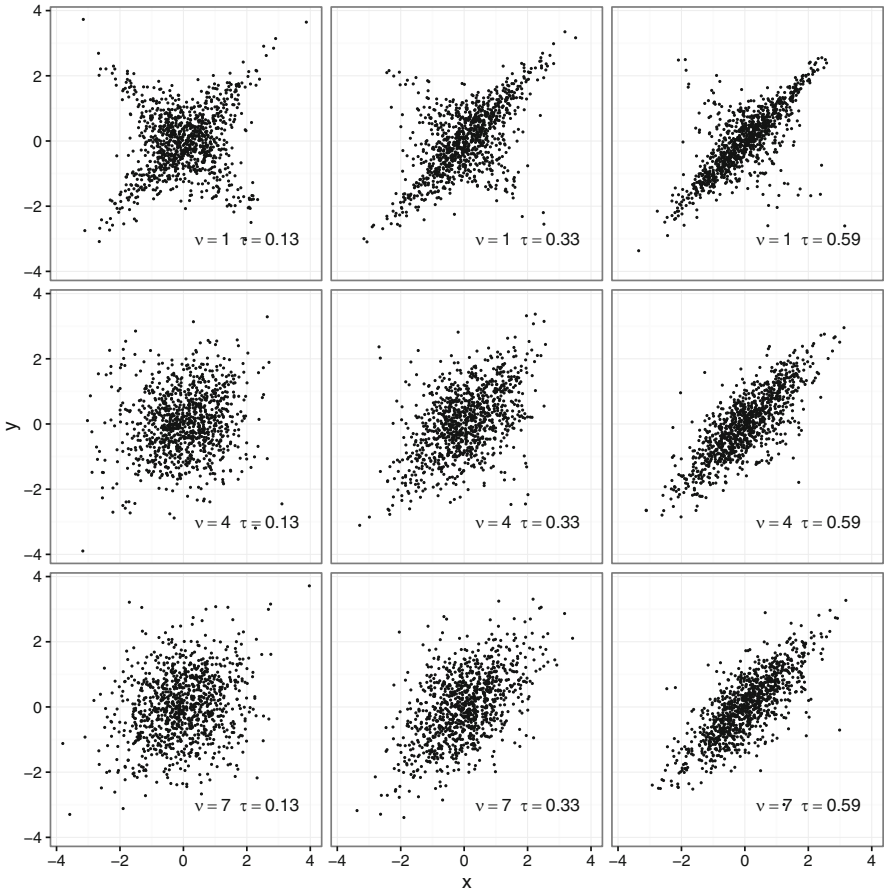

Fig. 3.6 Samples from standard normal random variables $X \sim \mathcal{N}(0, 1)$ and $Y \sim \mathcal{N}(0, 1)$ joined by a t-copula with several values of r and ν. The value of ν is constant in a row, and the value of r (and the corresponding τ) is constant in each column

copulas, are defined by a generator function, $\varphi(t)$ for $t \in [0, \infty)$. Given a generator, we define the quasi-inverse

$$\hat{\varphi}^{-1}(t) \equiv \begin{cases} \varphi^{-1}(t) & 0 \leq t \leq \varphi(0) \\ 0 & \varphi(0) < t < \infty \end{cases} . \tag{3.23}$$

With the generator and quasi-inverse, the Archimedean copula for $\varphi(t)$ is

$$C_{\varphi}(u, v) = \hat{\varphi}^{-1}\left(\varphi(u) + \varphi(v)\right) . \tag{3.24}$$

The term Archimedean arises from the development of the triangle inequality for probability spaces; in that context Archimedes of Syracuse's name is attached a particular norm that has the form of Eq. (3.24).

Archimedean copulas are commutative

$$C_\varphi(u, v) = C_\varphi(v, u),$$

associative

$$C_\varphi(C_\varphi(u, v), w) = C_\varphi(u, C_\varphi(v, w)),$$

and are order preserving

$$C(u_1, v_1) > C(u_2, v_2), \qquad u_1 > u_2, \quad v_1 > v_2.$$

The associative property will be used later to easily create Archimedean copulas for arbitrary numbers of variables.

Furthermore, an Archimedean copula can be related to Kendall's tau via the formula

$$\tau(U, V) = 1 + 4 \int_0^1 \frac{\varphi(t)}{\varphi'(t)}\, dt. \tag{3.25}$$

There are many Archimedean copulas one could define; we will discuss two below that are commonly used.

3.2.4.1 The Frank Copula

One common Archimedean copula is the Frank copula. This copula has a single parameter, $\theta \neq 0$, and a generator function given by

$$\varphi_{\mathrm{F}}(t) = -\log\left(\frac{e^{-\theta t} - 1}{e^{-\theta} - 1}\right). \tag{3.26}$$

The inverse is

$$\hat{\varphi}^{-1}(t) = -\frac{1}{\theta}\log\left(1 + e^{-t}(e^{-\theta} - 1)\right). \tag{3.27}$$

This makes the copula

$$C_{\mathrm{F}}(u, v) = -\frac{1}{\theta}\log\left(1 + \frac{(e^{-\theta u} - 1)(e^{-\theta v} - 1)}{e^{-\theta} - 1}\right). \tag{3.28}$$

Table 3.1 The corresponding value of θ for different values of Kendall's tau using the Frank copula

τ_F	θ
0.1	0.907368
0.2	1.860880
0.3	2.917430
0.4	4.161060
0.5	5.736280
0.6	7.929640
0.7	11.411500
0.8	18.191500
0.9	26.508600

Note that negative values of τ_F will have a corresponding negative value of θ

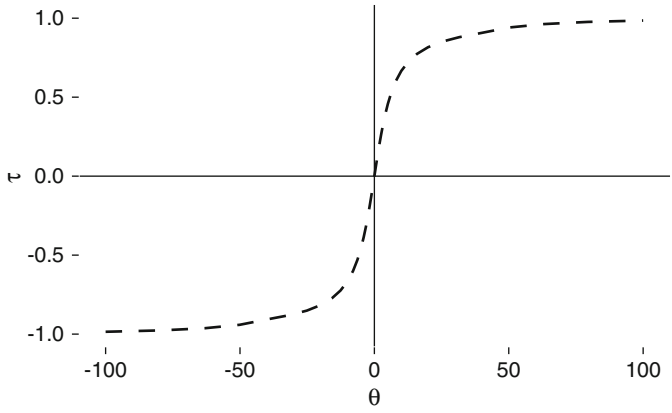

Fig. 3.7 Kendall's tau as a function of θ for the Frank copula

One property of the Frank copula is that as $\theta \to \infty$, the copula becomes the upper Fréchet copula: $C_F \to C_U$. As $\theta \to -\infty$, then the Frank copula approaches the lower Fréchet copula: $C_F \to C_L$.

The value of Kendall's tau for a Frank copula can be calculated from Eq. (3.25) as

$$\tau_F(U, V) = 1 - \frac{2\left(3\theta^2 - 6i\pi\theta + 6\theta - 6\theta \log\left(e^\theta - 1\right) - 6\mathrm{Li}_2\left(e^\theta\right) + \pi^2\right)}{3\theta^2}, \tag{3.29}$$

where $\mathrm{Li}_s(z)$ is the polylogarithm function. A table for matching a desired value of τ_F to θ is given in Table 3.1. Additionally, the value of τ_F as a function of θ is shown in Fig. 3.7. The Frank copula has a tail dependence of zero. Samples from a standard normals joined by a Frank copula are shown in Fig. 3.8, where we observe the lack of tail dependence.

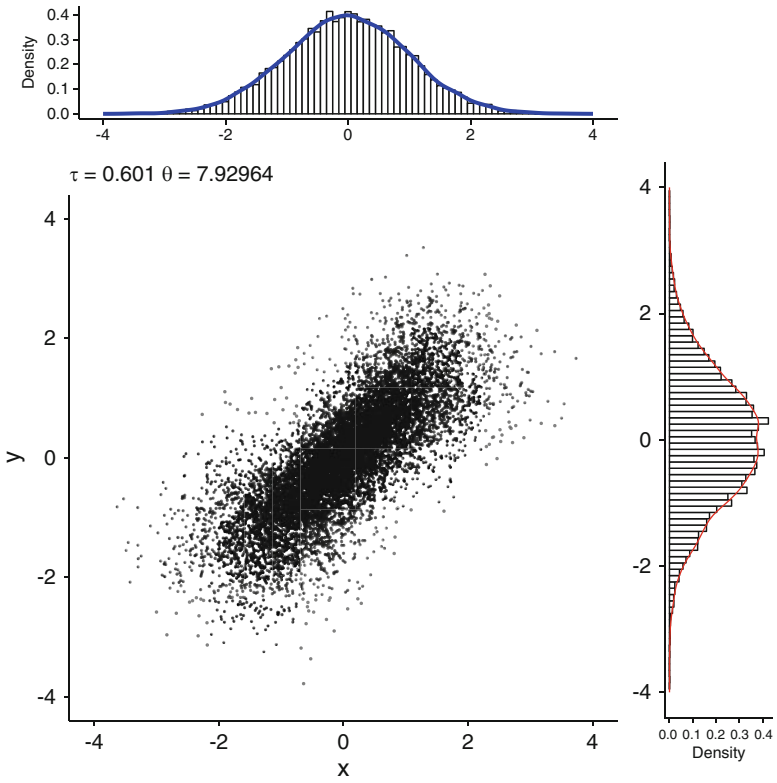

Fig. 3.8 Samples from standard normal random variables $X \sim \mathcal{N}(0, 1)$ and $Y \sim \mathcal{N}(0, 1)$ joined by a Frank copula with θ chosen to get $\tau = 0.6$. Note the lack of tail dependence in the lack of concentration near the upper right and lower left corners. Note that relative to the normal copula and the t-copula, these points form a rectangular-shaped band. The lack of tail dependence is also apparent in the lack of points along the diagonal

In Fig. 3.9 samples from Frank copula are shown with the values of θ given in Table 3.1. In this figure we can see that as θ gets larger, the distribution is pinched in the middle, but the tails of the distribution remain spread out.

3.2.4.2 The Clayton Copula

The Clayton copula generator function has a single parameter, $\theta > 0$, with generator function

$$\varphi_C(t) = t^{-\theta} - 1 \tag{3.30}$$

and inverse

$$\hat{\varphi}_C^{-1}(t) = (1 + t)^{-1/\theta}\,. \tag{3.31}$$

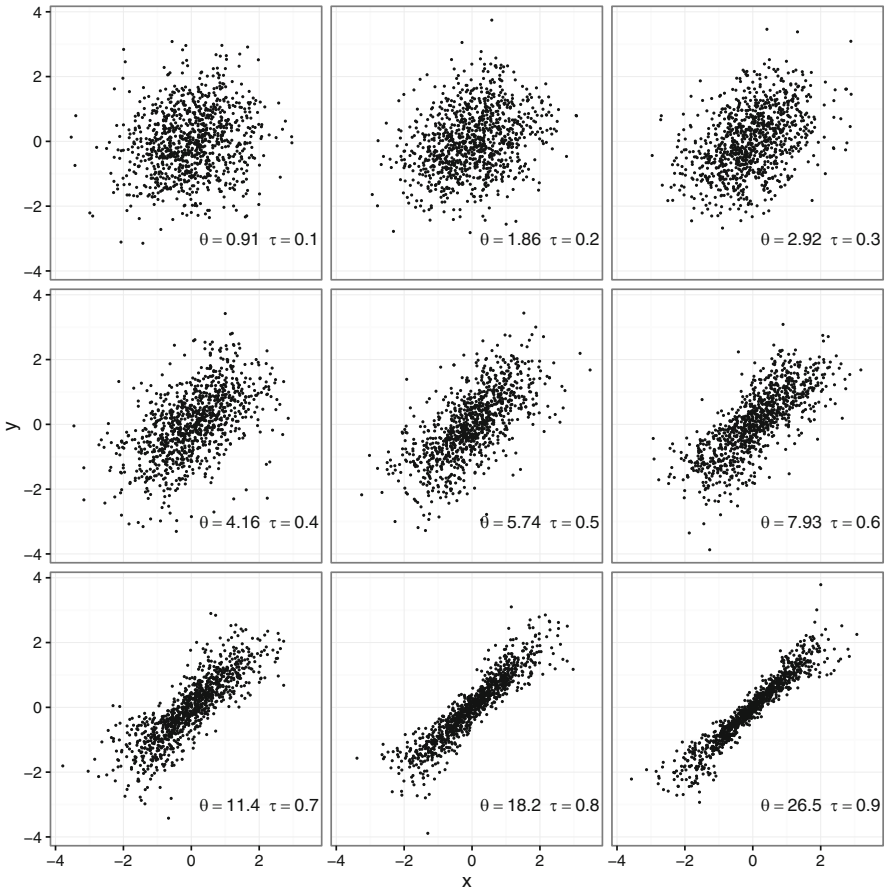

Fig. 3.9 Samples from standard normal random variables $X \sim \mathcal{N}(0, 1)$ and $Y \sim \mathcal{N}(0, 1)$ joined by a Frank copula with several values of θ taken from Table 3.1

The resulting copula is

$$C_{\mathrm{C}}(u, v) = \max\left(0, u^{-\theta} + v^{-\theta} - 1\right)^{-1/\theta}. \tag{3.32}$$

The Clayton copula has Kendall's tau for the resulting joint distribution given by

$$\tau_{\mathrm{C}}(U, V) = \frac{\theta}{\theta + 2}. \tag{3.33}$$

Additionally, the Clayton copula has zero upper tail dependence and nonzero lower tail dependence:

$$\lambda_l = 2^{-1/\theta}. \tag{3.34}$$

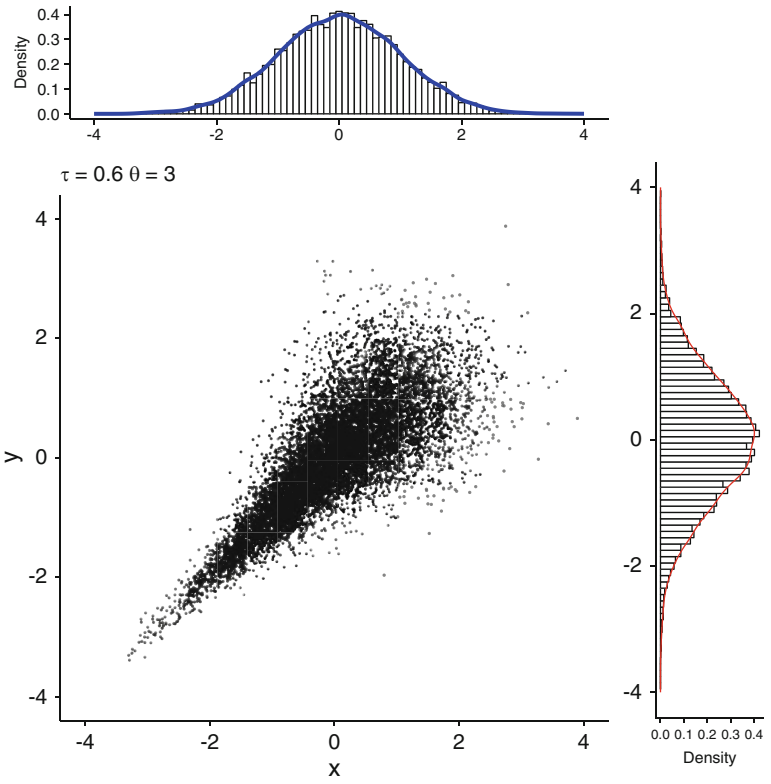

Fig. 3.10 Samples from standard normal random variables $X \sim \mathcal{N}(0, 1)$ and $Y \sim \mathcal{N}(0, 1)$ joined by a Clayton copula with θ chosen to get $\tau = 0.6$. There is strong lower tail dependence in the samples and the zero upper tail dependence

We can use the Clayton copula to produce joint distributions with upper tail dependence and no lower tail dependence by using the copula $C_C(1 - u, 1 - v)$. In Fig. 3.10 two standard normals are joined by a Clayton copula and the strong lower tail dependence can be seen.

The Clayton copula with different values of θ that correspond with values of Kendall's tau from 0.1 to 0.9 is shown in Fig. 3.11. As θ increases, the shape of the distribution becomes more tapered in the middle and makes the lower tail dependence more prominent, as predicted, making the samples form something akin to the celebrate emoji 🎉.

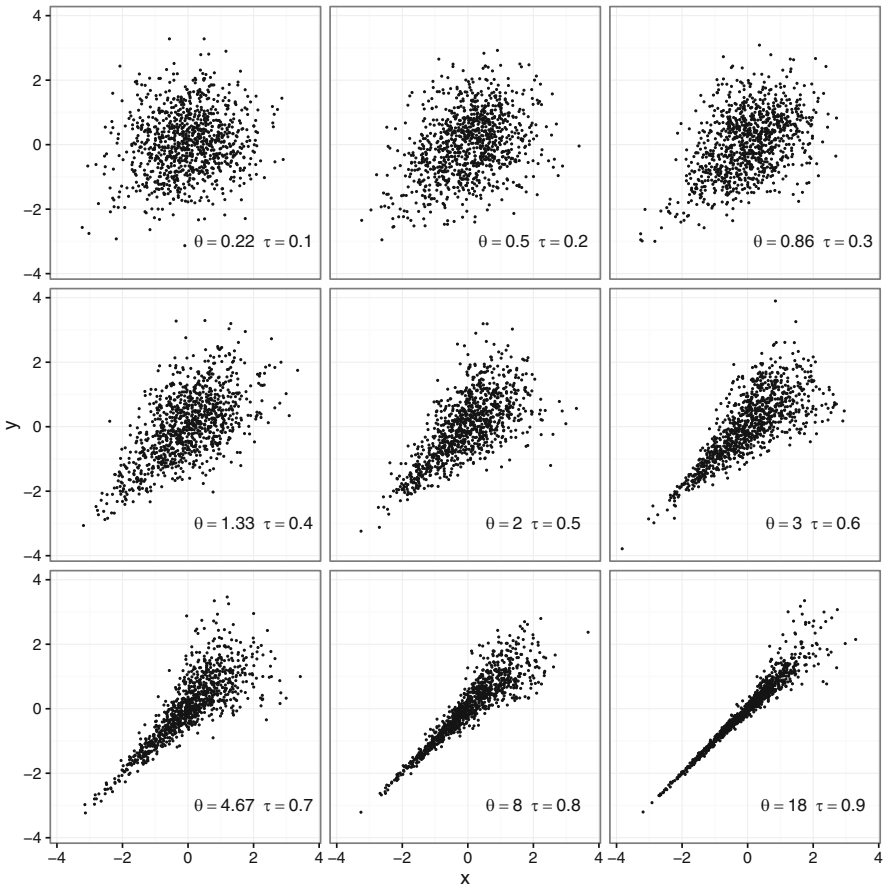

Fig. 3.11 Samples from standard normal random variables $X \sim \mathcal{N}(0, 1)$ and $Y \sim \mathcal{N}(0, 1)$ joined by a Clayton copula with several values of θ

3.2.5 Sampling from Bivariate Copulas

We have discussed how to sample from the t- and normal copulas, but these procedures do not extend to general copulas. There is a straightforward way to produce samples from a joint distribution produced by copulas. Consider the marginal CDFs for random variables X and Y, $F_X(x)$ and $F_Y(y)$, and a copula $C(u, v)$. The procedure to produce samples from the joint distribution given by $C(F_X(x), F_Y(y))$ is:

1. Produce two uniform random variables ξ_1 and ξ_2 where $\xi_i \sim \mathcal{U}(0, 1)$.
2. Set

$$w \equiv C^{-1}(\xi_2|\xi_2).$$

3. Then the samples x and y are $x = F_X^{-1}(\xi_1)$ and $y = F_Y^{-1}(w)$.

This sampling procedure is simple to perform with the possible exception of not knowing $C^{-1}(v|u)$. In this case we can use a nonlinear solver to perform the inversion.

As a demonstration we will show how this works for the Frank copula. This is a case where the inverse of the conditional CDF, $C^{-1}(v|u)$, can be explicitly calculated. For the Frank copula, we have

$$C_{\mathrm{F}}(v|u) = \frac{e^{\theta}\left(e^{\theta v} - 1\right)}{-e^{\theta} + e^{\theta+\theta u} - e^{\theta(u+v)} + e^{\theta+\theta v}}. \tag{3.35}$$

The inverse of this function is

$$C_{\mathrm{F}}^{-1}(\xi|u) = \frac{1}{\theta}\log\left(\frac{e^{\theta}\left(\xi\left(e^{\theta u} - 1\right) + 1\right)}{\xi e^{\theta u} - e^{\theta}(\xi - 1)}\right). \tag{3.36}$$

This algorithm was used to generate the samples in Fig. 3.8.

3.3 Multivariate Copulas

The idea of a copula can be extended to more than two random variables. For this discussion we will have a collection of p random variables $\mathbf{X} = (X_1, \ldots, X_p)^{\mathrm{T}}$. Each of these random variables has a known marginal CDF $F_{X_i}(x_i)$. A copula, C, on this collection of random variables is function that maps a p-dimensional vector $\mathbf{u}$ with each component in [0, 1] to a nonnegative real number. With this copula we then define a joint CDF for $\mathbf{X}$ as

$$F(\mathbf{x}) = C(F_{X_1}(x_1), \ldots, F_{X_p}(x_p)). \tag{3.37}$$

In the multivariate setting, the independence copula, C_I, is a product of each input:

$$C_I(\mathbf{u}) = \prod_{i=1}^{p} u_i. \tag{3.38}$$

The normal copula is extended in a simple manner:

$$C_{\mathrm{N}}(\mathbf{u}) = \Phi_{\mathbf{R}}(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_p)). \tag{3.39}$$

In this case the correlation matrix is of size $p \times p$. In an analogous fashion, the t-copula can be extended as well:

$$C_{\mathrm{t}}(\mathbf{u}) = F_{\mathbf{t}}(F_{\mathrm{t}}^{-1}(u_1), \ldots, F_{\mathrm{t}}^{-1}(u_p)). \tag{3.40}$$

For both of these copulas, we have already given a procedure for sampling from the joint distributions. The algorithms that we discussed earlier need to take p samples from a multivariate normal instead of two, and the rest of the algorithm proceeds naturally. The multivariate extensions of these copulas will have the correlation between variables specified by the **R** and **S** matrices.

Archimedean copulas in higher dimensions also have a natural extension. These copulas can be written as

$$C_{\varphi}(\mathbf{u}) = \hat{\varphi}^{-1}(\varphi(u_1) + \cdots + \varphi(u_p)). \tag{3.41}$$

Note that each generator needs to have the same value of θ in this definition. This means that Kendall's tau, and perhaps the tail dependence, will be the same for all the variables.

3.3.1 Sampling Multivariate Archimedean Copulas

To sample from an Archimedean copula, we will use the Marshall-Olkin algorithm. In this algorithm we need to take the inverse Laplace transform of the quasi-inverse of the generator function, $\varphi(t)$. It turns out that the inverse Laplace transform of a generator function is a cumulative distribution function. We denote the Laplace transform of the quasi-inverse of generator as $F(s) \equiv \mathscr{L}[\varphi^{-1}(t)]$. Using this cumulative distribution function, this algorithm is given as:

1. Sample $s = F^{-1}(\xi)$, where $\xi \sim \mathscr{U}(0, 1)$.
2. Create p samples $\mathbf{u}$ where $u_i \sim \mathscr{U}(0, 1)$.
3. Create p values $\mathbf{v}$ where $v_i = \varphi^{-1}(-\log(u_i)/s)$.
4. The samples from $F(\mathbf{x})$ are $x_i = F_{X_i}^{-1}(v_i)$.

For the Clayton copula with positive θ, the inverse Laplace transform of the generator yields the CDF for the gamma distribution with parameter[1] $\alpha + 1 = \theta^{-1}$ and $\beta = 1$. For the Frank copula, the function F, for positive θ, is a discrete random variable with CDF:

$$F(k) = \frac{(1 - e^{-\theta})^k}{k\theta}, \qquad k = 1, 2, \ldots.$$

[1] Here we use the notation for a gamma random variable as given in Sect. A.13.

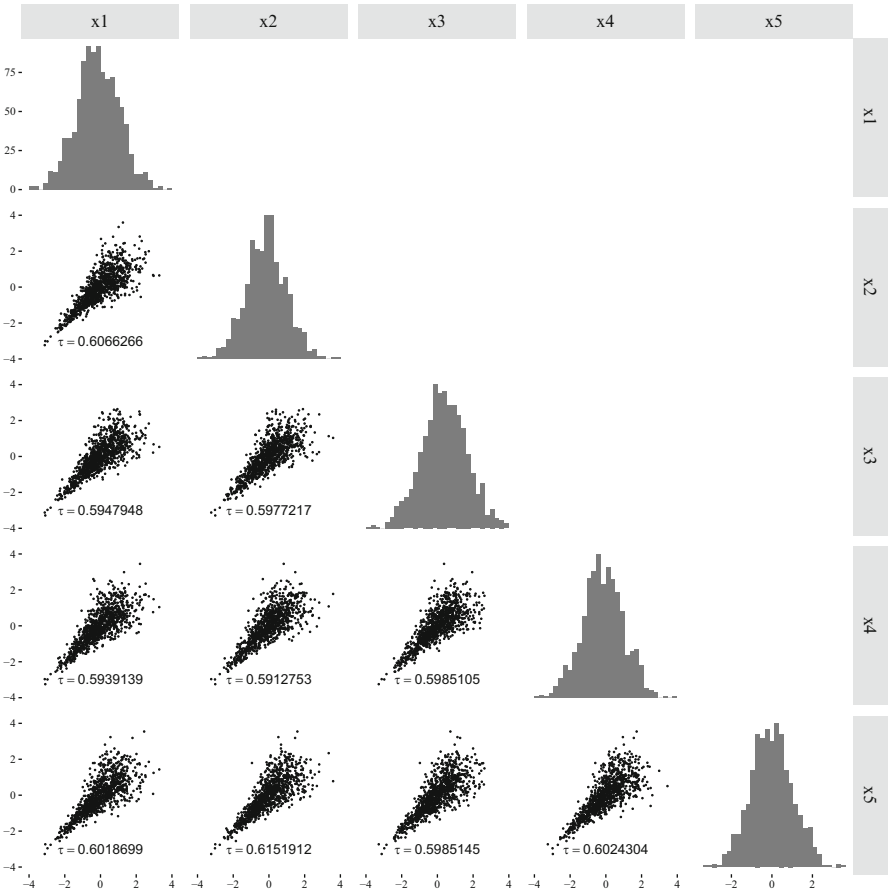

Fig. 3.12 Samples from five standard normal random variables joined by a Clayton copula with $\theta = 3$. Note how the Kendall's tau value between each pair of variables is constant

With multivariate copulas, Archimedean copulas, as we have defined them, have the same dependence between all variables, as seen in the example in Fig. 3.12. In this figure we show the 2-D projections of a 5-D joint distribution joined by a Clayton copula with $\theta = 3$. The marginal distributions are all standard normal. The case is different with a normal copula (or a t-copula) in that the correlation between variables can be different depending on the pair of variables. In Fig. 3.13 samples from a 5-D normal copula with correlation matrix are given by

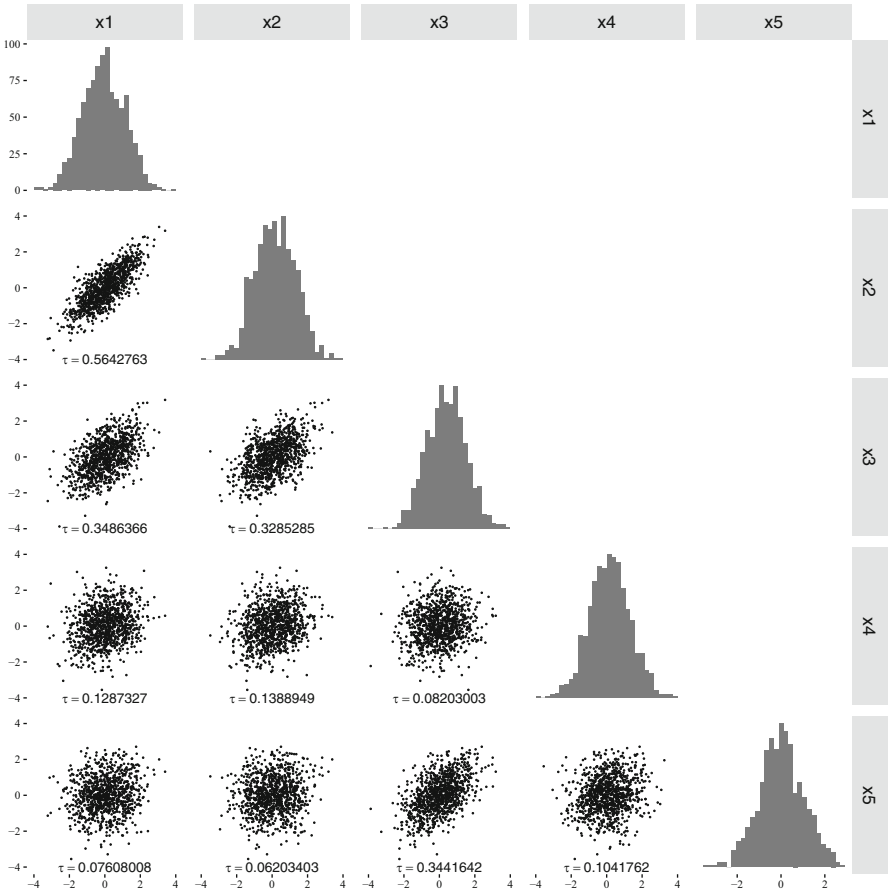

Fig. 3.13 Samples from five standard normal random variables joined by a normal copula with a correlation matrix that is not uniform. Note how the Kendall's tau value between each pair of variables is different

$$\mathbf{R} = \begin{pmatrix} 1.00 & 0.75 & 0.50 & 0.25 & 0.12 \\ 0.75 & 1.00 & 0.50 & 0.25 & 0.12 \\ 0.50 & 0.50 & 1.00 & 0.12 & 0.50 \\ 0.25 & 0.25 & 0.12 & 1.00 & 0.12 \\ 0.12 & 0.12 & 0.50 & 0.12 & 1.00 \end{pmatrix}.$$

In this case the dependence between pairs of variables does change.

3.4 Random Variable Reduction: The Singular Value Decomposition

In this section we will discuss a way to create uncorrelated random variables from a set of correlated random variables. We will do this using the singular value decomposition of the data. This procedure is known by several names, including principal component analysis, the Hotelling transform, or proper orthogonal decomposition. This procedure can be used to reduce the dimension of the data set by revealing a set of uncorrelated random variables that produce the observed correlated random variables.

Consider that we have a collection of p random variables $\mathbf{X}$, n samples of these random variables, and $n > p$. We can assemble these samples into a n by p matrix (n rows and p columns) of the form

$$\mathbf{A} = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_{p-1}^{(1)} & x_p^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_{p-1}^{(2)} & x_p^{(2)} \\ \vdots & & & & \\ x_1^{(n)} & x_2^{(n)} & \dots & x_{p-1}^{(n)} & x_p^{(n)} \end{pmatrix}. \tag{3.42}$$

We also assume that the matrix $\mathbf{A}$ is such that each row has a mean of zero; this can be done by subtracting the mean of each row from the matrix.

The matrix $\mathbf{A}$ will be rectangular in general, $n \neq p$. For such a matrix, we can factor it into what is known as the singular value decomposition (SVD):

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}. \tag{3.43}$$

In this decomposition:

- $\mathbf{U}$ is a $n \times p$ orthogonal matrix, i.e., $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$ and $\mathbf{U}\mathbf{U}^{\mathrm{T}} = \mathbf{I}$ where $\mathbf{I}$ is an identity matrix.
- $\mathbf{S}$ is a $p \times p$ diagonal matrix with nonnegative entries.
- $\mathbf{V}$ is a $p \times p$ orthogonal matrix.

The singular value decomposition is related to the eigenvalue decomposition of $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ or $\mathbf{A}^{\mathrm{T}}\mathbf{A}$. To see this we left multiply Eq. (3.42) by $\mathbf{A}^{\mathrm{T}}$ to get

$$\mathbf{A}^{\mathrm{T}}\mathbf{A} = \left(\mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}\right)^{\mathrm{T}} \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}} = \mathbf{V}\mathbf{S}\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}} = \mathbf{V}\mathbf{S}^2\mathbf{V}^{\mathrm{T}}$$

Similarly, right multiplying by $\mathbf{A}^{\mathrm{T}}$ gives

$$\mathbf{A}\mathbf{A}^{\mathrm{T}} = \mathbf{U}\mathbf{S}^2\mathbf{U}^{\mathrm{T}}.$$

Therefore, we can interpret the entries in $\mathbf{S}$ as the square root of the eigenvalues of $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ and $\mathbf{A}^{\mathrm{T}}\mathbf{A}$. Therefore, we will call these $\sqrt{\lambda_i}$ and order them in decreasing magnitude:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r,$$

where r is the number of nonzero eigenvalues of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.

Note that the matrix $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ is an approximation of the covariance matrix for the p random variables because the dot product between rows and columns is an approximation to the integral in the definition in covariance.

Given that we know how to interpret the values in the matrix $\mathbf{S}$, we can develop interpretations for the meaning of the $\mathbf{U}$ and $\mathbf{V}$ matrices. We can transform the matrix $\mathbf{A}$ into an orthogonal matrix by multiplying by $\mathbf{V}$ to get the $n \times p$ matrix:

$$\mathbf{T} \equiv \mathbf{A}\mathbf{V}.$$

The resulting matrix has columns that are linear combinations of the original columns. The columns in $\mathbf{T}$ will have zero covariance between them. This can be seen by multiplying $\mathbf{T}$ by its transpose. The matrix $\mathbf{T}^{\mathrm{T}}\mathbf{T}$ is the covariance matrix of the data matrix $\mathbf{T}$, and it is given by the diagonal matrix

$$\mathbf{T}^{\mathrm{T}}\mathbf{T} = (\mathbf{U}\mathbf{S})^{\mathrm{T}}\mathbf{U}\mathbf{S} = \mathbf{S}^2.$$

Therefore, the matrix $\mathbf{V}$ has columns that give the coefficients for a linear combination of the original variables to create p uncorrelated variables.

The rows of the matrix $\mathbf{U}$ are the values of the uncorrelated random variables created by the linear combinations defined by the columns of $\mathbf{V}$, divided by the $\sqrt{\lambda_i}$. To see this we can look at the matrix $\mathbf{T}$:

$$\mathbf{T} = \mathbf{A}\mathbf{V} = \mathbf{U}\mathbf{S},$$

or

$$\mathbf{U} = \mathbf{T}\mathbf{S}^{-1}.$$

Additionally, mean of each column in the matrix $\mathbf{U}$ is zero. Therefore, each row of $\mathbf{U}$ contains the values of p uncorrelated, mean-zero random variables.

To summarize, the SVD transforms the original data matrix, into a matrix $\mathbf{U}$ of mean-zero, uncorrelated variables that are rescaled linear combinations of the original variables. The linear combinations are defined in $\mathbf{V}$, and the scaling is given in the diagonal matrix $\mathbf{S}$.

3.4.1 Approximate Data Matrix

To examine the way the SVD works and see how we can use it to approximate the original matrix, we write it as a sum

$$\mathbf{A} = \sum_{i=1}^{r} \sqrt{\lambda_i}\mathbf{u}_i\mathbf{v}_i^{\mathrm{T}}, \tag{3.44}$$

where $\mathbf{u}_i$ is the ith column of $\mathbf{U}$ and $\mathbf{v}_i$ is the ith column of $\mathbf{V}$. Given that each term of this sum is a $n \times p$ matrix, we can write an approximation to $\mathbf{A}$ using a subset of the terms. Call the matrix using only k terms in the sum as $\mathbf{A}_k$ such that

$$\mathbf{A}_k = \sum_{i=1}^{k} \sqrt{\lambda_i}\mathbf{u}_i\mathbf{v}_i^{\mathrm{T}}.$$

It can be shown that $\mathbf{A}_k$ is the best rank k approximation to $\mathbf{A}$.

We can interpret this truncated expansion that gives $\mathbf{A}_k$ as the SVD

$$\mathbf{A}_k = \mathbf{U}\mathbf{S}_k\mathbf{V}^{\mathrm{T}},$$

where $\mathbf{S}_k$ has the first k entries of $\mathbf{S}$ and zeros afterward.

On a similar note, we can take the first k columns of $\mathbf{V}$ and call this matrix $\mathbf{V}_k$. Therefore, if we multiply $\mathbf{A}$ by this matrix, we get a $n \times k$ matrix $\mathbf{T}_k = \mathbf{A}\mathbf{V}_k$. We can interpret the columns of this matrix as k random variables that approximate the full set of p random variables.

3.4.2 Using the SVD to Reduce the Number of Random Variables

As we will see later, the number of input random variables is a strong determinant of the computational cost of performing a UQ study. In such an instance, it may be possible to use the SVD to reduce the number of input random variables. If we say that there are nominally p input random variables to our simulation and we have n samples of those random variables, we can form the matrix $\mathbf{A}$ as described above. Then we can perform the SVD on the matrix and determine how many uncorrelated variables there are. For instance, if $r < p$, then we know we can exactly represent the matrix $\mathbf{A}$ using fewer than p random variables.

We can also use the SVD to create a small number of solutions based on the numerical solutions to our model equations. It is also the case that if we have the numerical solution to our model equations at a finite number of points, we can use the SVD to represent the variability in the numerical solution with a handful

of uncorrelated random variables. Suppose we know the solution to the model equations at p points, these could be points in any number of dimensions, but we write them as a single vector. For each realization of the input random variables, we will get a different vector. Using these vectors, we can create a data matrix as described above.

Regardless of whether one wants to reduce the number of input or output random variables, it is likely that a large fraction of the variance in the data can be adequately represented by k uncorrelated random variables. We measure the fraction of variance explained to determine how many variables we need. We call the total variance in the data the sum of the λ_i from the SVD. The fraction of variance explained by the random variables in $\mathbf{T}_k$ is written as

$$s_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{r} \lambda_i}.$$

Clearly, the fraction of variance explained is 1 if k is equal to or greater than r.

It is often the case that a few uncorrelated random variables can represent the p correlated variables quite well. That is, with $k \ll r$, the value of the fraction of variance explained can be close to 1. The user can select a value of k that explains an appropriate amount of the total variance for the problem of interest. Once we have selected these k variables, we can consider these as our uncertain inputs to our model. We will demonstrate how to select these variables below.

A sketch of the procedure for using SVD to reduce the number of input variables is:

1. Select a desired fraction of variance explained, s.
2. Perform the SVD on the data matrix $\mathbf{A}$, and determine the value of k that gives a fraction of variance explained greater than or equal to s.
3. The independent random variables are given by $\mathbf{T}_k = \mathbf{A}\mathbf{V}_k$.
4. To transform to the original random variables compute $\mathbf{A}_k = \mathbf{T}_k \mathbf{V}_k^{\mathrm{T}}$.

We note that the uncorrelated variables produced by the SVD will not necessarily be a standard distribution. One exception is if the data were generated from a multivariate normal. In this case, the uncorrelated variables will be standard normal random variables. If the uncorrelated random variables are not normal, it is possible to fit a distribution.

As an example of how this works, we will consider the data matrix shown in Table 3.2. This data has $p = 9$, and $n \approx 10^4$. Given the range of units in each of these columns, we will first normalize our data so that the columns are mean 0 and standard deviation 1. That is, for each column, we subtract the column mean and divide the result by the standard deviation of the column. After this normalization, we take the SVD of the data matrix. It turns out for this data set, with $k = 4$, the fraction of variance explained is 0.9. The fraction of variance explained is shown in Fig. 3.14.

Table 3.2 Data matrix for the SVD example before normalization

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
13	46	10	0	2	24	0	1	20
45	93	16	0	17	53	0	0	62
20	46	6	2	0	14	8	5	23
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
51	87	20	2	22	60	0	3	17

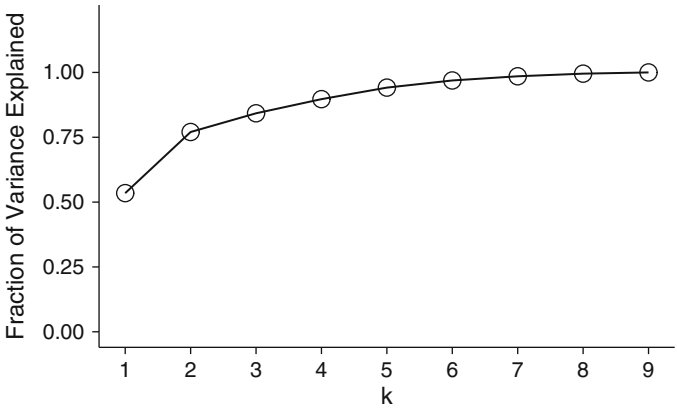

Fig. 3.14 The fraction of variance explained as a function of k for the SVD of the data in Table 3.2

When we create the approximate data matrix $\mathbf{A}_k$, we are then making an approximation to the full data set. To see how this approximation behaves, we can look at the scatter plots of X_1 versus X_2 from various approximations, $\mathbf{A}_k$. As shown in Fig. 3.15, as k increases, the reconstructed data set begins to resemble the original data set. This figure has converted the data back to the original units by multiplying each column by the standard deviation and adding the mean from the original data set.

The columns of the matrix $\mathbf{V}$, which we will call v_i for $i = 1 \ldots 9$, tell us what linear combinations of the original variables give us the uncorrelated random variables. For this example the matrix $\mathbf{V}$ is given in Table 3.3. These weights can give us an idea of what are the important features of the data.

To aid in the interpretation of the transformed variables, we plot the coefficients for the first three linear combinations (i.e., the first three columns of $\mathbf{V}$) in Fig. 3.16. From this we can see that most important uncorrelated variable has all positive coefficients, and we can think of this variable as a measure of the overall magnitude of observation i: when all the original variables are large for an observation, then this quantity will be large. Going back to our interpretation of the columns of $\mathbf{U}$, a row in the data matrix that has a large value for all the variables will have a large value in the first column of $\mathbf{U}$ on the appropriate row.

The second variable has large positive weights for X_5 and X_6 and large negative weights for X_4, X_7, and X_8. Therefore, we can interpret the variable as differentiating between those observations that have large values of X_5 and X_6 and those that

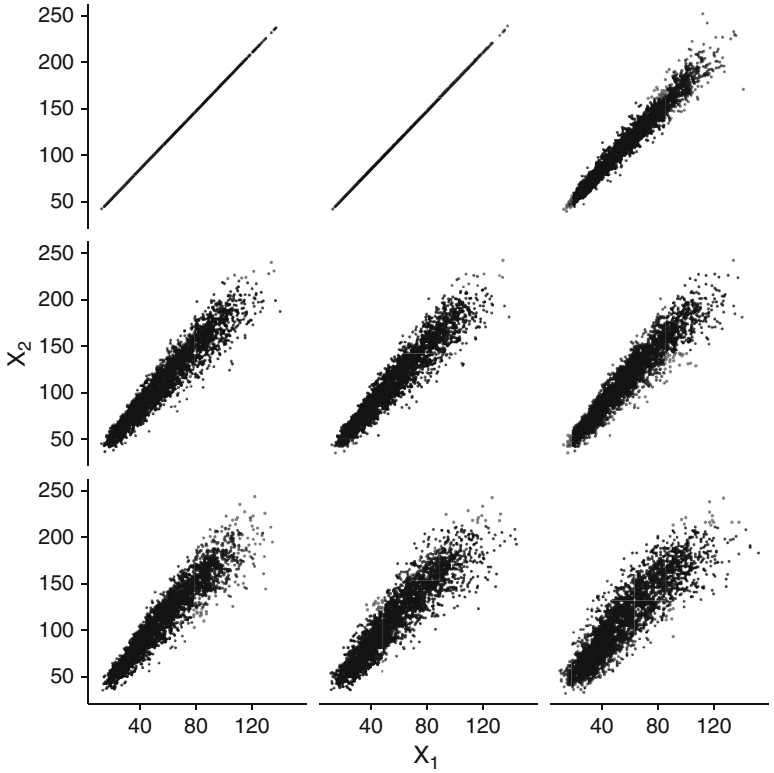

Fig. 3.15 Scatter plots of X_1 versus X_2 for $k = 1 \ldots 9$ in the original units. The top left plot is $k = 1$ and the bottom right is $k = 9$ (the original data set)

Table 3.3 The matrix **V** for the example data set

	1	2	3	4	5	6	7	8	9
X_1	0.4395	−0.0267	0.0191	−0.0276	0.0497	−0.1530	−0.2828	0.6513	0.5243
X_2	0.4219	−0.0278	−0.2149	0.2823	0.1095	0.0136	−0.6138	−0.1011	−0.5442
X_3	0.3813	0.1060	−0.2970	0.5144	0.2719	0.0045	0.6441	0.0416	−0.0051
X_4	0.1825	−0.4359	−0.6416	−0.5791	−0.0050	0.0939	0.1345	−0.0458	−0.0250
X_5	0.3303	0.3501	0.1525	−0.2686	−0.5914	−0.0170	0.2606	0.2722	−0.4253
X_6	0.3938	0.2765	−0.0438	−0.0159	−0.3143	0.0709	−0.1083	−0.6475	0.4812
X_7	0.1898	−0.5449	0.2943	0.0810	−0.1533	−0.6985	0.1308	−0.2054	−0.0570
X_8	0.1783	−0.5381	0.3456	0.2017	−0.2145	0.6827	0.0648	0.0388	0.0274
X_9	0.3437	0.1089	0.4715	−0.4472	0.6273	0.0907	0.0970	−0.1570	−0.1089

Each column gives the weights for a linear combination of the p original random variables X_i

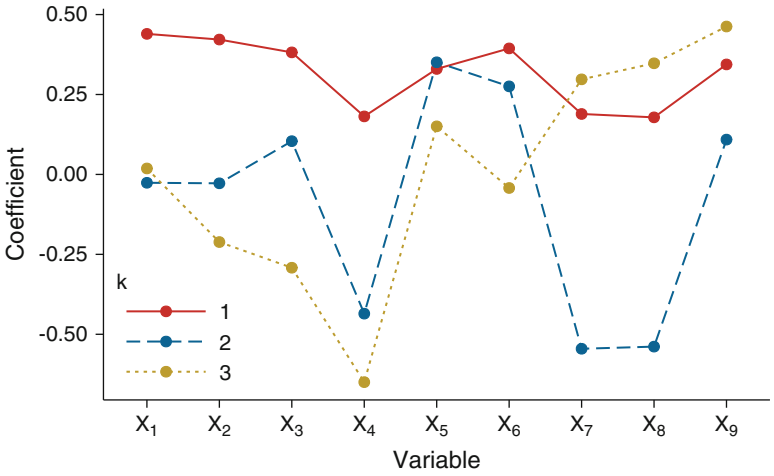

Fig. 3.16 Composition of the first three uncorrelated variables in the SVD of the normalized data matrix

have large values of X_4, X_7, and X_8. We could continue interpreting the uncorrelated random variables, but it is more important to point out that the SVD is set up so that all the variability in the data is mapped onto these uncorrelated variables.

Additional Interpretation of the Example

The data used in the above example was not contrived for the example. It is actually the season offensive statistics for Major League Baseball since 1980 for all players having over 200 at bats in a season from Lahman (2017). The variables $(X_1, \ldots, X_9)$ are:

1. Runs
2. Hits
3. Doubles
4. Triples
5. Home runs
6. Runs batted in (RBI)
7. Stolen bases
8. Times caught stealing
9. Walks

It is useful to know what the data represents so that it can aid in interpreting the variables. Given what the original variables are, we see that the first uncorrelated variable, the value in the first column of **U**, is a measure of the overall magnitude of a player's statistics. Additionally, the second uncorrelated random variable, column 2 of **U**, differentiates between those players with a high number of home runs and runs batted in, the so-called power hitters, and those that have high numbers of triples, stolen bases, and times caught stealing; these are the so-called speedsters. In the

data set, the largest value of in the second column of **U** belongs to Mark McGwire in 1998 when he hit 70 home runs in a allegedly steroid-tainted campaign. This is the most extreme power hitter in this measure. The lowest value in this column is Rickey Henderson in 1982 when he set the modern-day record for stolen bases with 130 (and was caught 42 times).

When looking at the SVD results in this light, we can see that the coefficients are telling us something about the data. In this case it tells us that one measure of a baseball player is the amount of power versus speed. These results also indicate that the SVD can be useful even when we are not looking to reduce the data because it can give us a different lens to see how a data is varying.

3.5 The Karhunen-Loève Expansion

The Karhunen-Loève expansion (KL expansion) is the analog of the SVD for a stochastic process. Recall that a stochastic process can be thought of as a collection of random variables where the number of random variables goes to infinity. In this case, we represent the stochastic process as an expansion in basis functions instead of the basis vectors in the **V** matrix in the SVD. To compute the KL expansion, we need to know only the mean function, $\mu(x)$, and covariance function, $k(x_1, x_2)$. With this knowledge we can write the KL expansion of a stochastic process $u(x; \boldsymbol{\xi})$ where $x \in [a, b]$ is the deterministic spatial variable and $\boldsymbol{\xi}$ denotes the random component as

$$u(x; \boldsymbol{\xi}) = \mu(x) + \sum_{\ell=0}^{\infty} \sqrt{\lambda_\ell} \xi_\ell g_\ell(x). \tag{3.45}$$

Notice that this form looks nearly identical to the SVD in Eq. (3.44). The ξ_ℓ are random variables with zero mean and unit variance. The ξ_ℓ are also uncorrelated, but they are not necessarily independent.

The λ_ℓ and $g_\ell(x)$ are eigenvalues and eigenfunctions of the covariance operator:

$$\int_a^b k(x, y) g_\ell(y)\, dy = \lambda_\ell g_\ell(x). \tag{3.46}$$

The functions $g_\ell(x)$ are orthonormal just like the matrix **V** was orthogonal in the SVD. Also, we order the eigenvalues as we did in the SVD case, $\lambda_1 \geq \lambda_2 \geq \ldots$, and the eigenvalues have a finite sum of squares:

$$\sum_{\ell=0}^{\infty} \lambda_\ell^2 \leq \infty.$$

Determining the eigenvalues and eigenfunctions is not a trivial task as it involves determining the spectrum of an integral operator. There are cases where the solution is known, and we will focus on these cases.

For the KL expansion to exist, there are some technical details that need to be met by the stochastic process. Firstly, it needs to be square integrable over the x domain, i.e., the integral of $u^2(x;\boldsymbol{\xi})$ must be finite. Also, the covariance function must be positive definite. If these are satisfied, the KL expansion will exist.

The KL expansion is most useful if the stochastic process is Gaussian. This is because in this case we know that the ξ_ℓ will be independent, standard normal random variables because the sum of normal random variables is normal. If the stochastic process is not normal, then we know that the ξ_ℓ must not be independent because, by the central limit theorem, the sum will limit to a normal random variable. Therefore, if the stochastic process is not Gaussian, we need more information about the ξ_ℓ.

If we restrict ourselves to an independent ξ_ℓ, we can still model non-Gaussian stochastic processes with the KL expansion. We could do this by writing the stochastic process as a nonlinear transformation of a Gaussian process. Two possible ways of doing this are with a logarithmic transform, where $\log u(x,\boldsymbol{\xi}) = \hat{u}(x,\boldsymbol{\xi})$, where $\hat{u}(x,\boldsymbol{\xi})$ is a Gaussian stochastic process. The other commonly used approach to transforming a stochastic process is the Nataf transform. This method is beyond the scope of our study, but it does allow a general stochastic process to be represented with a Gaussian stochastic process so that the KL expansion could be used.

3.5.1 Truncated Karhunen-Loève Expansion

The KL expansion turns a stochastic process into a sum over random variables. Therefore, if we truncate the expansion, we have effectively discretized it in terms of randomness: rather than infinite collection of random variables, we write the process as a finite sum of random variables with known properties. Going back to our definition of the UQ problem in Chap. 1, if we have a calculation that depends on a stochastic process as input, we can consider the ξ_ℓ as our uncertain inputs and get a map to the input stochastic process. The number of terms that we need to keep in the expansion depends on the covariance function and how fast the λ_ℓ go to zero in magnitude.

3.5.1.1 The Exponential Covariance

As we mentioned before, the determination of the eigenvalues and eigenvectors of the covariance function is not a trivial task. For a general covariance function, this can be quite difficult. There are a handful of cases where the solution is known, and here we will present the results for a simple, but useful, case. The detailed derivation of this expansion can be found in Ghanem and Spanos (1991).

If the covariance function has the form of an exponential of an absolute value,

$$k(x_1, x_2) = ce^{-b|x_1 - x_2|}, \tag{3.47}$$

we can find the eigenvalues and eigenvectors exactly. The case we will consider has $x \in [-a, a]$, but we can use these results over any domain provided we define a shifted spatial variable.

The eigenvectors for this covariance function can be expressed in terms of cosines and sines, and we will write the KL expansion in a slightly different way:

$$u(x; \boldsymbol{\xi}) = \mu(x) + \sum_{\ell=0}^{\infty} \left[\sqrt{\lambda_\ell} \xi_\ell g_\ell(x) + \sqrt{\lambda_\ell^*} \xi_\ell^* g_\ell^*(x) \right]. \tag{3.48}$$

The eigenvalues are

$$\lambda_\ell = \frac{2cb}{\omega_\ell^2 + b^2}, \qquad \lambda_\ell^* = \frac{2cb}{\omega_\ell^{*2} + b^2}, \tag{3.49}$$

where the ω_ℓ and ω_ℓ^* are solutions to the transcendental equations:

$$b + \omega \tan(\omega a) = 0, \qquad b - \omega^* \tan(\omega^* a) = 0. \tag{3.50}$$

The eigenfunctions are

$$g_\ell = \frac{\cos(\omega_\ell x)}{\sqrt{a + \frac{\sin(2\omega_\ell a)}{2\omega_\ell}}}, \tag{3.51}$$

and

$$g_\ell^* = \frac{\sin(\omega_\ell x)}{\sqrt{a - \frac{\sin(2\omega_\ell a)}{2\omega_\ell}}}. \tag{3.52}$$

The value of b has an important impact on the eigenvalues. A smaller value of b makes the eigenvalues decay to zero faster than a larger value. This is demonstrated in Fig. 3.17 where the first 10 eigenvalues of the exponential covariance are shown for several values of b. When b is large, the eigenvalues decay slowly. This implies that we can capture the behavior of the stochastic process with a few terms in the KL expansion.

To demonstrate how the KL expansion behaves as more terms are added, we show a single realization of a stochastic process in Fig. 3.18. In that figure, we see that with two terms in the KL expansion (in this case one λ^* and one λ term) give a smooth, slowly varying function. As the number of terms increases, the complexity

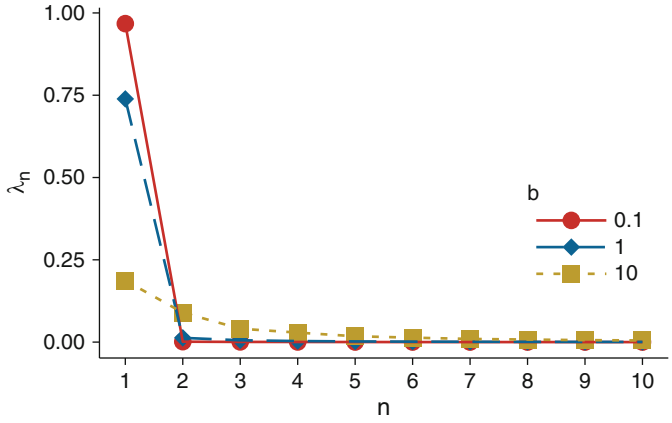

Fig. 3.17 The eigenvalues λ_n and λ_n^* for various values of b and $a = 0.5$ and $c = 1$. The odd n are λ_n^*, and even n are λ_n

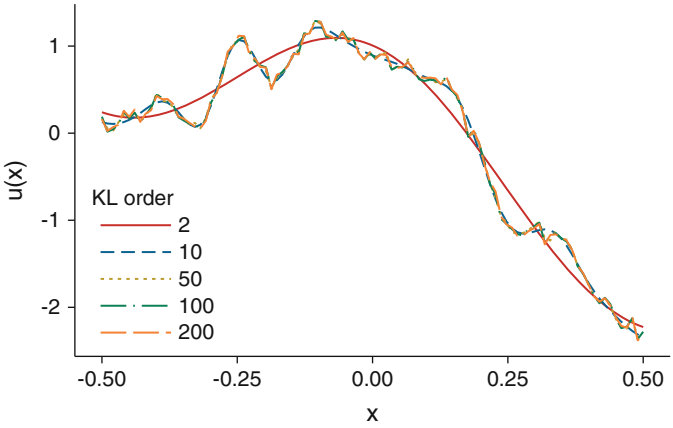

Fig. 3.18 A single realization of a Gaussian stochastic process over $[-0.5, 0.5]$ with $\mu(x) = \cos 2\pi x$ and an exponential covariance with $b = c = 1$ using various number of expansion terms

of the realization increases by having finer-scale variations in the solution: at 10 terms there is more variability in the solution, and by 100 terms, there are sharp oscillations at a very fine scale.

Another way to look at the behavior of the expansion is to compare several realizations of a stochastic process with different expansion orders. This comparison is made in Fig. 3.18 using the same stochastic process as in Fig. 3.17; samples from this process are shown in Fig. 3.19. When comparing the high expansion orders with the low-order expansions (e.g., two and ten terms), there is much less structure in the low-order expansions. However, as more and more terms are added, the character

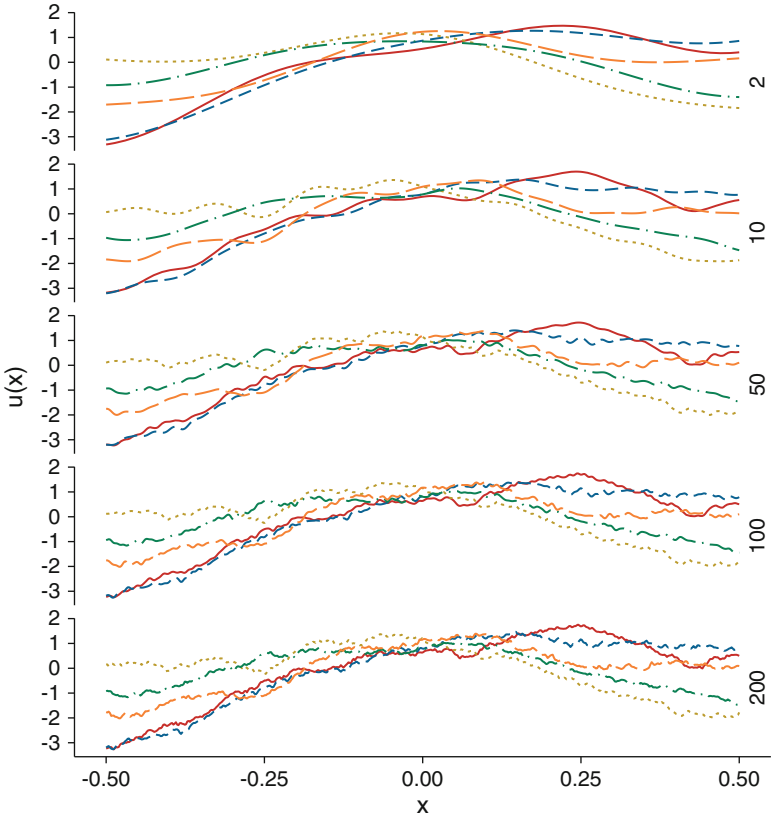

Fig. 3.19 Five realizations of a Gaussian stochastic process over $[-0.5, 0.5]$ with $\mu(x) = \cos 2\pi x$ and an exponential covariance with $b = c = 1$ at different numbers of expansion terms

of the expansion approaches that of the full process. In many cases, the fine-scale structure of the process is not what is important; rather the overall behavior is of interest. If this were the case, we would likely be able to adequately model this process with just a few terms in the expansion.

3.6 Choosing Input Parameter Distributions

One basic question regarding an uncertain parameter regards how we would like to represent that uncertainty. From the preceding chapter, we know that once we have a CDF or PDF for a random variable, we can then compute quantities like the mean, variance, and any number of other properties of the distribution. Nevertheless, it is generally not possible to have a unique mapping the other way: to go from moments

of the distribution, e.g., mean, variance, skewness, kurtosis, etc. to produce a PDF or CDF.

Unfortunately, we usually do not know the distribution of our input parameters. It is much more typical to have some number of samples from the distribution. For instance, if the system we are interested in simulating has manufactured parts and the properties of those parts have a distribution, we will be able to take a number of parts and measure the properties. This gives us samples from the distribution of the properties, from which we can estimate moments like the mean and variance. However, we cannot robustly quantify the behavior of the tails of the distribution from a small number of samples. This is because, by definition, our samples will, with high probability, not have any values from the tails of the distribution. Therefore, the best we can do is make a guess as to the tail behavior of the system. We need to acknowledge that we have made this assumption about the tail behavior of the distribution, and not make overly specific claims about the probability of a tail event is.

A common approach to modeling a random variable is to select a distribution from the standard set of distributions (such as those provided in Appendix A). There are several considerations that are important when selecting a distribution for an input random variable. For a given parameter, we want the distribution we assume it follows to be consistent with the parameter in the following regards:

1. The range, e.g., real numbers, positive real numbers, or a certain range
2. The known moments, or other properties of the distribution, e.g., mean, median, variance, or various quantiles.

The first of these conditions can eliminate many possible distributions. For instance, if we know the parameter can only take on a range of values or is positive, then we know that we cannot use a normal distribution without an ad hoc procedure for ignoring the probability of getting an invalid parameter. The known information about the parameter's behavior will also eliminate some possible distributions. As an example, if the parameter is known to possess some skewness or excess kurtosis, then a normal distribution will not be able to capture those properties. Once a distribution is chosen, then one can fit the remaining known information about the distribution. That is, select the parameters of the distribution so that the input random variable's properties are preserved.

Many times it will not be the case that all of the desired properties of the distribution can be fit with a standard distribution. It may be the case that a standard distribution is not flexible enough to reproduce the desired properties (e.g., there is a fixed relationship between moments of the distribution). In this case one could compromise and decide to not match all of the desired properties. The other possibility is to blend distributions together to get the desired properties. For instance, if the desired distribution is multimodal, i.e., it has multiple local maxima in the PDF, one could write this as the sum of normal distributions and fit the mean and standard deviation of each normal to match the desired distribution.

3.6.1 Choosing Joint Distributions

It is potentially even more complicated to choose a joint distribution for a set of inputs. We have already mentioned that in general one will not know much about the joint distribution functions for a collection of random variables. Therefore, it is typical to be less constrained in the selection of the joint distribution, and this freedom can be a double-edged sword. We have already mentioned that choosing a joint distribution, through the selection of copula, that does not have any tail dependence can lead to erroneous conclusions about the probability of the parameters going to extremes together.

One of the measures we want to match for a joint distribution is a measure of correlation between the variables. This could be any of the measures we discussed: Pearson or Spearman correlation or Kendall's tau. We also noted in our discussion of copulas that it may be possible to produce a desired tail dependence in the joint distribution. Nevertheless, it is likely not possible to match both the correlation and tail dependence. Therefore, one often has to make a decision as to which feature is more important for the analysis being performed.

If the uncertainty analysis being performed is looking for understanding the behavior of the system under conditions near the median inputs, then the tail dependence of the distribution is less important than the measure of the correlation. In such a situation, it is reasonable to choose a joint distribution without tail dependence. It is not reasonable, however, to then use this distribution to make statements about extreme events using this joint distribution.

In the case where one cares about distribution of system performance near the median inputs and also wants to make assertions of the system behavior near the tails of the distribution, it is possible to use both distributions. For instance, one could perform an analysis using a joint distribution that has zero tail dependence and use this to quantify the system behavior near the nominal inputs. Then, to predict the behavior at the extremes, use a different joint distribution that does have tail dependence. The analysis should make clear the caveat that the behavior near the nominal inputs and near the extremes was produced using different assumptions about the distributions.

3.6.2 Distribution Choice as a Source of Epistemic Uncertainty

In the selection of a distribution for input parameters, there are necessarily assumptions that are made. These assumptions are a type of epistemic uncertainty in the uncertainty modeling. For the distribution of a single parameter, i.e., its marginal distribution, the behavior of that distribution in the tails could have an impact on the conclusions of the analysis. For instance, if one is interested in the percentage of time the system's maximum temperature exceeds some threshold, one could get

an answer of 0.01% using normal distributions for the input parameters, and 0.05% using a t-distribution for the parameters. Given that we do not actually know which is the is correct distribution to use, the range 0.01–0.05% is the epistemic uncertainty in the result.

Furthermore, the assumptions on the joint distribution lead to epistemic uncertainty. For a given measure of relation between two variables, there are an infinite number of joint distributions that could match this quantity. In fact, we discussed several possible joint distributions when we discussed copulas. Each of these joint distributions has properties that could affect an uncertainty analysis. For example, both the Frank copula and the normal copula could match any particular value of Kendall's tau,, but the behavior of the joint distribution is not the same: when we look at samples from the joint distributions, Frank gives an almost rectangular distribution versus the elliptically shaped normal copula.

It is likely that the number and impact of outliers, that is, low-probability events, will be underestimated by any finite sample of a distribution. Indeed, if the distribution looks normal, except for a single sample, the analyst is likely to "ignore" that sample. One would like the prediction from an uncertainty study to be robust to the presence of outliers, but this needs to be a conscious decision of the practitioners, and distributions need to be chosen that give this property. In the same vein, tail dependence is very difficult to estimate from a set of samples. In any uncertainty study, it should be carefully considered what the implications of tail dependence are and how to conservatively estimate the impact of such a dependence.

3.7 Notes and References

The topic of principal component analysis is covered in detail by Jolliffe (2002), and proper orthogonal decomposition for numerical calculations is covered by Schilders et al. (2008). Additional discussion of copulas can be found in Kurowicka and Cooke (2006).

3.8 Exercises

1. Demonstrate that $\rho(X, Y) = \text{sign}(a)\rho(aX + b, Y)$.
2. Assume you have 100 samples of a pair of random variables (X_1, X_2) that have a positive correlation, and call this set of pairs, $\mathbf{A}_1$. You then draw another 100 samples and call this set $\mathbf{A}_2$. The Pearson correlation between (X_1, X_2) in $\mathbf{A}_1$ is positive and the Pearson correlation between (X_1, X_2) in $\mathbf{A}_2$ is negative. What can you say about the Pearson correlation for all 200 samples?
3. For the data in Table 3.4, compute by hand the Pearson and Spearman correlations and Kendall's tau.

Table 3.4 Data for Problem 2

X_1	X_2
55.01	82.94
54.87	55.02
57.17	85.18
36.01	−84.27
35.88	−106.30
36.33	−119.65
43.49	−112.03
41.44	−71.69
54.43	−3.50
36.47	140.57

4. Demonstrate the tail dependence of a bivariate normal random variable is 0.
5. Another Archimedean copula is the Joe copula with generator

$$\varphi_{\mathrm{J}}(t) = -\log\left(1 - (1-t)^{\theta}\right),$$

and

$$\varphi_{\mathrm{J}}^{-1}(t) = 1 - (1 - \exp(-t))^{1/\theta}.$$

   a. Compute the bivariate copula for this generator.
   b. Derive the upper and lower tail dependence for this copula.
   c. Compute the value of Kendall's tau for this copula
   d. Generate 1000 samples from the copula with standard normal marginals and a value of Kendall's tau of 0.6.

6. Consider the covariance function:

$$k(x_1, x_2) = \exp\left[-|x_1 - x_2|\right].$$

Generate four realizations of a Gaussian stochastic process with zero mean, $\mu(x, y) = 0$, and this covariance function defined on the unit interval, $x \in [0, 1]$. Compare these with realizations of KL expansions of this process with 1–10 terms. For the realizations, evaluate the process at 50 equally spaced points in each direction. Plot the realizations.

Part II
Local Sensitivity Analysis

Chapter 4
Local Sensitivity Analysis Based on Derivative Approximations

Maybe you might have some advice to give on how to be insensitive

—Jan Arden, *Insensitive*

In Part I of this book, we discussed the selection of quantities of interest (QoIs) and the determination of the inputs and their associated uncertainties. In this part we begin to explore the impact of those uncertainties on the QoIs. We begin with answering a simple question: given knowledge of the QoI at a particular value of the inputs, how would we expect the QoIs to vary for small, expected perturbations in the inputs? To accomplish this we are interested in the derivative of the QoI at a nominal input value.

Local sensitivity analyses typically rely on perturbations to the nominal state and, therefore, neglect most interactions between parameters on the QoI. As a result, the analysis is only applicable to perturbations around the nominal state. While we cannot make global statements about QoI behavior using a local analysis, performing a local sensitivity analysis is a useful step in an uncertainty analysis.

The importance of local sensitivity analysis is largest in situations where one wants to estimate the variability of behavior around some nominal operating conditions for the system. The variations to the nominal conditions could arise from a variety of sources. For instance, there could be parameters that have a known distribution due to uncertainties in their measurement or production. If the variabilities in these distributions are small, it may be possible to quantify the uncertainty in the QoI to these parameters using only local information.

Local sensitivity analysis can also be used to determine the parameters in a calculation that have the largest impact on the quantity of interest and, under some approximations, estimate the impact on the distribution of the QoI. Due to the nature of a local sensitivity analysis, it typically requires a smaller number of evaluations

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-319-99525-0_4) contains supplementary material, which is available to authorized users.

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0_4

of the QoI. Therefore, it is possible to use the local sensitivity analysis to screen out unimportant parameters before performing a more in-depth analysis. Such a reduction in the number of parameters will make later analyses more efficient. Nevertheless, one must be mindful that a parameter that is unimportant in one region of input space may be important in another.

As we will see in this chapter, the nominal amount of evaluations of the QoI to perform a sensitivity analysis is the number of parameters plus one. The number goes up significantly if second-order information is calculated. In later chapters we will see how these numbers can be reduced using regularized regression and adjoint techniques. We begin with the straightforward calculation of sensitivities.

4.1 First-Order Sensitivity Approximations

Consider a QoI as a function of a vector $\mathbf{x} = (x_1, x_2, \ldots x_p)$ of p parameters that are potentially random variables x_i, i.e., $Q(\mathbf{x})$. We can then expand this function in a Taylor series about some nominal value of $\mathbf{x}$ that we denote as $\bar{\mathbf{x}}$:

$$\begin{aligned} Q(\mathbf{x}) = Q(\bar{\mathbf{x}}) &+ \Delta_1 \frac{\partial Q}{\partial x_1}\bigg|_{\bar{\mathbf{x}}} + \Delta_2 \frac{\partial Q}{\partial x_2}\bigg|_{\bar{\mathbf{x}}} + \cdots + \Delta_p \frac{\partial Q}{\partial x_p}\bigg|_{\bar{\mathbf{x}}} \\ &+ \frac{\Delta_1^2}{2}\frac{\partial^2 Q}{\partial x_1^2}\bigg|_{\bar{\mathbf{x}}} + \frac{\Delta_1 \Delta_2}{2}\frac{\partial^2}{\partial x_1 \partial x_2} Q\bigg|_{\bar{\mathbf{x}}} \\ &+ \cdots + \frac{\Delta_{p-1}\Delta_p}{2}\frac{\partial^2}{\partial x_{p-1}\partial x_p} Q\bigg|_{\bar{\mathbf{x}}} + \frac{\Delta_p^2}{2}\frac{\partial^2 Q}{\partial x_p^2}\bigg|_{\bar{\mathbf{x}}} \\ &+ \text{higher-order terms.} \end{aligned} \tag{4.1}$$

In this equation $\Delta_i = x_i - \bar{x}_i$. The value of $\bar{\mathbf{x}}$ is typically chosen to be the mean or median of the uncertain parameters. We can write Eq. (4.1) in shorthand form as

$$Q(\mathbf{x}) = Q(\bar{\mathbf{x}}) + \sum_{i=1}^{p} \Delta_i \frac{\partial Q}{\partial x_i}\bigg|_{\bar{\mathbf{x}}} + \sum_{i=1}^{p}\sum_{j=1}^{p} \frac{\Delta_i \Delta_j}{2}\frac{\partial^2 Q}{\partial x_i \partial x_j}\bigg|_{\bar{\mathbf{x}}} + O(\Delta^3). \tag{4.2}$$

From this expansion, we can expect that for small variations to $\mathbf{x}$, the Taylor expansion of the QoI would give an accurate description to the behavior of the QoI to changes in the input parameters. The question remains how to estimate the derivatives in the expansion. Once we do know these derivatives, we could predict the behavior with changes to parameters.

The error in the Taylor series is only small for points close to the expansion point, $\bar{\mathbf{x}}$. As one moves away from the expansion point, the error can become very large, even if the underlying function is smooth and a high-order series is used. This can be seen in the fact that the error term is proportional to Δ to some power. Therefore, when Δ becomes large enough, the error will be large.

4.1.1 Scaled Sensitivity Coefficients and Sensitivity Indices

For the expansion in Eq. (4.2), if we neglect the second-order and higher terms, we can express the behavior of the quantity of interest using only first derivatives of the QoI. This will give us the ability to predict which parameters have a larger effect on the QoI and the expected change of the QoI to a small perturbation in a parameter. This use of derivatives to predict the behavior of a QoI is commonly called local sensitivity analysis. The first-order derivatives of the QoI are often called the first-order sensitivities of the QoI.

By ranking the sensitivities by magnitude, we can gauge which uncertain parameters are likely to have the largest impact. To compare the sensitivities, we need to cast them in the same units because, for example, the units of sensitivity i will have the inverse units of x_i. One way to do this is with *scaled sensitivity coefficients*. The scaled sensitivity coefficient for parameter i is the nominal value of parameter i, $\bar{x}_i$, multiplied by the derivative of the QoI with respect to x_i:

$$(\text{Scaled Sensitivity Coefficient})_i = \bar{x}_i \left. \frac{\partial Q}{\partial x_i} \right|_{\bar{\mathbf{x}}}. \tag{4.3}$$

This definition of the scaled sensitivity coefficient can use any nominal value of the user's choosing. Often the nominal value will be the mean, but it could be any value.

The scaled sensitivity coefficients indicate which parameters are most sensitive about a value of the parameter. This can be misleading, however, because it is possible that a parameter has a large scaled sensitivity coefficient, but a small overall uncertainty, i.e., we know that parameter to within a small degree of uncertainty. To correct this case, *sensitivity indices* are used. In this case we multiply by the characteristic range of variation of the parameter; often this is chosen to be the standard deviation of the parameter i, σ_i:

$$(\text{Sensitivity Index})_i = \sigma_i \left. \frac{\partial Q}{\partial x_i} \right|_{\bar{\mathbf{x}}}. \tag{4.4}$$

The parameter with the largest product of the derivative and the standard deviation will have the highest sensitivity index. Note that the parameter σ_i might be replaced by some other measure of the variability of the parameter about $\bar{\mathbf{x}}$.

Both of these measures of sensitivity are useful in eliminating parameters that do not appear to be important to the QoI, at least near their nominal value. The utility of such knowledge is most evident in a system where there is a large number of uncertain parameters. Knowing the sensitivities allows the UQ practitioner to narrow the focus to a smaller number of parameters and then apply the more time-consuming techniques we shall discuss later, e.g., sampling methods or polynomial chaos expansions. One must keep in mind, however, the fact that sensitivities are only local quantities and extrapolating far from the nominal value $\bar{\mathbf{x}}$ may require understanding of higher-order terms and the interactions between different parameters.

4.2 First-Order Variance Estimation

With knowledge of the first-order sensitivities, we can estimate the variance in the QoI due to the covariances in the input parameters. As the derivation will demonstrate, the variance estimate assumes that the linear Taylor series is sufficient to describe the QoI.

This calculation requires that the value of $\bar{\mathbf{x}}$ is the mean of the parameters. Recall that the variance of a random variable $Q(\mathbf{x})$ with joint PDF, $f(\mathbf{x})$, is written as

$$\begin{aligned} \text{Var}(Q) &= E[Q(\mathbf{x})^2] - E[Q(\mathbf{x})]^2 \\ &= \left(\int d\mathbf{x}\, Q(\mathbf{x})^2 f(\mathbf{x}) \right) - \left(\int d\mathbf{x}\, Q(x)\, f(\mathbf{x}) \right)^2 \\ &= \left(\int d\mathbf{x}\, Q(\mathbf{x})^2 f(\mathbf{x}) \right) - E[Q(\mathbf{x})]^2. \end{aligned} \tag{4.5}$$

To estimate the expectation of $Q(\mathbf{x})^2$, we use the first-order Taylor expansion from Eq. (4.2) and ignore the second-derivative and higher terms:

$$Q(\mathbf{x})^2 \approx Q(\bar{\mathbf{x}})^2 + \left(\sum_i (x_i - \bar{x_i}) \frac{\partial Q}{\partial x_i}\bigg|_{\bar{\mathbf{x}}} \right)^2 + 2Q(\bar{\mathbf{x}}) \left(\sum_i (x_i - \bar{x_i}) \frac{\partial Q}{\partial x_i}\bigg|_{\bar{\mathbf{x}}} \right). \tag{4.6}$$

Also, the linear first-order expansion implies that

$$E[Q(\mathbf{x})]^2 \approx Q(\bar{\mathbf{x}})^2.$$

Using the expansion from Eq. (4.6) in Eq. (4.5), we get, to second order,

$$\begin{aligned} \text{Var}(Q) = &-Q(\bar{\mathbf{x}})^2 \\ &+ \int d\mathbf{x} \left[Q(\bar{\mathbf{x}})^2 + \left(\sum_i (x_i - \bar{x_i}) \frac{\partial Q}{\partial x_i}\bigg|_{\bar{\mathbf{x}}} \right)^2 \right. \\ &\left. + 2Q(\bar{\mathbf{x}}) \left(\sum_i (x_i - \bar{x_i}) \frac{\partial Q}{\partial x_i}\bigg|_{\bar{\mathbf{x}}} \right) \right] f(\mathbf{x}). \end{aligned} \tag{4.7}$$

Notice that the integral of the $Q(\bar{\mathbf{x}})^2$ does not depend on $\mathbf{x}$,

$$\int d\mathbf{x}\, Q(\bar{\mathbf{x}})^2 f(\mathbf{x}) = Q(\bar{\mathbf{x}})^2, \tag{4.8}$$

and this will cancel the other quadratic Q term. In addition, the cross terms are linear in $\mathbf{x}$ about the mean and will integrate to zero. The remaining term to deal with is

$$\int d\mathbf{x}\, f(\mathbf{x}) \left(\sum_i (x_i - \bar{x_i}) \frac{\partial Q}{\partial x_i}\bigg|_{\bar{\mathbf{x}}} \right)^2$$
$$= \sum_i \sum_j \frac{\partial Q}{\partial x_i}\bigg|_{\bar{\mathbf{x}}} \frac{\partial Q}{\partial x_j}\bigg|_{\bar{\mathbf{x}}} \int dx_i \int dx_j\, f_{ij}(x_i, x_j)(x_i - \bar{x_i})(x_j - \bar{x_j}), \quad (4.9)$$

where the $f_{ij}(x_i, x_j)$ is the joint marginal distribution of $f(\mathbf{x})$. The integral in Eq. (4.9) is the covariance matrix that we previously defined. The covariance matrix indicates how parameters vary together and was defined in Sect. 2.3 as

$$\sigma_{ij} = \int dx_i \int dx_j\, f_{ij}(x_i, x_j)(x_i - \bar{x_i})(x_j - \bar{x_j}). \quad (4.10)$$

Therefore, if we know the covariance of the parameters, we can directly estimate the variance of our QoI as

$$\mathrm{Var}(Q) \approx \sum_i \sum_j \frac{\partial Q}{\partial x_i}\bigg|_{\bar{\mathbf{x}}} \frac{\partial Q}{\partial x_j}\bigg|_{\bar{\mathbf{x}}} \sigma_{ij}.$$

This calculation is approximate because we assumed the first-order Taylor series could approximate Q.

The formula for the variance can also be written in terms of the covariance matrix, Σ, and a vector of the sensitivities,

$$\frac{\partial Q}{\partial \mathbf{x}} = \left(\frac{\partial Q}{\partial x_1}, \ldots, \frac{\partial Q}{\partial x_p} \right)^{\mathrm{T}},$$

as

$$\mathrm{Var}(Q) \approx \frac{\partial Q}{\partial \mathbf{x}}^{\mathrm{T}} \Sigma \frac{\partial Q}{\partial \mathbf{x}}. \quad (4.11)$$

It is important to keep in mind that this formula is approximate because it contains no information about the interaction between parameters and their effects on the QoI.

4.3 Difference Approximations

As discussed above, the scaled sensitivity coefficients and sensitivity index require the derivative of the QoI with respect to each x_i. We can approximate these derivatives easily using finite differences:

$$\frac{\partial Q}{\partial x_i}\bigg|_{\bar{\mathbf{x}}} \approx \frac{Q(\bar{\mathbf{x}} + \delta_i \hat{e}_i) - Q(\bar{\mathbf{x}})}{\delta_i}, \quad (4.12)$$

where δ is a small, positive parameter and $\hat{e}_i$ is a vector that is one in the ith position. Given that we need to compute p derivatives, we need to compute the QoI at $p+1$ points (i.e., $p+1$ runs of the code): 1 for the mean value, $\bar{\mathbf{x}}$, and 1 for each of the i parameters.

This finite difference formula is known as the forward difference formula: it perturbs the nominal state in the positive, or forward, direction to estimate the derivative. Other types of finite differences include backward difference where the perturbation is in the negative direction and central difference where the parameter is adjusted forward and backward (this can be thought of as an average of the forward and backward differences). The central difference formula has the benefit of having an error that decreases at a rate of δ_i^2, compared with δ_i for forward and backward differences, at a cost of requiring two function evaluations per derivative approximation.

4.3.1 Simple ADR Example

We can use the advection-diffusion-reaction (ADR) equation to explore the application of difference approximations. In this case we will use a steady ADR equation in one-spatial dimension with a spatially constant, but uncertain, diffusion coefficient, a linear reaction term, and a prescribed, uncertain source:

$$v\frac{du}{dx} - \omega\frac{d^2u}{dx^2} + \kappa(x)u = S(x), \tag{4.13}$$

$$u(0) = u(10) = 0,$$

where v and ω are spatially constant with means

$$\mu_v = 10, \quad \mu_\omega = 20,$$

and variances

$$\mathrm{Var}(v) = 0.0723493, \quad \mathrm{Var}(\omega) = 0.3195214.$$

The reaction coefficient, $\kappa(x)$, is given by

$$\kappa(x) = \begin{cases} \kappa_l & x \in (5, 7.5) \\ \kappa_h & \text{otherwise} \end{cases}, \tag{4.14}$$

with $\mu_{\kappa_h} = 2$, $\mathrm{Var}(\kappa_h) = 0.002778142$, and $\mu_{\kappa_l} = 0.1$, $\mathrm{Var}(\kappa_l) = 8.511570 \times 10^{-6}$. The value of the source is given by

$$S(x) = qx(10 - x),$$

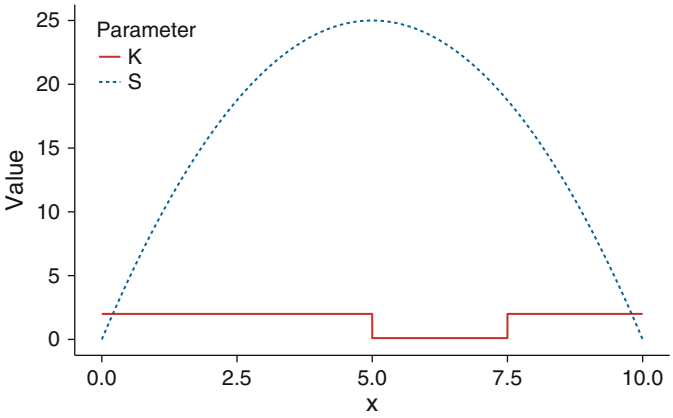

Fig. 4.1 Values of the mean function for κ and S for our ADR example

with $\mu_q = 1$, $\text{Var}(q) = 7.062353 \times 10^{-4}$. The mean functions for $\kappa(x)$ and $S(x)$ are shown graphically in Fig. 4.1. We also prescribe that the parameters ordered as $(v, \omega, \kappa_l, \kappa_u, q)$ have a correlation matrix given by

$$\mathbf{R} = \begin{pmatrix} 1.00 & 0.10 & -0.05 & 0.00 & 0.00 \\ 0.10 & 1.00 & -0.40 & 0.30 & 0.50 \\ -0.05 & -0.40 & 1.00 & 0.20 & 0.00 \\ 0.00 & 0.30 & 0.20 & 1.00 & -0.10 \\ 0.00 & 0.50 & 0.00 & -0.10 & 1.00 \end{pmatrix}. \tag{4.15}$$

The QoI for this example will be the total reaction rate in the problem:

$$Q = \int_0^{10} dx\, \kappa(x) u(x). \tag{4.16}$$

At the nominal values of parameters, that is, evaluating v, ω, κ_l, κ_h, and q at their mean values, the solution $u(x)$ is shown in Fig. 4.2. Using a solution with 2000 equally spaced spatial zones, we get $Q(\mu_v, \mu_\omega, \mu_{\kappa_l}, \mu_{\kappa_h}, \mu_q) = 52.390$. The python code used to produce these solutions is given in Algorithm 4.1.

For our ADR example, we will compute the sensitivities to each parameter using the same mesh used above ($\Delta x = 0.005$). For each parameter we compute the derivative using $\delta_i = \mu_i \times 10^{-6}$. The results from the six simulations needed to compute the five sensitivities are shown in Table 4.1. Based on the scaled sensitivity coefficient and the sensitivity index, q has the largest sensitivity. Also, both measures indicate that κ_h is the second most important parameter. This table gives many digits for each number so that we can compare with other approaches for computing the derivatives in later chapters.

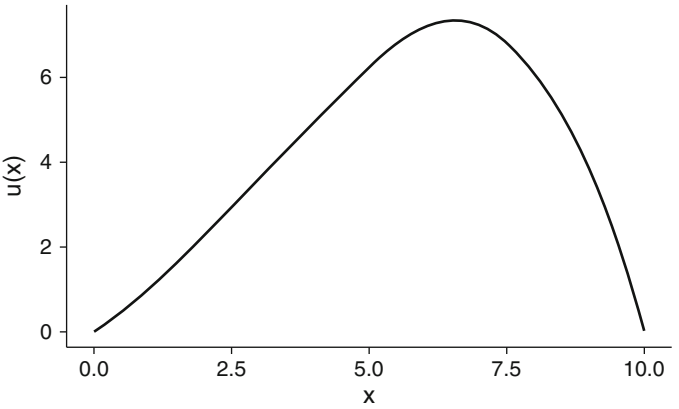

Fig. 4.2 The solution $u(x)$ evaluated at the mean value of the uncertain parameters

Algorithm 4.1 Numerical method to solve the advection-diffusion-reaction equation

```
import numpy as np
import scipy.sparse as sparse
import scipy.sparse.linalg as linalg
def ADRSource(Lx, Nx, Source, omega, v, kappa):
    A = sparse.dia_matrix((Nx,Nx),dtype="complex")
    dx = Lx/Nx
    i2dx2 = 1.0/(dx*dx)
    #fill diagonal of A
    A.setdiag(2*i2dx2*omega + np.sign(v)*v/dx + kappa)
    #fill off diagonals of A
    A.setdiag(-i2dx2*omega[1:Nx] +
              0.5*(1-np.sign(v[1:Nx]))*v[1:Nx]/dx,1)
    A.setdiag(-i2dx2*omega[0:(Nx-1)] -
              0.5*(np.sign(v[0:(Nx-1)])+1)*v[0:(Nx-1)]/dx,-1)
    #solve A x = Source
    Solution = linalg.spsolve(A,Source)
    Q = np.sum(Solution*kappa*dx)
    return Solution, Q
```

Table 4.1 Sensitivities to the five parameters in the ADR reaction rate

Parameter	Sensitivity	Scaled sensitivity coef.	Sensitivity index
v	−1.7406	−17.406	−0.46819
ω	−0.97020	−19.404	−0.54842
κ_l	12.868	1.2868	0.037542
κ_h	17.761	35.523	0.93616
q	52.390	52.390	1.3923

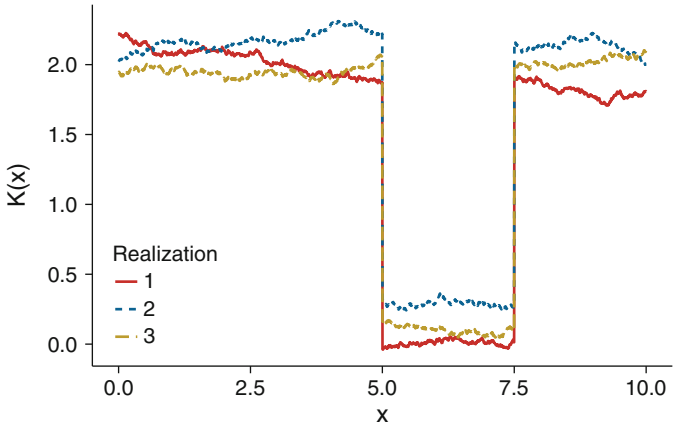

Fig. 4.3 Three realizations of the random process version of $\kappa(x)$ at 2000 points

The variance in Q can be estimated from Eq. (4.11) using the data from the sensitivity column in Table 4.1 and forming the covariance matrix using the given variances and correlation matrix for the parameters (see Eq. (3.3)). This estimate of the variance gives $\text{Var}(Q) \approx 2.0876$. If we assume that the parameters are a multivariate normal, the actual variance in Q, as estimated via Monte Carlo[1] with 4×10^4 code runs, is 2.0699, a difference of about 8.5%. This result indicates that for this problem the variance estimate is a reasonable approximation to the true variance.

4.3.2 Stochastic Process Example

We can significantly increase the size of the parameter space by making the parameter κ be a Gaussian stochastic process with mean function given by Eq. (4.14) and covariance function given by

$$k_\kappa(x_1, x_2) = 0.025e^{-0.1|x_1 - x_2|}.$$

The case of a stochastic process which is a function of a spatial coordinate is often referred to as a *random field*. For this problem, the value of κ in each spatial zone is a parameter. Three realizations of κ for this process are shown in Fig. 4.3.

In the numerical study below, the other parameters, v, ω, and q, will be fixed at their nominal values. Therefore, using 2000 spatial zones as before gives $p = 2000$. To compute the sensitivities, we need to compute the value of Q 2001 times to get

[1] The use of Monte Carlo to estimate the variance in a QoI will be discussed in Chap. 7.

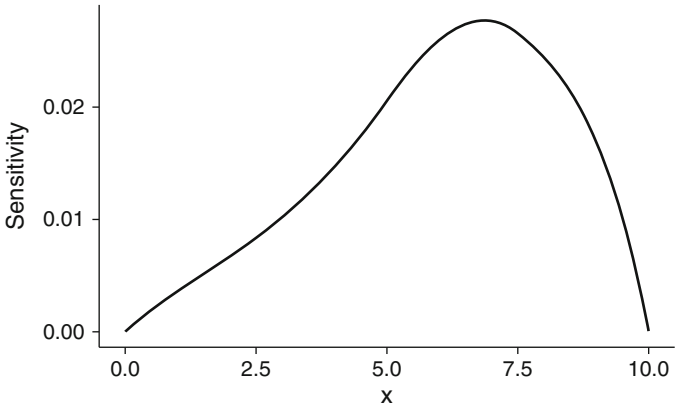

Fig. 4.4 Sensitivity of Q to $\kappa(x)$ using finite differences and 2001 solutions to the ADR equations

the sensitivity via finite differences. For each parameter we perturb by a relative amount of 10^{-6} and get the sensitivities to Q to the value of κ in each cell. These sensitivities are shown in Fig. 4.4. Notice that the sensitivity looks similar to the solution at the mean function of κ as given in Fig. 4.2. We will explore this connection further when we discuss adjoint methods in Chap. 6.

To estimate the variance using Eq. (4.11), we form a covariance matrix as

$$\Sigma_{ij} = k(x_i, x_j),$$

where x_i and x_j are the centers of the ith and jth mesh cell, respectively. The variance estimate from Eq. (4.11) gives $\text{Var}(Q) \approx 18.672$, compared with a Monte Carlo estimate of 19.049 using 4×10^4 realizations of κ. This difference is only about 2%: this indicates that even when there are a large number of parameters, the first-order variance estimate can be accurate.

4.3.3 Complex Step Approximations

The finite difference formula given in Eq. (4.12) could be replaced with other common finite difference formulas, such as a central difference formula. Alternatively, the complex step formula from Lyness and Moler (1967) could be used if the underlying function is analytic. This method computes the derivative by perturbing the parameter with an imaginary perturbation to compute a second-order approximation to the derivative as

$$\left.\frac{\partial Q}{\partial x_i}\right|_{\bar{\mathbf{x}}} = \frac{\mathscr{I}\left\{Q(\bar{\mathbf{x}} + \mathrm{i}\delta_i \hat{e}_i)\right\}}{\delta_i} + O(\delta_i^2), \tag{4.17}$$

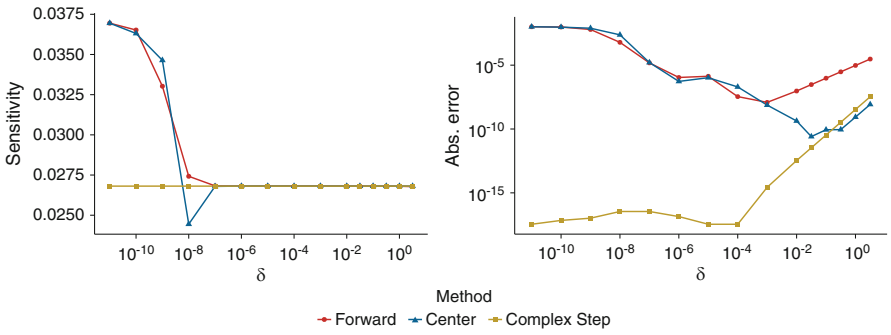

Fig. 4.5 Sensitivity of Q to $\kappa(6.25)$ using forward difference, centered difference, and complex step methods for different values of δ

where $\mathrm{i} = \sqrt{-1}$ and $\mathscr{I}\{\cdot\}$ denotes the imaginary part of the argument. It has been demonstrated that this method can produce derivative approximations as accurate as the floating point arithmetic on a particular computer will allow. This is because the approximation does not take a small difference divided by a small number, which can amplify small round-off errors.

To use the complex step method, the computer code must be able to appropriately handle complex arithmetic, which is not the usual case. However, if this method is available for use, it can be a powerful technique because it can save one evaluation of the code by not requiring the evaluation of $Q(\bar{\mathbf{x}})$, and the derivatives can be approximated more accurately. As an example of this, Fig. 4.5 demonstrates how different derivative approximations to the sensitivity of Q to the value of $\kappa(6.25)$ perform. In this figure we see that as $\delta \to 0$ the methods converge to a different answer. Initially, when δ is still greater than 10^{-5}, the methods seem to be converging to the same point, but eventually precision errors in the finite difference calculation dominate. This occurs even when central differences are used for the derivative approximation.

4.4 Second-Derivative Approximations

It seems natural to extend the approximation of the local sensitivity to include the second derivatives in the Taylor series approximation of Q. The number of extra terms in the expansion is p^2. To estimate these terms, we need to evaluate Q at more points. For the derivatives that are the second derivative with respect to an individual value, the simplest formula is

$$\left.\frac{\partial^2 Q}{\partial x_i^2}\right|_{\bar{\mathbf{x}}} \approx \frac{Q(\bar{\mathbf{x}} + \delta_i \hat{e}_i) - 2Q(\bar{\mathbf{x}}) + Q(\bar{\mathbf{x}} - \delta_i \hat{e}_i)}{\delta_i^2}. \tag{4.18}$$

This formula is second-order accurate in δ_i. When the first-order sensitivities are estimated via forward differences, as in Eq. (4.12), these second derivatives will require an additional p evaluations of Q.

The cross-derivative terms will require more function evaluations to approximate. A basic formula for these derivatives is

$$\left.\frac{\partial^2 Q}{\partial x_i \partial x_j}\right|_{\bar{\mathbf{x}}} \approx \frac{Q(\bar{\mathbf{x}}+\delta_i\hat{e}_i+\delta_j\hat{e}_j)-Q(\bar{\mathbf{x}}+\delta_i\hat{e}_i-\delta_j\hat{e}_j)-Q(\bar{\mathbf{x}}-\delta_i\hat{e}_i+\delta_j\hat{e}_j)+Q(\bar{\mathbf{x}}-\delta_i\hat{e}_i-\delta_j\hat{e}_j)}{4\delta_i\delta_j}. \tag{4.19}$$

Upon inspection of this formula, we see that each of the terms in the numerator is not contained in either Eq. (4.12) or Eq. (4.18). Therefore, each cross-derivative term requires four new function evaluations. For the $p(p-1)$ cross-derivative terms, there will be $2p(p-1)$ additional evaluations of Q; a factor of two is saved by symmetry due to the fact that

$$\frac{\partial^2 Q}{\partial x_i \partial x_j} = \frac{\partial^2 Q}{\partial x_j \partial x_i}.$$

This is a large number of additional code runs to estimate the second-order sensitivities.

In toto the computation of the entire second-order Taylor series expansion will require $2p^2+1$ total evaluations of Q. This is comprised of $p+1$ evaluations for the first derivatives, p for the simple second derivatives, and $2p(p-1)$ evaluations for the cross-derivative terms.

As an example of computing the second-derivative sensitivities, we turn to the ADR solution from Sect. 4.3.1. To compare the second-order sensitivities, we present in Table 4.2 a second-order version of the scaled sensitivity coefficients where the second derivative is multiplied by the mean value of each of the parameters in question. The table uses $\delta_i = 10^{-4}\mu_i$ as the finite difference parameter. In this table we see that the largest second-derivative sensitivity is the second derivative of Q with respect to κ_h, whereas the smallest is the second derivative with respect to q. The fact the second-derivative sensitivity of q is zero can be explained by the fact that the response of Q to the source strength is linear. The large second derivative of κ_h indicates that when κ_h is increased, there is a second-order effect that causes the increase to be less than linear. This is due to the fact that increasing κ_h does decrease the overall solution despite increasing the overall reaction rate.

Table 4.2 Second-derivative scaled sensitivity coefficients for the five parameters in the ADR reaction rate

	ν	ω	κ_l	κ_h	q
ν	3.56	–	–	–	–
ω	19.38	9.36	–	–	–
κ_l	−0.14	−0.47	0.07	–	–
κ_h	−5.40	−8.89	−0.64	−20.60	–
q	−17.40	−19.40	1.29	35.52	−0.00

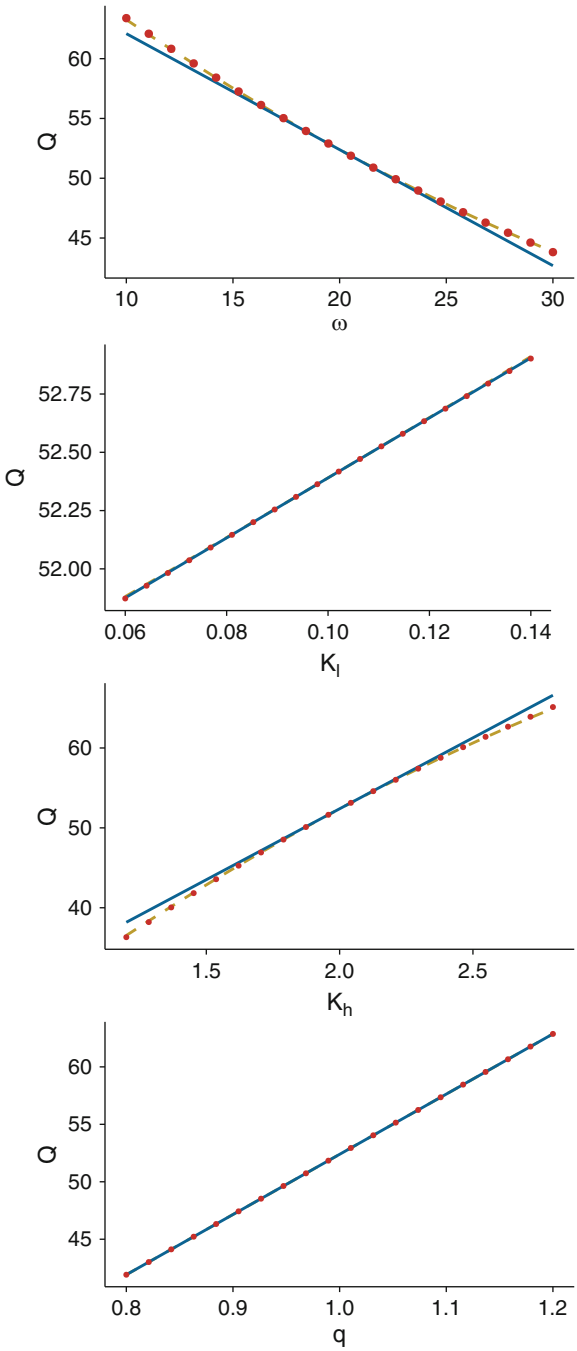

Fig. 4.6 Value of Q as a function of ω, κ_h, κ_l, and q compared to the first- and second-order Taylor series expansions. The symbols are the actual values of Q, the solid line is the linear approximation, and the dashed line is the quadratic approximation

The nonlinear behavior of Q as a function of four of the parameters is demonstrated in Fig. 4.6. Here we can see that for small perturbations from the nominal value of $\kappa_h = 2$, the linear expansion is a good approximation, but at larger perturbations, including "perturbations" on the order of 50%, the quadratic term is necessary to explain the change in Q; ω also has nonlinear effects for larger perturbations. The variation of Q with respect to q is exactly linear and κ_l is approximately linear over a similar relative range.

Of course for parameters with a weak second-derivative sensitivity, these terms are not needed to describe the behavior of Q. If we knew ahead of time which parameters would have large second derivatives, we could only perform the evaluations of Q required to compute those derivatives. It seems reasonable to suggest that those terms with large first-derivative sensitivities would have important second-derivative sensitivities. This is largely the case in this example: q and κ_h were the most sensitive parameters from the first-derivative analysis, and κ_l was the least important. With the exception of q having a zero value for Q's second derivative, the important first-derivative sensitivities were important in the second derivatives. In other words, we could have not calculated the second and mixed derivatives that have κ_l without losing much in terms of the overall response of Q.

In this chapter we have presented the local sensitivity analysis based on Taylor series approximations to the QoI. This approach allowed us to estimate which variables were important, the variance in the QoI, and value of the QoI at different inputs. These estimates require knowledge of the derivatives of the QoI with respect to the parameters. In the next two chapters, we present methods to estimate these derivatives without finite differences.

4.5 Notes and References

Automatic differentiation, also known as algorithmic differentiation, is another approach to determining the derivative of a QoI to a parameter by applying derivative rules to each step in the source code that produces a numerical result. Griewank and Walther (2008) cover this method in detail. There are other methods that do not rely on the local approximation of a Taylor series to determine sensitivity. A class of these sensitivity estimates are known as variance-based sensitivities or Sobol indices. These are based on a decomposition of the variance into fractions that are due to certain sets of inputs. The estimates are based on generating samples of the QoI from samples of the inputs. See Saltelli et al. (2010, 2008).

4.6 Exercises

1. Consider a quantity of interest that depends on a single, normally distributed parameter. Using a second-order Taylor series expansion of Q about the mean of the parameter, derive a formula to estimate the variance in Q.
2. Repeat the previous exercise with Q now depending on a p-dimensional multivariate normal random variable.
3. Compute the first-order sensitivity parameters for the example in Sect. 4.3.1 using $\delta_i = \mu_i \Delta$ where $\Delta = 10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}$ and μ_i is the mean of parameter i.
4. Using a discretization of your choice, solve the equation

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = D\frac{\partial^2 u}{\partial x^2} - \omega u,$$

for $u(x, t)$ on the spatial domain $x \in [0, 10]$ with periodic boundary conditions $u(0^-) = u(10^+)$ and initial conditions

$$u(x, 0) = \begin{cases} 1 & x \in [0, 2.5] \\ 0 & \text{otherwise} \end{cases}.$$

The time interval for the problem is $t \in [0, 5]$. Use the solution to compute the total reactions in a particular part of the domain:

$$\int_5^6 dx \int_0^5 dt\, \omega u(x, t).$$

Compute scaled sensitivity coefficients and sensitivity indices for normal random variables:

a. $\mu_v = 0.5, \sigma_v = 0.1$,
b. $\mu_D = 0.125, \sigma_D = 0.03$,
c. $\mu_\omega = 0.1, \sigma_\omega = 0.05$,

Also, estimate the variance in the total reactions. How do these results change with changes in Δx and Δt?

Chapter 5
Regression Approximations to Estimate Sensitivities

Wo! Nemo, toss a lasso to me now!

—Dona Smith

In the previous chapter, we introduced the concept of local sensitivities as derivatives of the QoI at some nominal point. We indicated how finite differences can be used to estimate the first derivatives, at the cost of $p + 1$ simulation runs where p is the number of parameters. For the second-derivative sensitivities, the number of calculations increases considerably. In practice, not all p first-derivative sensitivities are significant; almost certainly not all the second-order sensitivities will be important either.

In this chapter we give some methods that attempt to automatically select the parameters that the QoI is sensitive to. These methods will be based on extensions to the common method of linear regression. We begin by casting the sensitivity equations as a regression problem.

5.1 Least-Squares Regression for Sensitivity

Consider a QoI, $Q(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_J)^{\mathrm{T}}$ of J parameters.[1] We are interested in the first-order sensitivities about some nominal point $\bar{\mathbf{x}}$. We also have I calculations of the QoI: $Q_i = Q(\mathbf{x}_i)$.

Using the linear Taylor series expansion of Q, we can write out I equations that relate the known values of Q_i and $\mathbf{x}_i$ to the unknown sensitivities

$$Q_i := Q(\mathbf{x}_i) \approx Q(\bar{\mathbf{x}}) + (x_{i1} - \bar{x}_1)\frac{\partial Q}{\partial x_1}\bigg|_{\bar{\mathbf{x}}} + (x_{i2} - \bar{x}_2)\frac{\partial Q}{\partial x_2}\bigg|_{\bar{\mathbf{x}}} + \cdots + (x_{iJ} - \bar{x}_J)\frac{\partial Q}{\partial x_J}\bigg|_{\bar{\mathbf{x}}},$$

[1] We have switched the notation for number of parameters here so that when we form matrices the indices will be the common i and j for row and column, respectively.

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0_5

which, using gradient notation, allows us to write the total collection of data as

$$\begin{aligned} Q_1 &:= Q(\mathbf{x}_1) \approx Q(\bar{\mathbf{x}}) + \nabla Q(\bar{\mathbf{x}})(\mathbf{x}_1 - \bar{\mathbf{x}}), \\ Q_2 &:= Q(\mathbf{x}_2) \approx Q(\bar{\mathbf{x}}) + \nabla Q(\bar{\mathbf{x}})(\mathbf{x}_2 - \bar{\mathbf{x}}), \\ &\vdots \\ Q_I &:= Q(\mathbf{x}_I) \approx Q(\bar{\mathbf{x}}) + \nabla Q(\bar{\mathbf{x}})(\mathbf{x}_I - \bar{\mathbf{x}}). \end{aligned}$$

where the subscripts are ordered so that x_{ij} is the value of input j for the ith evaluation of Q. We can rearrange equations so that it can be written in the shorthand form

$$\mathbf{X}\boldsymbol{\beta} = \mathbf{y}, \tag{5.1}$$

where the matrix $\mathbf{X}$ has entries given by

$$X_{ij} = (x_{ij} - \bar{x}_j),$$

the right-hand side vector, $\mathbf{y}$ is

$$\mathbf{y} = \begin{pmatrix} Q_1 - Q(\bar{\mathbf{x}}) \\ Q_2 - Q(\bar{\mathbf{x}}) \\ \vdots \\ Q_I - Q(\bar{\mathbf{x}}) \end{pmatrix},$$

and the vector $\boldsymbol{\beta}$ contains the sensitivities:

$$\boldsymbol{\beta} = \begin{pmatrix} \frac{\partial Q}{\partial x_1}\Big|_{\bar{\mathbf{x}}} \\ \frac{\partial Q}{\partial x_2}\Big|_{\bar{\mathbf{x}}} \\ \vdots \\ \frac{\partial Q}{\partial x_J}\Big|_{\bar{\mathbf{x}}} \end{pmatrix}.$$

The vector $\mathbf{y}$ is often called the dependent variables and $\mathbf{X}$ styled the data matrix of independent variables.

The natural reaction is to seek $\boldsymbol{\beta}$ by solving Eq. (5.1). Of course, unless $I = J$, i.e., $\mathbf{X}$ is a square matrix, there is not a unique solution or necessarily even a solution. Therefore, we need at least $J + 1$ simulations to estimate the sensitivities.[2] The

[2]The extra solve comes from needing to compute $Q(\bar{\mathbf{x}})$.

implication is that we need to do as much work as using finite differences to estimate the sensitivities.

In the case where $I > J$, the problem does, however, resemble the classical linear regression problem where we have an overdetermined system to determine the coefficients of an assumed functional relationship between independent and dependent variables. In this case $\boldsymbol{\beta}$ is found by forming the normal equations by multiplying by $\mathbf{X}^{\mathrm{T}}$ and solving the resulting square system

$$\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^{\mathrm{T}}\mathbf{y}, \tag{5.2}$$

to make the coefficient vector

$$\hat{\boldsymbol{\beta}}_{\mathrm{LS}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}, \tag{5.3}$$

where the hat denotes that this is not solution to Eq. (5.1), rather an approximation. We note that the system in Eq. (5.2) is called the system of normal equations, and they will only have a solution if $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is full rank. Additionally, in practice the normal equations are not formed and then solved; typically QR factorization or the SVD is used to find $\hat{\boldsymbol{\beta}}_{\mathrm{LS}}$.

This approximation given by Eq. (5.3) has the often useful property that it minimizes the total squared error over the data. In particular one can show that the solution given by Eq. (5.3) is equivalent to the solution to the minimization problem of finding the $\boldsymbol{\beta}$ that minimizes the sum of the squared error:

$$\hat{\boldsymbol{\beta}}_{\mathrm{LS}} = \min_{\boldsymbol{\beta}} \frac{1}{2}\sum_{i=1}^{I}(y_i - \boldsymbol{\beta}\cdot\mathbf{x}_i)^2, \tag{5.4}$$

where $\mathbf{x}_i$ is the ith row of the data matrix. The subscript "LS" denotes that this is the least-squares solution. As noted above this is the solution we can obtain when $I > J$, that is when the number of simulations is greater than the number of parameters—a case that is not useful to our goal of reducing the number of simulations required to estimate the sensitivities.

5.2 Regularized Regression

We have discussed that the ordinary least-squares regression approach cannot be used to estimate sensitivities when the number of simulations, I, is less than the number of variables, J. The reason that we cannot use this approach is that there are several possible values of $\boldsymbol{\beta}$ that satisfy Eq. (5.1) because the number of degrees of freedom is greater than the number of constraints.

To select a unique value of $\boldsymbol{\beta}$, we will need to change the minimization problem to further constrain it. The minimization problem is said to be regularized by adding

an additional term to minimize. There are many possible regularizations, and we will talk about three here.

Before covering regularizations, we will modify our problem slightly. In particular we will normalize the data matrix and the coefficients so that they are dimensionless. In particular we write the data matrix as

$$X_{ij} = \frac{(x_{ij} - \bar{x}_j)}{\bar{x}_j}, \tag{5.5a}$$

and the coefficients are then scaled sensitivity coefficients:

$$\boldsymbol{\beta} = \begin{pmatrix} \bar{x}_1 \frac{\partial Q}{\partial x_1}\Big|_{\bar{\mathbf{x}}} \\ \bar{x}_2 \frac{\partial Q}{\partial x_2}\Big|_{\bar{\mathbf{x}}} \\ \vdots \\ \bar{x}_J \frac{\partial Q}{\partial x_J}\Big|_{\bar{\mathbf{x}}} \end{pmatrix}. \tag{5.5b}$$

A normalization is necessary because the regularized regression techniques that we discuss attempt to force the coefficients to be small in magnitude, if possible. Therefore, if the coefficients were not dimensionless, an important sensitivity could be set to zero based solely on the units we measured it in. Though we chose a normalization that makes the coefficients in the regression problem scaled sensitivity coefficients, we could, alternatively, have normalized the data matrix by the standard deviation of each parameter. This would have made the coefficient sensitivity indices.

5.2.1 Ridge Regression

A simple regularized regression method is ridge regression (Hoerl and Kennard 1970) which adds a penalty term based on the Euclidean norm (e.g., the 2-norm) of the coefficients. In particular the ridge minimization problem is

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \min_{\boldsymbol{\beta}} \sum_{i=1}^{I} (y_i - \boldsymbol{\beta} \cdot \mathbf{x}_i)^2 + \lambda \|\boldsymbol{\beta}\|_2, \tag{5.6}$$

where we write a p-norm as

$$\|\mathbf{u}\|_p = \left(\sum_{i=1}^{I} |u_i|^p \right)^{1/p}. \tag{5.7}$$

This norm can also be referred to as the L_p norm.

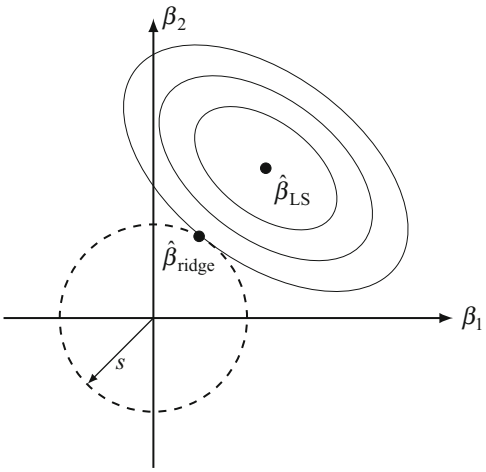

Fig. 5.1 Depiction of the ridge regression result for a two-parameter problem compared with least squares. The ellipses are the surfaces of equal value of the sum of the squared error in the regression estimate. Given that the error has a quadratic form, the ellipses further away from $\hat{\beta}_{\text{LS}}$ have a larger error. The circle is $\beta_1^2 + \beta_2^2 = s^2$. The ridge regression solution occurs where an ellipse touches the circle

This new problem seeks a value of $\boldsymbol{\beta}$ that minimizes the sum of the squares of the data while also minimizing the 2-norm of the coefficients. To put it more colloquially, the goal is to get a value of $\boldsymbol{\beta}$ that matches the data as well as possible while also having a small magnitude. Another name for the ridge regularization is Tikhonov regularization.

An equivalent formulation of the regression problem casts the regularization as a constraint rather than a penalty.[3] In particular, this form is

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \min_{\boldsymbol{\beta}} \sum_{i=1}^{I} (y_i - \boldsymbol{\beta} \cdot \mathbf{x}_i)^2 \quad \text{subject to} \quad \|\boldsymbol{\beta}\|_2 \leq s. \tag{5.8}$$

This form helps us to visualize how ridge regression works, and we will do so by considering a system with $J = 2$ as shown in Fig. 5.1. The cost function to be minimized in Eq. (5.8) is a quadratic function in the two coefficients. The contours of this quadratic function will appear as ellipses in the (β_1, β_2) plane. In the center of these ellipses is the LS estimate.

The circle in Fig. 5.1 has a radius s, and the solution must lie inside or on this circle. Because the LS estimate is outside the circle, the solution will be where the circle intersects a contour line of the sum of the squared error at the minimal possible value of the error. Notice that the magnitude of both β_1 and β_2 has decreased in the ridge estimate compared with the LS estimate and that both are nonzero.

The solution to the ridge problem can be shown to be equivalent to the solution to the system

[3]The constraint form can be changed into the penalty form by considering λ as a Lagrange multiplier. There is a one-to-one relationship between λ and s.

$$\left(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}\right)\boldsymbol{\beta} = \mathbf{X}^{\mathrm{T}}\mathbf{y}, \tag{5.9}$$

where $\mathbf{I}$ is an identity matrix of size $J \times J$. This system will always have a solution for $\lambda > 0$.

A feature of ridge regression is that the larger the value of λ, the smaller the values in $\hat{\boldsymbol{\beta}}_{\text{ridge}}$ will be in magnitude relative to the values of $\hat{\boldsymbol{\beta}}_{\text{LS}}$, when the LS values exist. This makes λ a free parameter that must be chosen based on another consideration. We will discuss using cross-validation to choose this parameter later.

As a simple example of ridge regression, consider the problem of estimating a function of the form $y = ax + b$ given the data $y(2) = 1$. That is, we are interested in fitting a line to single data point. This problem is formulated as

$$\mathbf{X} = (2, 1), \qquad \boldsymbol{\beta} = (a, b)^{\mathrm{T}}, \qquad \mathbf{y} = 1.$$

Using these values in Eq. (5.9), we get

$$\begin{pmatrix} 4+\lambda & 2 \\ 2 & 1+\lambda \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}.$$

The solution to this equation is

$$a = \frac{2}{\lambda + 5}, \qquad b = \frac{1}{\lambda + 5},$$

for $\lambda > 0$. From this we can see that the limit of this solution as λ approaches zero from the right is

$$\lim_{\lambda \to 0^+} a = \frac{2}{5}, \qquad \lim_{\lambda \to 0^+} b = \frac{1}{5}.$$

Notice that for $\lambda > 0$, the fitted solution does not pass through the data, that is, $2a + b \neq 1$, as the original data showed.

In this example we can see that we can fit a line to single data point, but the result is not necessarily faithful to the original data, but it does give us a means to fit a solution when $I < J$. This property will be useful for estimating local sensitivities when we have fewer QoI evaluations than parameters.

5.2.2 *Lasso Regression*

The ridge prescription can be modified to make the penalty be the 1-norm of the coefficients. The resulting solution is known as the least absolute shrinkage and selection operator, often shortened to lasso (Tibshirani 1996). This approach tends

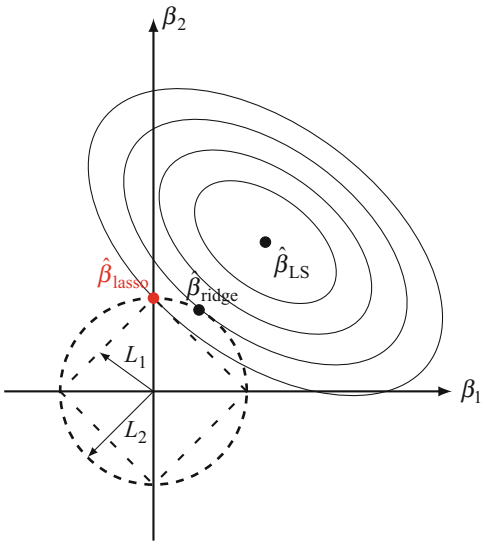

Fig. 5.2 Comparison of lasso and ridge regression result for a two-parameter problem compared with least squares. The diamond shape is the curve $|\beta_1| + |\beta_2| = s$. With lasso the solution is where the ellipse touches the diamond

to set several coefficients to be zero and, as such, "lassos" the important coefficients. The lasso problem is given by

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \min_{\boldsymbol{\beta}} \sum_{i=1}^{I} (y_i - \boldsymbol{\beta} \cdot \mathbf{x}_i)^2 + \lambda \|\boldsymbol{\beta}\|_1. \tag{5.10}$$

The small difference between ridge and lasso is in the choice of norm for the penalty. It would seem that this small difference would not make a major difference in the result. Nevertheless, the introduction of the 1-norm makes the solution to the problem more difficult, and the L_1 penalty tends to set some of the coefficients to zero. Making some of the coefficients zero is said to produce a sparse model or, alternatively, a model that is parsimonious in that it does not include variables that are not important.

The property of setting some of the coefficients to zero is precisely what we would like our method to do in the case where many of the sensitivities are small and there are a few, large nonzero sensitivities. In fact, lasso will select at most I nonzero coefficients when $I < J$. The property of setting certain coefficients to zero is demonstrated in Fig. 5.2. The L_1 norm has a diamond-shaped level curve with the points of the diamond on the axes. Therefore, the intersection between the L_1 penalty and the level curves of the squared residuals is likely to be on an axis. This is indeed what happens in Fig. 5.2. Notice that the sum of the squares of the error in the lasso solution will be larger than that for ridge because the point is on an ellipse that is farther away from the least-squares solution. This will not always be the case, however.

The solution to the lasso problem is more difficult because the minimization problem is now non-quadratic because the derivative of the L_1 norm is not smooth

due to the singular derivative of the absolute value at zero. The problem, however, is still a convex optimization problem, and there are numerical methods for efficiently solving convex optimization problems.

5.2.3 *Elastic Net Regression*

Elastic net regression (Zou and Hastie 2005) is a combination of the ridge and lasso penalties that keeps some of the sparsity of lasso but gives more accurate predictions. The elastic net minimization problem is

$$\hat{\boldsymbol{\beta}}_{\text{el}} = \min_{\boldsymbol{\beta}} \sum_{i=1}^{I} (y_i - \boldsymbol{\beta} \cdot \mathbf{x}_i)^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2^2. \tag{5.11}$$

The elastic net solution does promote sparseness in the coefficient vector, like lasso, but it is not limited to find only I nonzero coefficients. The elastic net tends to set groups of coefficients to nonzero values, as we will see in an example below.

In practice, the trade-off between the L_1 and the L_2 penalty is quantified by the parameter α:

$$\alpha = \frac{\lambda_1}{\lambda_1 + \lambda_2}.$$

This definition implies that $\alpha = 1$ is equivalent to lasso and $\alpha = 0$ is equivalent to ridge. Elastic net regression is illustrated graphically in Fig. 5.3: the curve of equal

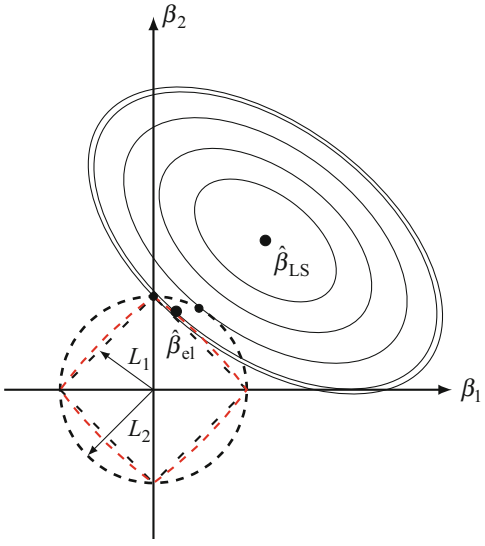

Fig. 5.3 Comparison of elastic net with $\alpha = 0.6$, lasso, and ridge regression result for a two-parameter problem compared with least squares. The curve between the diamond and circles is the curve $\alpha(|\beta_1| + |\beta_2|) + (1 - \alpha)\sqrt{\beta_1^2 + \beta_2^2} = s$. The resulting solution is in between the ridge and lasso solutions

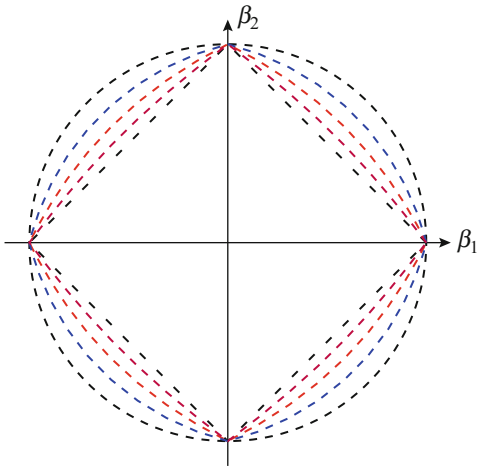

Fig. 5.4 Curves of equal values of $\alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2$ for $\alpha = 0, 0.25, 0.5, 0.75, 1$, in order of increasing α where $\alpha = 0$ is the outer circle

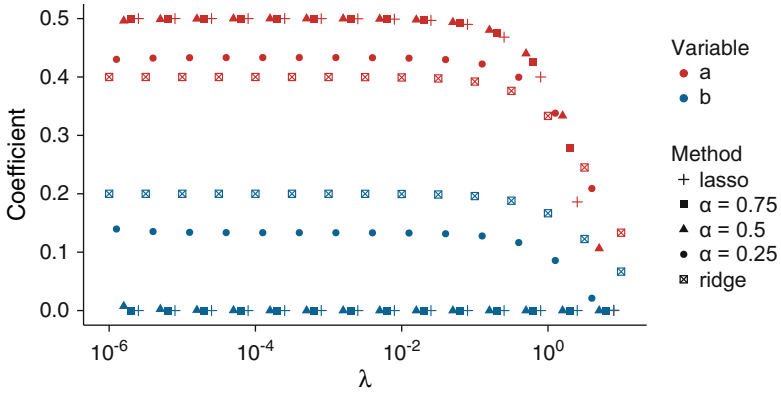

Fig. 5.5 Comparison of three methods on the problem of fitting $y = ax + b$ to the given data $y(2) = 1$ as a function of λ

values of $\alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2$ is between the diamond of L_1 and the circle of the Euclidean norm. As α approaches one, the solution moves from the ridge result to the lasso value of β.

The effect of the elastic net penalty is further illustrated in Fig. 5.4 where the curves of equal value for $\alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2$ two independent variables are shown. As α is decreased from 1 to 0, the curves transition from the circle of the L_2 norm to the diamond of the L_1 norm. During the transition the points on the axes stay the same, which gives the curves for $0 < \alpha < 1$ a blunted point on the axes (i.e., the curve is still smooth here for $\alpha < 1$). This transition gives elastic net the ability to find sparse solutions but also allows non-sparse solutions to be found.

It is worthwhile to compare elastic net and lasso to the ridge result on the simple problem of fitting $y = ax + b$ to the given data $y(2) = 1$. In Fig. 5.5 the results

are shown as a function of λ. For the elastic net regression results, $\lambda_1 = \alpha\lambda$ and $\lambda_2 = (1-\alpha)\lambda$. In this figure we can see that even for small values of λ, the lasso and ridge results are different: the lasso result sets $a = 0.5$ and $b = 0$. This is the "sparsity" we referred to before. Additionally, on this problem with a small number of coefficients, the elastic net results with $\alpha = 0.75$ and 0.5 are nearly identical to the lasso results. When α is reduced to 0.25, the result is something between the ridge and lasso. Note that both the lasso and ridge result with a small value of λ exactly match the data.

5.3 Fitting Regularized Regression Models

The regularized regression models we have discussed have a λ parameter that needs to be selected. To choose these parameters, we use a method called cross-validation. In cross-validation we split the data repeatedly into training and test sets. To illustrate this we imagine that we choose the first 90% of the data points and call this the training set. We fit a regularized regression model with several different values of λ to this training data and test each fit to the test portion of the data, recording the mean-squared error at each value of λ. We repeat this ten times, called the number of folds, randomly selecting the data that is contained in the test set each time. At the end of the procedure, we have ten values of the error at each value of λ we tested. Using these values of the error, we can compute a mean value for the mean-squared error at each value of λ as well as the standard deviation and the standard error of the mean. We could then choose the value of λ with the smallest mean mean-squared error, or more typically we select the largest value of λ with a mean value of the mean-squared error that is within one standard error of the minimum mean value of the mean-squared error. For elastic net regression where we have to pick a value of α, we can perform cross-validation on both α and λ.

Cross-validation can be run with different number of folds. The largest number of folds is equal to the number of data points. This is called leave-one-out cross-validation: each test set has only one data point, and the training data has $I - 1$ points. This type of cross-validation mimics the scenario where we have a data set and want to predict the next data point. Leave-one-out cross-validation is especially useful when the data set is small to avoid fitting the model with too small a number of data points.

We also have to specify how to select the points to evaluate the QoI. Let us assume that the number of evaluations that can be afforded is set at I. We are then faced with selecting the I values of the vector $\mathbf{x}$. This is a problem of the design of a computer experiment that will be covered in a later chapter. Suffice it to say that these points should be selected randomly, possibly using a stratified sampling technique such as Latin hypercube sampling.[4]

[4] Latin hypercube designs are covered in Sect. 7.2.2.

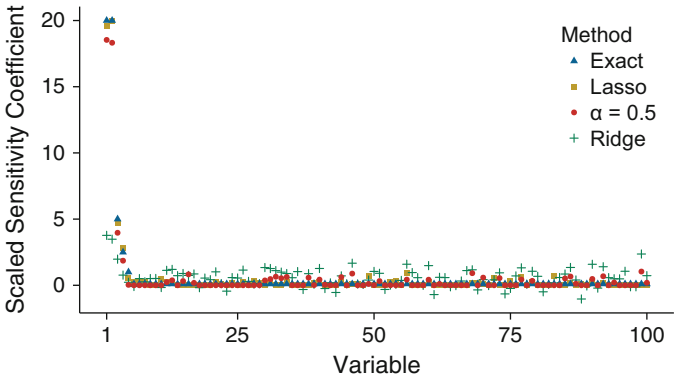

Fig. 5.6 Comparison of the different regularized regression methods to estimate the scaled sensitivity coefficients using 40 evaluations of a QoI with 100 parameters

As an example of a realistic scenario, let us assume that we have a simulation that has 100 input parameters (in our notation $J = 100$). Let us further assume that the QoI obeys the formula

$$Q(\mathbf{x}_i) - Q(\bar{\mathbf{x}}) = 20\left(\frac{x_{i1} - \bar{x}_1}{\bar{x}_1}\right) + 20\left(\frac{x_{i2} - \bar{x}_2}{\bar{x}_2}\right) + 5\left(\frac{x_{i3} - \bar{x}_3}{\bar{x}_3}\right) + 2.5\left(\frac{x_{i4} - \bar{x}_4}{\bar{x}_4}\right) + \left(\frac{x_{i5} - \bar{x}_5}{\bar{x}_5}\right) + 0.1\sum_{j=6}^{100}\frac{x_{ij} - \bar{x}_j}{\bar{x}_j} + \epsilon,$$

where $\epsilon \sim N(0, \sigma^2 = 0.01^2)$. For this QoI, only the scaled sensitivity coefficients for the first five variables is large: the contribution to the QoI from variables 6 through 100 is small. If we knew this ahead of time, we could do finite difference only on these five variables. Of course, we would typically not know this information.

Let us assume that we can afford 40 simulations. We choose 40 values of $\mathbf{x}_i$ using Latin hypercube sampling and fit a lasso model using cross-validation. With the value of λ selected based on a leave-one-out cross-validation and selecting the largest λ with a mean value of the mean-squared error within one standard error of the minimum error, we get $\lambda = 0.0055$. The results for the scaled sensitivity coefficients are given in Fig. 5.6. In this figure we can see that lasso does correctly pick the five largest scaled sensitivity coefficients, with a slight inaccuracy in the actual values of the coefficients. For the many small sensitivities, it does not give the correct value of 0.1 but instead estimates several coefficients to be nonzero and larger than 0.1. On the whole, this is a positive result: we have correctly identified the important parameters when the number of simulations was much smaller than the number of parameters.

Repeating the process for ridge and elastic net with $\alpha = 0.5$, we obtain the other results in Fig. 5.6. In these results we see that ridge has difficulty with this problem: it underestimates the large coefficients and says that far too many variables have a significant coefficient. Elastic net regression with $\alpha = 0.5$ gives results similar to lasso with smaller estimated values for the large coefficients, but unlike lasso gives more nonzero coefficients to the variables with a low sensitivity. These results indicate that on a problem like this, where $I < J$ and there are many variables that have a sensitivity near zero, lasso or elastic net regression is superior to the ridge.

On another type of problem, we can see the benefit of elastic net with a value of α close to 0. In Sect. 4.3.2 we defined an ADR problem where the reaction rate coefficient, κ, was a Gaussian process. As a result a problem with N_x spatial cells had nominally N_x parameters. In Sect. 4.3.2 we used $N_x = 2000$, and as a result, the finite difference estimates of the first-order sensitivities required 2001 simulations to calculate. In this case we solve the same problem and use a Latin hypercube design to sample from the random process the values of κ to run in the simulation. The results for the scaled sensitivity coefficients using 100 sample points and different regularized regression techniques, with λ chosen using leave-one-out cross-validation, are shown in Fig. 5.7. In the figure the finite difference estimates are shown as a reference. These results represent 5% of the number of evaluations of Q required to estimate the points via finite difference. One feature of the finite difference estimates is that they are nonzero everywhere. In this case lasso gives us a sparse solution with few nonzero coefficients. At the opposite end of the spectrum, the ridge result captures the overall trend of the finite difference estimates without exact quantitative agreement. For example, ridge correctly predicts that the most significant values are on the edge of the region where κ is low. The elastic net results with a small value of α have similar values to the ridge results, with amplified oscillations in the parameters. As α grows the elastic net result transitions to the lasso value.

To really stress the regression methods, we can reduce the number of simulations used to estimate the coefficients to 10, i.e., 0.5% of the number of simulations required for finite difference estimation. The results, shown in Fig. 5.8, indicate that, once again, lasso does not properly estimate the sensitivities. In this case a small amount of L_1 penalty in the elastic net ($\alpha = 10^{-3}$) gives the best estimates; the ridge result gives larger amplitude oscillations in the estimates on the left side of the problem and a flat behavior on the right side.

The following conclusions are in order. On problems with large number of parameters with a small number of significant sensitivities, lasso regression or elastic net regression with α near 1 can estimate the sensitivity coefficients. On the other hand, if there is correlation between the sensitivities, as there was spatial correlation between the sensitivities in the random process example, ridge or elastic net with a small value of α gives adequate estimate of the qualitative behavior of the sensitivity, even when the number of simulations is two orders of magnitude smaller than the number of parameters. Nevertheless, some user judgment is required to estimate to which situation a given problem corresponds.

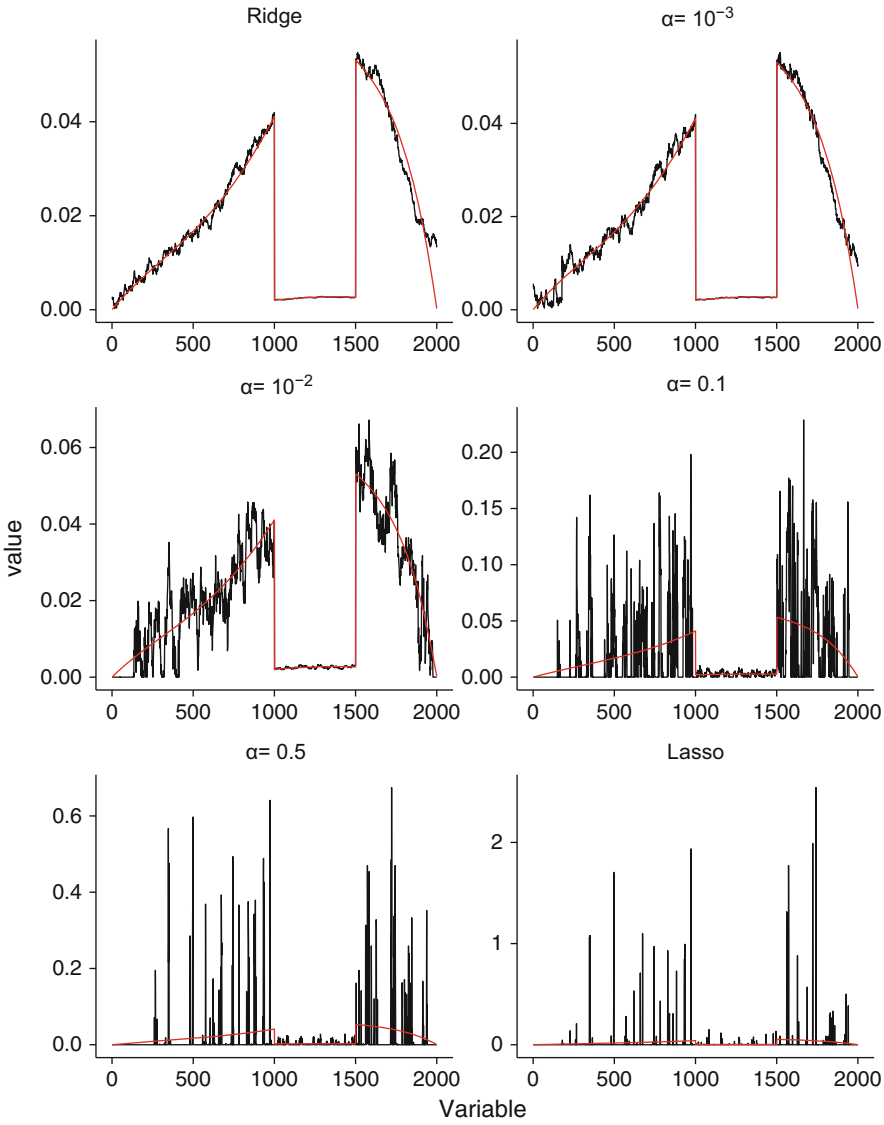

Fig. 5.7 Estimates of the scaled sensitivity coefficients for the ADR problem with ω defined by a random process evaluated at 2000 points using several regression techniques and a 100 point Latin hypercube design. The smooth, solid lines indicate the finite difference estimates

In many applications the quantitative estimate of the sensitivities is less important than the ranking of the sensitivities in order of magnitude. To accomplish this, the regression approach is clearly adequate given our results. In both tests that we performed, one of the regression methods was able to select the most sensitive variables with many fewer evaluations of the QoI than finite differences. In the next

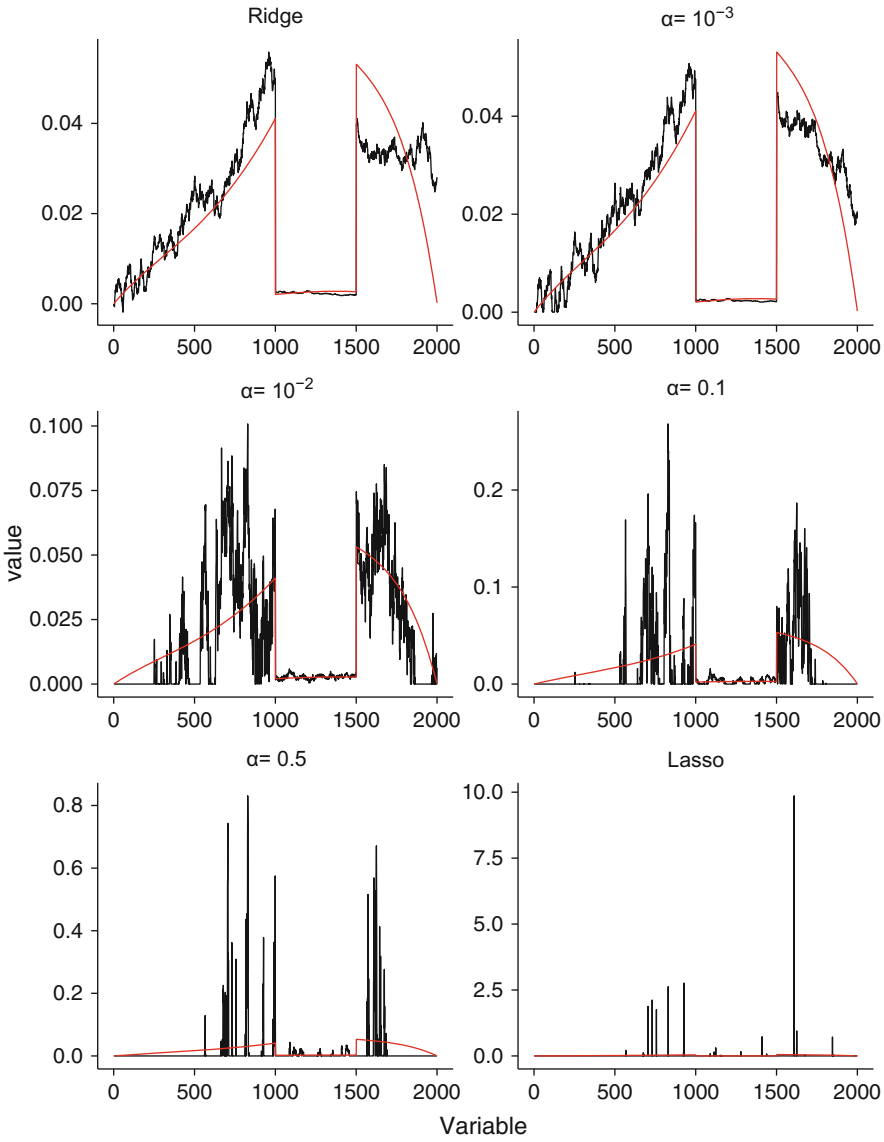

Fig. 5.8 Estimates of the scaled sensitivity coefficients for the ADR problem with ω defined by a random process evaluated at 2000 points using several regression techniques and a 10 point Latin hypercube design. The smooth, solid lines indicate the finite difference estimates

chapter, we will cover adjoint-based techniques that allow an arbitrary number of parameters to be estimated by performing two simulations. The trade-off is that adjoints are an intrusive technique meaning that we need more than just the ability to evaluate the QoIs to estimate the sensitivities.

5.3.1 Software for Regularized Regression

In the examples above, I used the `glmnet` library for R to fit the regression models. This library has a built-in cross-validation function to choose λ and a reasonable user interface. It can fit elastic net models and, therefore, ridge and lasso regression models as well. For python, the `sklearn` library has an elastic net function in the `sklearn.linear_model` module.

5.4 Higher-Derivative Sensitivities

The regression approach can be used to estimate the second derivatives of Q, including the mixed derivatives of two parameters. These derivatives are expensive to estimate via finite differences, so the regularized regression approach can give large savings over finite differences. As with the first-order sensitivities, we consider I evaluations of Q where there are J parameters. The I equations relating the first and second derivatives to the computed values of Q via Taylor series are

$$Q_i - Q(\bar{\mathbf{x}}) = \sum_{j=1}^{J} (x_{ij} - \bar{x}_j) \frac{\partial Q}{\partial x_j}\bigg|_{\bar{\mathbf{x}}} + \frac{1}{2} \sum_{j=1}^{J} (x_{ij} - \bar{x}_j)^2 \frac{\partial^2 Q}{\partial x_j^2}\bigg|_{\bar{\mathbf{x}}} + \sum_{j=1}^{J} \sum_{j'=1}^{j-1} (x_{ij} - \bar{x}_j)(x_{ij'} - \bar{x}_{j'}) \frac{\partial^2 Q}{\partial x_j \partial x_{j'}}\bigg|_{\bar{\mathbf{x}}}. \tag{5.12}$$

We have ignored the higher-order correction terms in this equation. We can write Eq. (5.12) as regression system so that the coefficients, β, are the scaled sensitivity coefficients by making the entries in the data matrix $\mathbf{X}$ have the appropriate scaled values. A common mistake is to forget the inclusion of the factor of one-half in the single-variable second derivatives.

As with the first-derivative sensitivities, we can estimate the sensitivities in Eq. (5.12). In this equation there are $\frac{1}{2}J(J+3) = \frac{J^2}{2} + \frac{3J}{2}$ sensitivities to estimate: J first derivatives, J single-variable second derivatives, and $\frac{1}{2}J(J-1)$ terms from the mixed-variable second derivatives. In this case standard least-squares regression may require fewer function evaluations than finite difference, where previously we showed that the number of function evaluations was $2J^2+1$. Therefore, it is possible to save a large number of simulations relative to finite difference without needing regularized regression.

We can apply regression to the ADR problem we solve in Sect. 4.4. This problem had $J = 5$ and, therefore, 20 total first- and second-derivative sensitivities and requires 51 function evaluations for finite difference. Using regression, and a Latin hypercube design covering $\pm 10\%$ around the nominal values of the parameters, we computed the estimates in Fig. 5.9. Given that there are 20 sensitivities, 20

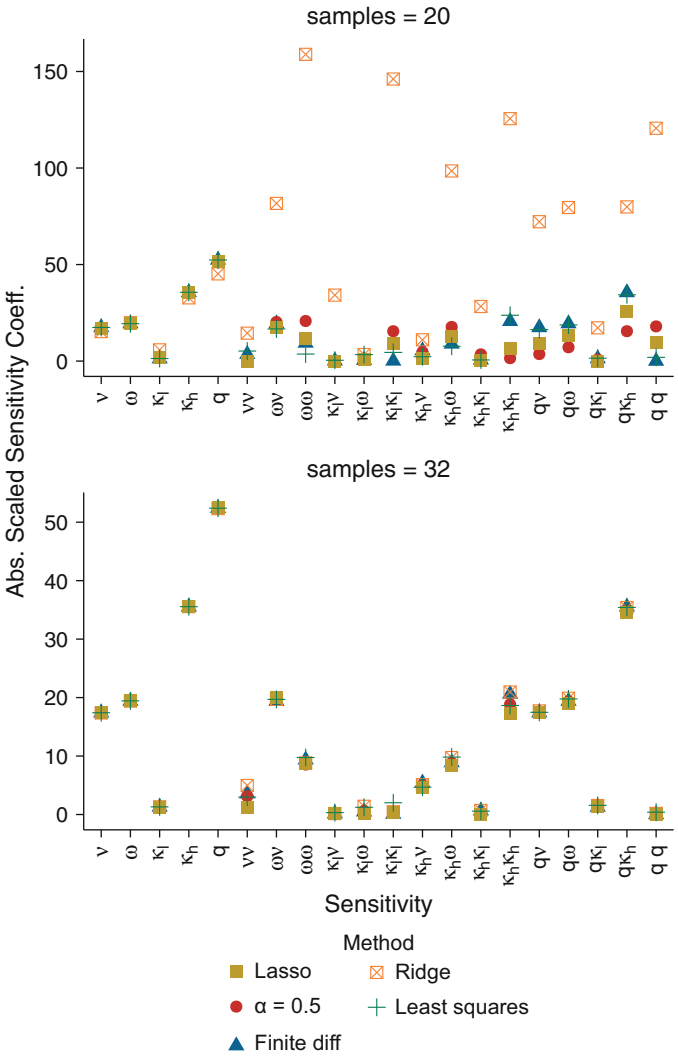

Fig. 5.9 Comparison of the different regression methods to estimate the first- and second-derivative scaled sensitivity coefficients using 20 and 32 evaluations of a QoI with 5 parameters. The finite difference estimates required 51 QoI evaluations

QoI evaluations are sufficient to use least-squares regression. Indeed, with both 20 and 32 samples, the least-squares estimates give estimates that are closest to the finite difference estimate. With 32 evaluations of the QoI, the regularized regression estimates are all accurate. With only 20 evaluations of the QoI, the regularized regression estimates lose some accuracy; ridge regression has large errors in the estimation of the second-derivative sensitivities, however.

5.5 Notes and References

In this chapter we covered the regularized regression techniques to estimate sensitivities when the number of parameters is greater than the number of evaluations of the quantity of interest. There are additional approaches that we did not cover that, in some circumstances, can be effective at the same problem. Some examples are least-angle regression, forward stepwise regression, and principal component regression.

A comprehensive and recent reference for the detailed theory behind the methods covered in this chapter (and those mentioned in the previous paragraph) is the monograph by Hastie et al. (2009).

5.6 Exercises

1. Fit the data in Table 5.1 to a linear model using
   (a) least-squares regression
   (b) ridge regression
   (c) elastic net with $\alpha = 0.5$
   (d) lasso regression

Table 5.1 Data to fit to linear model $y = a + bx_1 + cx_2$

	x_1	x_2	y
1	0.99	0.98	6.42
2	−0.75	−0.76	0.20
3	−0.50	−0.48	0.80
4	−1.08	−1.08	−0.57
5	0.09	0.09	4.75
6	−1.28	−1.27	−1.42
7	−0.79	−0.79	1.07
8	−1.17	−1.17	0.20
9	−0.57	−0.57	1.08
10	−1.62	−1.62	−0.15
11	0.34	0.35	2.90
12	0.51	0.51	3.37
13	−0.91	−0.92	0.05
14	1.85	1.86	5.50
15	−1.12	−1.12	0.17
16	−0.70	−0.70	1.72
17	1.19	1.18	3.97
18	1.24	1.23	6.38
19	−0.52	−0.52	3.29
20	−1.41	−1.44	−1.49

Be sure to do cross-validation for each fit, and for each method present your best estimate of the model.

2. Using a discretization of your choice, solve the equation

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = D\frac{\partial^2 u}{\partial x^2} - \omega u,$$

for $u(x,t)$ on the spatial domain $x \in [0, 10]$ with periodic boundary conditions $u(0^-) = u(10^+)$ and initial conditions

$$u(x,0) = \begin{cases} 1 & x \in [0, 2.5] \\ 0 & \text{otherwise} \end{cases}.$$

Use the solution to compute the total reactions

$$\int_5^6 dx \int_0^5 dt\, \omega u(x,t).$$

Sample values of parameters using a uniform distribution centered at the mean with upper and lower bounds $\pm 10\%$ for the following variables:

(a) $\mu_v = 0.5$,
(b) $\mu_D = 0.125$,
(c) $\mu_\omega = 0.1$,

and sample values of the following parameters in their given ranges:

(a) $\Delta x \sim U[0.001, 0.5]$,
(b) $\Delta t \sim U[0.001, 0.5]$.

Using regression estimate the sensitivities to each parameter.

Chapter 6
Adjoint-Based Local Sensitivity Analysis

I didn't think so much of him at first. But now I get it, he's everything that I'm not.

—from the film *The Royal Tenenbaums*

6.1 Adjoint Equations for Linear, Steady-State Models

Adjoints are useful in local sensitivity analysis because they can give information about any perturbed quantity with only one solve of the forward system (i.e., the system we solve to compute the QoI) and one solve of the adjoint equations. The adjoint problem is defined as a system where the physics, in a sense, happen in reverse. Therefore, adjoint solution allows us to see the effect of perturbations in the QoI parameters. Importantly, solving the forward problem once, and the adjoint problem once, allows us to compute the sensitivity to *all* the parameters. This compares to the $p+1$ solutions needed to compute the sensitivities for p parameters using finite differences.

One important distinction between the adjoint method and finite differences is that each QoI requires a separate adjoint system to be solved, whereas finite differences can be applied to any number of QoIs without additional solutions. On balance, when the number of QoIs is small relative to the number of uncertain parameters, the adjoint approach can be more efficient. The issue with the adjoint approach is that it can be difficult to define the adjoint equations. In this section we will deal with linear, time-independent partial differential equations. In a latter section, we relax this assumption with a concomitant increase in complexity.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-319-99525-0_6) contains supplementary material, which is available to authorized users.

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0_6

6.1.1 Definition of Adjoint Operator

To define an adjoint, we begin by defining an inner product:

$$(f, g) = \int_D dV\, fg, \tag{6.1}$$

where D is the phase space domain of the functions, f and g are functions that are square integrable over the domain D, and dV is a differential phase space element. The adjoint for an operator L is typically denoted L^* and is defined as

$$(Lu, u^*) = (u, L^*u^*), \tag{6.2}$$

using the definition of the inner product above. One can think of the operator L as the part of the differential equations that operates on the dependent variables, as we will see in an example soon. Using the definition of the adjoint in Eq. (6.2), it is easy to show that adjoints make taking inner products of solution variables trivial if the adjoint solution is known.

For a PDE with differential operator L,

$$Lu = q,$$

with an adjoint operator L^* of L, and an adjoint equation

$$L^*u^* = w,$$

then by the above definition:

$$(q, u^*) = (Lu, u^*) = (u, L^*u^*) = (u, w). \tag{6.3}$$

In other words, the inner product of u and w is the same as the inner product of q and u^*.

Now consider a quantity of interest Q given by an integral of the solution u against a weighting function, $w(\mathbf{r})$:

$$Q = \int_D dV\, w(\mathbf{r})u(\mathbf{r}) = (u, w) \tag{6.4}$$

Equation (6.4) indicates that we can define a QoI as an inner product by picking a weighting function. Using relation Eq. (6.3) above, we see that the Q is just (u^*, q). In other words, the adjoint solution with source $w(\mathbf{r})$ integrated against the source q gives the Q, that is, we can calculate the QoI two ways:

$$Q = (u, w) = (u^*, q). \tag{6.5}$$

This is not magical, however, because the adjoint equation is typically as hard to solve as the original PDE as we will see in an example.

We now make the notion of the adjoint concrete for the steady ADR equation with a linear reaction term for $u(x)$ on the domain $(0, X)$ with zero Dirichlet boundary conditions. Under these conditions the ADR equation and boundary conditions are

$$\begin{aligned} v\frac{du}{dx} - \omega\frac{d^2u}{dx^2} + \kappa u &= q \\ u(0) = u(X) &= 0 \end{aligned} \tag{6.6}$$

Using the notation above, we define the operator L as

$$L = v\frac{d}{dx} - \omega\frac{d^2}{dx^2} + \kappa. \tag{6.7}$$

For this domain the inner product given by

$$(u, v) = \int_0^X uv\, dx. \tag{6.8}$$

We will postulate an adjoint form of this system and then show that it satisfies the definition in Eq. (6.2). The form of the adjoint we propose is basically the same equation, with the sign of the advection term flipped:

$$\begin{aligned} L^* &= -v\frac{d}{dx} - \omega\frac{d^2}{dx^2} + \kappa \\ u^*(0) = u^*(X) &= 0. \end{aligned} \tag{6.9}$$

Proof We need to show $(Lu, u^*) = (u, L^*u^*)$ which is equivalent to

$$\int_0^X \left(vu^*\frac{du}{dx} - \omega u^*\frac{d^2u}{dx^2} + \kappa u^* u\right)dx = \int_0^X \left(-vu\frac{du^*}{dx} - \omega u\frac{d^2u^*}{dx^2} + \kappa uu^*\right)dx. \tag{6.10}$$

We will show that these are equivalent term by term. While the κuu^* term is obvious, the advection term needs integration by parts:

$$\int_0^X vu^*\frac{du}{dx}\,dx = \cancel{vu^*u\Big|_0^X} - v\int_0^X u\frac{du^*}{dx}\,dx = \int_0^X -vu\frac{du^*}{dx}\,dx \tag{6.11}$$

which is the term on the RHS of Eq. (6.10). The diffusion term just needs integration by parts twice:

$$\int_0^X u^* \frac{d^2u}{dx^2}\,dx = u^* \cancel{\frac{du}{dx}\Big|_0^X} - \int_0^X \frac{du}{dx}\frac{du^*}{dx}\,dx = \cancel{\frac{du^*}{dx}u\Big|_0^X} + \int_0^X u\frac{d^2u^*}{dx^2}\,dx \tag{6.12}$$

which matches the diffusion term on the RHS of Eq. (6.10). □

With the known form of the adjoint ADR equation, we can use it to compute a QoI. Notice that because the adjoint equation is also an ADR equation, it is no easier to solve than the original equation.

As an example, if our QoI is the average of u over the middle third of the domain, this would make $w(x)$:

$$w(x) = \begin{cases} \frac{3}{X} & x \in [\frac{X}{3}, \frac{2}{3}X] \\ 0 & \text{otherwise} \end{cases}, \tag{6.13}$$

which leads to a Q:

$$Q = \int_{\frac{X}{3}}^{\frac{2}{3}X} \frac{3}{X} u(x)\,dx. \tag{6.14}$$

To get our QoI we could solve

$$Lu = q \quad \text{or} \quad L^*u^* = w, \tag{6.15}$$

and compute

$$Q = (u, w) = (q, u^*). \tag{6.16}$$

The choice of which equation to solve is seemingly immaterial: each involves solving an ADR-like equation and then computing the inner product. There are instances where having an estimate of the adjoint solution can make the forward problem easier to solve, a salient example being source-detector problems in Monte Carlo particle transport simulations (Wagner and Haghighat 1998). The reason that this works is that the adjoint solution is, in a sense, a measure of how important a region of space is to the QoI.

6.1.2 *Adjoints for Computing Derivatives*

Our interest in the adjoint solution arises from the manner in which they allow first-order sensitivities to be computed. In some situations, this is called perturbation

analysis, but as we will see it is the same as the sensitivity analyses discussed above. Consider the perturbed problem:

$$(L + \delta L)(u + \delta u) = q + \delta q \tag{6.17}$$

where δL and δq are perturbations to the original problem and δu is the change in the solution due to changing the problem. In the ADR example, the δL would involve changing the advection speed, diffusion coefficient, or reaction operator, and δq would be a change to the source.

Expanding the product on the LHS of Eq. (6.17) we get

$$Lu + L\delta u + \delta L u = q + \delta q + \mathrm{O}(\delta^2). \tag{6.18}$$

Henceforth, we will ignore second-order perturbations, i.e., the δ^2 terms. Now, $Lu = q$ so those terms can be cancelled to give

$$L\delta u + \delta L u = \delta q. \tag{6.19}$$

Upon multiplying by u^* and taking the inner product, this becomes

$$(L\delta u, u^*) + (\delta L u, u^*) = (\delta q, u^*). \tag{6.20}$$

This equation is useful, except we do not know what δu is. It is simple to compute the perturbation to L and apply it to a known forward solution u (this just involves taking derivatives). Similarly, we can compute δq easily because q is a parameter.

To remove the δu from Eq. (6.20) we will use the property of the adjoint that we can "switch" L and L^* in the inner product to make the relation

$$(L\delta u, u^*) = (\delta u, L^* u^*) = (\delta u, w), \tag{6.21}$$

where $L^* u^* = w$ was used in the second equality. This makes Eq. (6.20)

$$(\delta u, w) + (\delta L u, u^*) = (\delta q, u^*). \tag{6.22}$$

Therefore, if we can get another relation for $(\delta u, w)$, then we can eliminate δu from our equations.

The definition of the perturbed QoI is

$$Q + \delta(Q) = \int_D dV\, wu + \int_D dV\, w\delta u + \int_D dV\, (\delta w)u + O(\delta^2). \tag{6.23}$$

Here we have allowed for the case where w may be dependent on a parameter by including the δw. In the ADR equation this could be the case if, for example, the QoI were the reaction rate in a particular region. Then w would depend on the reaction coefficient.

We can rearrange Eq. (6.23) to get

$$(\delta u, w) = \delta(Q) - (u, \delta w).$$

Using this result in Eq. (6.22) gives an equation for the perturbation to the QoI in terms of perturbations to parameters and the forward and adjoint solution:

$$\delta(Q) = (\delta q, u^*) + (u, \delta w) - (\delta L u, u^*) \tag{6.24}$$

That is, if we know u and u^*, we can compute $\delta(Q)$. In general, for a quantity θ, we interpret the quotient $\delta Q/\delta\theta$ as the partial derivative of the QoI with respect to θ. This interpretation is reasonable because the perturbation can be as small as we like. Therefore we write

$$\frac{\partial Q}{\partial \theta} = \left(\frac{\partial q}{\partial \theta}, u^*\right) + \left(u, \frac{\partial w}{\partial \theta}\right) - \left(\frac{\partial L}{\partial \theta} u, u^*\right). \tag{6.25}$$

This derivative formula gives us a way to compute sensitivity coefficients without taking finite difference derivatives. Also, for each parameter θ we can use the same u^* and u to compute the sensitivity by changing what goes into the inner product.

6.1.2.1 ADR Example: Computing Derivatives from Each Parameter

Using the same data as the example in the previous chapter (Sect. 4.3) where the source and κ varied over space and the QoI was the total reaction rate,

$$\begin{aligned} Q = (u, \kappa) &= \int_0^{10} \kappa(x) u(x)\, dx \\ &= \int_0^5 \kappa_h u(x)\, dx + \int_5^{7.5} \kappa_l u(x)\, dx + \int_{7.5}^{10} \kappa_h u(x)\, dx, \end{aligned}$$

we can compute the sensitivities using the adjoint. For convenience we will write the source in the example as $q(x) = \hat{q}x(10 - x)$ so that there is no confusion between the general source in our adjoint derivations, q, and the source strength in the ADR example, now set to $\hat{q}$.

We can use the code from that example to implement the adjoint operator by simply running the code with $v \rightarrow -v$ and setting the source in the adjoint equation to be κ. The adjoint solution u^* at the mean of the all the parameters is shown in Fig. 6.1. In this case if we compute the QoI using the forward or adjoint solution using Eq. (6.5) we get a match to 12 digits for our QoI, the total reaction rate:

$$(u, \kappa) = 52.3903954692 \qquad (S, u^*) = 52.3903954692.$$

The numerical method to solve the adjoint ADR equation is given in Algorithm 6.1. Notice that this algorithm differs from the forward model only by the sign

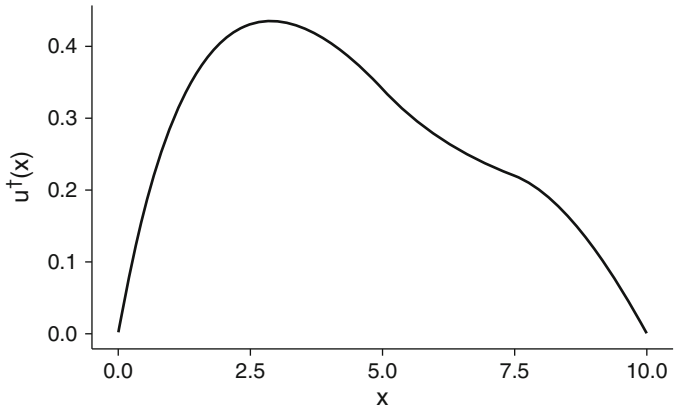

Fig. 6.1 The solution $u^*(x)$ evaluated at $\bar{\theta}$

Algorithm 6.1 Numerical method in `python` to solve the adjoint advection-diffusion-reaction equation that will be used in this chapter

```
import numpy as np
import scipy.sparse as sparse
import scipy.sparse.linalg as linalg
def ADRSource(Lx, Nx, Source, omega, v_in, kappa):
    A = sparse.dia_matrix((Nx,Nx),dtype="complex")
    dx = Lx/Nx
    v = -1*v_in
    i2dx2 = 1.0/(dx*dx)
    #fill diagonal of A
    A.setdiag(2*i2dx2*omega + np.sign(v)*v/dx + kappa)
    #fill off diagonals of A
    A.setdiag(-i2dx2*omega[1:Nx] +
              0.5*(1-np.sign(v[1:Nx]))*v[1:Nx]/dx,1)
    A.setdiag(-i2dx2*omega[0:(Nx-1)] -
              0.5*(np.sign(v[0:(Nx-1)])+1)*v[0:(Nx-1)]/dx,-1)
    #solve A x = kappa
    Solution = linalg.spsolve(A,Source)
    Q = np.sum(Solution*Source*dx)
    return Solution, Q
```

of v, the righthand side of the resulting linear system is now κ, and the calculation of Q now uses the source.

To compute the inner products we use simple quadrature based on the midpoint rule. The derivative of Q with respect to v is computed using Eq. (6.25) to get

$$\frac{\partial Q}{\partial v} = -\left(\frac{\partial u}{\partial x}, u^*\right) = -1.74049052049,$$

where we used the fact that q and w were independent of v and

$$\frac{\partial L}{\partial v} = \frac{\partial}{\partial v}\left(v\frac{d}{dx} - \omega\frac{d^2}{dx^2} + \kappa\right) = \frac{d}{dx}.$$

The derivative of u must be estimated from the forward solution using finite differences. Here we used forward finite differences because this is consistent with what we used in the solution technique. This number agrees with the finite difference result given previously to four digits.

Similarly, the derivative with respect to ω involves the integral of the second derivative of the forward solution times the adjoint solution:

$$\frac{\partial Q}{\partial \omega} = \left(\frac{\partial^2 u}{\partial x^2}, u^*\right) = -0.970207772262.$$

For κ_l the derivative is based on an integral only over the range $x \in (5, 7.5)$ because the operator only depends on κ_l in that range. Additionally, however, w also depends on κ_l meaning that we will have two terms in our uncertainty:

$$\frac{\partial Q}{\partial \kappa_l} = \int_5^{7.5} u(x)\,dx - \int_5^{7.5} u(x)u^*(x)\,dx = 12.862742303.$$

The first term here is the derivative with respect to w term in Eq. (6.25), and the second term is the derivative of L term.

The sensitivity to κ_h is an integral over the parts of the problem where $\kappa = \kappa_h$, again with terms relating to the derivatives of w and L:

$$\begin{aligned}\frac{\partial Q}{\partial \kappa_h} &= \int_0^5 u(x)\,dx + \int_{7.5}^{10} u(x)\,dx - \int_0^5 u(x)u^*(x)\,dx - \int_{7.5}^{10} u(x)u^*(x)\,dx \\ &= 17.7613932101.\end{aligned}$$

The final sensitivity to compute is involves the source strength, q. From Eq. (6.25) we get

$$\frac{\partial Q}{\partial \hat{q}} = (x(10-x), u^*) = 52.3903954692.$$

Notice that with the $\hat{q}$ sensitivity we only have the derivative with respect to q contributing to the sensitivity in this case.

These results all agree with the first-order derivative results in Table 4.1 to several digits.

6.1.2.2 ADR Example: Random Process for κ

Previously, we considered the situation in Sect. 4.3.2 where κ was defined by a random process so that each value of κ in the system was a random variable. When

using finite differences, we required 2001 solutions to the ADR equations in this case to get the sensitivity of Q with respect to κ when we used 2000 mesh cells. With the adjoint approach, we only need a single forward and a single adjoint solve to arrive at the same answer. To compute the sensitivity of Q to the value of κ in any mesh cell i, we evaluate

$$\begin{aligned}\frac{\partial Q}{\partial \kappa_i} &= \int_{x_{i-1/2}}^{x_{i+1/2}} u(x)\,dx - \int_{x_{i-1/2}}^{x_{i+1/2}} u(x)u^*(x)\,dx \\ &= \int_{x_{i-1/2}}^{x_{i+1/2}} u(x)(1-u^*)\,dx \approx u_i(1-u_i^*)\Delta x,\end{aligned}$$

where $x_{i+1/2}$ is the right edge of the ith mesh cell, u_i is the average value of $u(x)$ in mesh cell i, and Δx is the width of the mesh cell.

This calculation gives identical results to those in Sect. 4.3.2 with a factor 1000 fewer solutions to the ADR equation (2 compared to 2001).

6.2 Adjoints for Nonlinear, Time-Dependent Equations

In the previous section, we had to make some strong assumptions about the underlying mathematical model to use adjoints, namely, that it was steady in time and linear. In this section we relax that assumption and show how an adjoint equation can be formed. To begin we have a time-dependent PDE of the form

$$F(u(x,t), \dot{u}(x,t)) = 0, \qquad x \in V,\ t \in [0, t_\mathrm{f}], \tag{6.26}$$

where $F(u, \dot{u})$ is an operator, $\dot{u}$ is the time derivative of u, V is the spatial domain, and the time domain goes from time 0 to t_f. We also have boundary conditions such that the solution goes to zero on the boundary of the domain of interest:

$$u(x,t) = 0 \qquad x \in \partial V,$$

where ∂V denotes the boundary. The choice of homogenous boundary conditions is made here primarily for convenience and could be relaxed.

As before, we define an inner product (u, v) as the integral of uv over phase space. It will be useful to write the QoI as an integral over phase-space and time separately. In particular

$$Q = \int_{t_0}^{t_\mathrm{f}} (u, w)\,dt. \tag{6.27}$$

We can modify this equation for the QoI by adding a Lagrange multiplier, u^*, times F without changing the QoI because $F(u, \dot{u}) = 0$. We call this new quantity the adjoined metric and write it as

$$\mathscr{L} = \int_{t_0}^{t_\mathrm{f}} \left[(u, w) - (F, u^*)\right] dt. \tag{6.28}$$

The first-order sensitivity (i.e., first variation) to a parameter, θ, for a functional $L(u, \dot{u})$ is defined as

$$\frac{dL}{d\theta} = \frac{\partial L}{\partial u}\frac{\partial u}{\partial \theta} + \frac{\partial L}{\partial \dot{u}}\frac{\partial \dot{u}}{\partial \theta} + \frac{\partial L}{\partial \theta},$$

where the partial derivatives hold the other quantities constant, e.g., $\partial L/\partial\theta$ is taken with u and $\dot{u}$ constant. One can think of the first variation as the total derivative of the functional with respect to a parameter.

Using this definition of the first variation, we can then write for $L_Q \equiv (u, w)$,

$$\frac{dL_Q}{d\theta} = \left(\frac{\partial u}{\partial u}, w\right)\frac{\partial u}{\partial \theta} + \left(u, \frac{\partial w}{\partial \theta}\right) = (1, w)u_\theta + (u, w_\theta), \tag{6.29}$$

where subscripts indicate partial derivatives. Also, we can write

$$\frac{d}{d\theta}(F, u^*) = \left(\frac{\partial F}{\partial u}, u^*\right)u_\theta + \left(\frac{\partial F}{\partial \dot{u}}, u^*\right)\dot{u}_\theta + \frac{\partial}{\partial \theta}\left(F, u^*\right). \tag{6.30}$$

Using these relations we get the first-order sensitivity as

$$\frac{d\mathscr{L}}{d\theta} = \int_{t_0}^{t_\mathrm{f}} \left[(1, w)u_\theta + (u, w_\theta) - \frac{\partial}{\partial \dot{u}}(F, u^*)\dot{u}_\theta - \frac{\partial}{\partial u}(F, u^*)u_\theta - \frac{\partial}{\partial \theta}(F, u^*)\right] dt. \tag{6.31}$$

As before we would like to eliminate the u_θ and $\dot{u}_\theta$ terms because we do not know the derivative of the solution or its time derivative with respect to the parameter. To make this elimination, we first integrate the $\dot{u}_\theta$ term by parts to get

$$\begin{aligned}\frac{d\mathscr{L}}{d\theta} = &- \frac{\partial}{\partial \dot{u}}(F, u^*)u_\theta\Big|_{t_0}^{t_\mathrm{f}} \\ &+ \int_{t_0}^{t_\mathrm{f}} \left[(1, w)u_\theta + (u, w_\theta) + u_\theta \frac{d}{dt}\frac{\partial}{\partial \dot{u}}(F, u^*)\right. \\ &\left. - u_\theta \frac{\partial}{\partial u}(F, u^*) - \frac{\partial}{\partial \theta}(F, u^*)\right] dt.\end{aligned} \tag{6.32}$$

In order to eliminate u_θ from this equation, except for the boundary term, we will define the Lagrange multiplier so that

$$(1, w) + \frac{d}{dt}\frac{\partial}{\partial \dot{u}}(F, u^*) - \frac{\partial}{\partial u}(F, u^*) = 0. \tag{6.33}$$

The boundary term in Eq. (6.32) evaluates u_θ at $t = t_0$ and t_f. At the final time, we can state that $u^*(t_\mathrm{f}) = 0$ to represent the fact that anything that happens beyond the final time does not contribute to the quantity of interest. The issue of the final

condition on u^* indicates a subtlety of the adjoint equation: it runs backward in time. One has to solve the adjoint equation starting at the final time and solve backward to get to t_0. At $t = t_0$ this term indicates how the initial conditions for u are perturbed by the parameter. Therefore, we only need to consider this quantity if the initial conditions are dependent on θ.

Given that Eq. (6.33) gives a relationship between integrals, it will also be true if the relation holds at each point in space. Therefore, we can write the stronger statement:

$$-\frac{d}{dt}\frac{\partial}{\partial \dot{u}}Fu^* = -\frac{\partial}{\partial u}Fu^* + w. \tag{6.34}$$

However, the integral form in Eq. (6.33) will be needed to form boundary conditions.

Upon solving Eq. (6.33) we can compute the sensitivity of the QoI to parameter θ through the equation:

$$\frac{d\mathscr{L}}{d\theta} = -\frac{\partial}{\partial \dot{u}}(F, u^*)u_\theta\Big|_{t_0} + \int_{t_0}^{t_f}\left[(u, w_\theta) - \frac{\partial}{\partial \theta}(F, u^*)\right]dt. \tag{6.35}$$

Notice that to evaluate the equation we need the full forward solution and adjoint solution for all time to compute the integrals. This could represent a storage problem for large-scale systems.

6.2.1 Linear ADR Equation

The above derivation of the adjoint for a time-dependent problem was fairly abstract. We shall show it applies to the ADR problem we have seen before. For the linear ADR equation we saw before, the system can be written in as $F(u, \dot{u}) = 0$ where

$$F(u, \dot{u}) = \dot{u} + v\frac{\partial u}{\partial x} - \omega\frac{\partial^2 u}{\partial x^2} + \kappa u - S.$$

We also consider a problem domain given by $x \in [0, X]$, with $u(0, t) = u(X, t) = 0$. For a generic QoI weighting function w, terms in Eq. (6.33) are computed below. We begin with the term involving the derivative with respect to $\hat{u}$:

$$\begin{aligned}\frac{d}{dt}\frac{\partial}{\partial \dot{u}}(F, u^*) &= \frac{d}{dt}\frac{\partial}{\partial \dot{u}}\int_0^X u^*(x, t)\left(\dot{u} + v\frac{\partial u}{\partial x} - \omega\frac{\partial^2 u}{\partial x^2} + \kappa u - S\right)dx \\ &= \int_0^X \frac{\partial u^*}{\partial t}\,dx.\end{aligned}$$

In this equation the derivative is simple to compute because $\dot{u}$ only appears in a single term. The other term in the definition of the adjoint from Eq. (6.33) will require integration by parts. This term is

$$\begin{aligned}\frac{\partial}{\partial u}(F,u^*) &= \frac{\partial}{\partial u}\int_0^X u^*(x,t)\left(\dot{u}+v\frac{\partial u}{\partial x}-\omega\frac{\partial^2 u}{\partial x^2}+\kappa u-S\right)dx\\ &= \frac{\partial}{\partial u}\int_0^X u(x,t)\left(-v\frac{\partial u^*}{\partial x}-\omega\frac{\partial^2 u^*}{\partial x^2}+\kappa u^*\right)dx\\ &= \int_0^X\left(-v\frac{\partial u^*}{\partial x}-\omega\frac{\partial^2 u^*}{\partial x^2}+\kappa u^*\right)dx.\end{aligned}$$

where we used integration by parts to move the derivatives onto the adjoint variables. In doing so we relied on the fact that $u(0,t) = u(X,t) = 0$ and that we are free to define the boundary conditions for u^* to be $u^*(0,t) = u^*(X,t) = 0$.

If we assert that Eq. (6.33) holds at every point in the medium, then we have the adjoint equation

$$-\frac{\partial u^*}{\partial t}-v\frac{\partial u^*}{\partial x}-\omega\frac{\partial^2 u^*}{\partial x^2}+\kappa u^* = w,$$

with boundary and final conditions,

$$u^*(0,t) = u^*(X,t) = 0, \qquad u^*(x,t_\mathrm{f}) = 0.$$

This equation is the time-dependent version of Eq. (6.1.1). As a result our new approach to deriving an adjoint is equivalent to the previous one in the steady-state limit; except now we can, in principle, handle more complicated equations.

6.2.2 Nonlinear Diffusion-Reaction Equation

As an example of a more complicated PDE that we can derive an adjoint for, we will look at a nonlinear diffusion-reaction equation inspired by a common model of radiative transfer in the high-energy density regime (Humbird and McClarren 2017). In this case we redefine $F(u,\dot{u})$ as

$$F(u,\dot{u}) = \rho\dot{u}-\omega\frac{\partial^2 u^4}{\partial x^2}+\kappa u^4-S. \tag{6.36}$$

The boundary conditions we use are $u(0,t) = u(X,t) = 0$ and the initial condition is 0. Notice that for this new form we have the time derivative linear in u and the other terms involve u^4. For this new form of F we will have to compute

$$\frac{\partial}{\partial u}\int_0^X u^*(x,t)\left(-\omega\frac{\partial^2 u^4}{\partial x^2}+\kappa u^4-S\right)dx = \int_0^X\left(-4\omega u^3\frac{\partial^2 u^*}{\partial x^2}+4\kappa u^3 u^*\right)dx.$$

As a result, the adjoint equation is

$$-\rho\frac{\partial u^*}{\partial t} - 4\omega u^3\frac{\partial^2 u^*}{\partial x^2} + 4\kappa u^3 u^* = w.$$

This is a linear equation, but it requires knowledge of $u(x,t)$ at every time and point in space to evaluate.

To demonstrate how this works we will solve a problem where, $X = t_\mathrm{f} = 2$,

$$\kappa(x) = \begin{cases} \kappa_h & 1 \le x \le 1.5 \\ \kappa_l & \text{otherwise} \end{cases},$$

$$S(x) = \begin{cases} q & 0.5 \le x \le 1.5 \\ 0 & \text{otherwise} \end{cases}.$$

In the problem the nominal values will be

$$\rho = 1, \quad \omega = 0.1, \quad \kappa_l = 0.1, \quad \kappa_h = 2, \quad q = 1.$$

The quantity of interest will be given by

$$Q = \int_{1.8}^{2} dt \int_{1.5}^{1.9} dx\, \kappa(x) u(x,t),$$

this makes

$$w(x,t) = \begin{cases} \kappa(x) & x \in [1.5, 1.9], t \in [1.8, 2] \\ 0 & \text{otherwise} \end{cases}.$$

Notice that we only need the adjoint solution in the time range $t \in [1.8, 2]$. The solution to the forward and adjoint versions of this problem are shown in Fig. 6.2.

With this system we will find the sensitivities to ρ, ω, κ_l, and κ_h, q using finite differences and the adjoint methodology. The results are given in Table 6.1. These results were obtained using 200 mesh cells of finite difference and a second-order predictor-corrector Runge-Kutta method with $\Delta t = 0.0001$. The finite difference parameter used was $\delta = 10^{-6}$.

In these results we see that the finite difference and adjoint estimates of the sensitivities agree to four significant digits, numerically demonstrating that the adjoint approach can be extended to nonlinear and time-dependent problems.

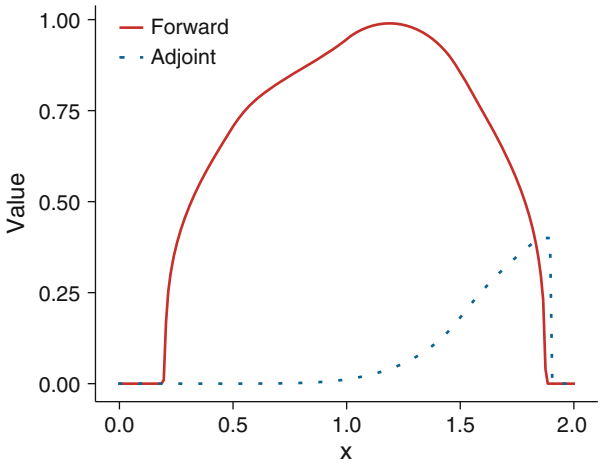

Fig. 6.2 The solution $u(x, 2)$ and $u^*(x, 1.8)$ to the nonlinear diffusion-reaction problem

Table 6.1 First-order sensitivity results for the nonlinear diffusion-reaction problem

	Finite difference	Adjoint estimate	Abs. Rel. difference (10^{-5})
ρ	−0.099480	−0.099484	4.584074
ω	0.288975	0.288994	6.309322
κ_l	−0.030224	−0.030226	6.013714
κ_h	0.032156	0.032158	5.221466
q	0.096382	0.096387	5.469452

6.3 Notes and Further Reading

In this chapter we have introduced the notion of adjoint methods to estimate sensitivities to quantities of interest to parameter perturbation. We have only considered first-order perturbations in our discussion. It is possible to derive adjoint equations to estimate higher-order sensitivities. See Wang et al. (1992), Cacuci (2015) for a discussion of these methods.

We did not discuss the issue of compatibility between the numerical methods used for the forward and adjoint solutions. We derived adjoint equations for the continuous operators. When solving the equations, the discretizations used may not preserve the properties of adjoints. In our examples, this happened to be case, but will not always be true. A discussion of this phenomenon can be found in Wilcox et al. (2015).

Finally, the generation of adjoint systems can be automated. A recent attempt at automating the construction and solution of adjoint equations can be found in Farrell et al. (2013).

6.4 Exercises

1. Derive the adjoint operator for the equation

$$-\nabla^2\phi(x, y, z) + \frac{1}{L^2}\phi(x, y, z) = \frac{Q}{D},$$

$$\phi(0, y, z) = \phi(x, 0, z) = \phi(x, y, 0) = \phi(X, y, z)$$
$$= \phi(x, Y, z) = \phi(x, y, Z) = 0.$$

Compute the sensitivity to the QoI:

$$\text{QoI} = \int_0^X dx \int_0^Y dy \int_0^Z dz \, \frac{D}{L^2}\phi(x, y, z),$$

for X, Y, Z, L, D, and Q.

2. Using a discretization of your choice, solve the equation

$$v\frac{\partial u}{\partial x} = D\frac{\partial^2 u}{\partial x^2} - \omega u + 1,$$

for $u(x)$ on the spatial domain $x \in [0, 10]$ with periodic boundary conditions $u(0) = u(10)$. Use the solution to compute the total reactions

$$\int_5^6 dx \, \omega u(x).$$

Derive the adjoint equation for this equation, and use its numerical solution to compute the sensitivities to the following parameters.

(a) $\mu_v = 0.5$,
(b) $\mu_D = 0.125$,
(c) $\mu_\omega = 0.1$,

Part III
Uncertainty Propagation

In this part we will apply various techniques to understand the distribution of quantities of interest due to the distribution of input parameters. These methods range from straightforward, robust, and expensive sampling-based techniques to inexpensive, but fragile, reliability methods. The content of this part is not independent from the local sensitivity work in the previous part. Sensitivity analysis can be used to screen out those inputs that have a small impact on the QoI. This can save considerable time when propagating uncertainties. At the same time, there is the danger that an unimportant variable at one point of input space may be important in another.

In the next chapter, we present methods based on random samples from the input distributions to get samples of the QoI and then use those samples to infer properties of the QoI distribution.

Chapter 7
Sampling-Based Uncertainty Quantification: Monte Carlo and Beyond

What were Stephen's and Bloom's quasisimultaneous volitional quasisensations of concealed identities?

—James Joyce, *Ulysses*

7.1 Basic Monte Carlo Methods: Simple Random Sampling

In its most basic form, we will use Monte Carlo methods to produce samples from distributions and use those samples to infer information regarding the distribution. Oftentimes, we are interested in the distribution of a QoI, but the methods we cover are not restricted to these cases.

Assume that we have N independent and identically distributed samples from the probability distribution of a QoI, $Q(\mathbf{X})$, where $\mathbf{X}$ is a p-dimensional vector of random variables. These samples are obtained by sampling values of $\mathbf{X}$ and evaluating $q(\mathbf{x})$. We are interested to know the expectation of some function of the QoI. Using our previous definitions, we have

$$E[g(Q)] = \int dx_1 \cdots \int dx_p \, g(q(\mathbf{x})) f(\mathbf{x}), \tag{7.1}$$

where $f(\mathbf{x})$ is the probability density function for the inputs to the QoI. The standard Monte Carlo estimator defines an estimate of the integral as

$$I_N \equiv \frac{1}{N} \sum_{n=1}^{N} g(q(\mathbf{x}_n)). \tag{7.2}$$

The value of I_N will limit to $E[g(Q)]$ as $N \to \infty$ by the law of large numbers. This result will hold even if the variance in any component of $\mathbf{X}$ is unbounded.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-319-99525-0_7) contains supplementary material, which is available to authorized users.

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0_7

This result tells us that if we estimate moments of the QoI using samples, we can get good estimates given "enough" samples. A good question to ask is, what is enough?

We can estimate the variance in I_N using the variance in the samples of $\mathbf{X}$. Consider the sample variance of Q:

$$\sigma_N^2 = \frac{1}{N-1} \sum_{n=1}^{N} \left(\bar{Q}_N - Q(\mathbf{x}_n) \right)^2,$$

where $\bar{Q}_N$ is the sample mean of the N samples. If the true variance of the QoI is finite, then by the central limit theorem, the error in the estimate, $I_N - E[g(Q)]$, will converge, as $N \to \infty$, to a normal distribution with mean zero and variance given by

$$\text{Var}\left(I_N - E[g(X)]\right) = \frac{\sigma_N^2}{N}.$$

In other words, the error in the Monte Carlo estimator, as represented by the standard deviation of the estimator, goes to zero as the square root of the number of samples, with a constant that depends on the sample variance of the QoI.

A classical example of the Monte Carlo estimator considers the QoI given by

$$Q(x, y) = \begin{cases} 4 & x^2 + y^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}, \tag{7.3}$$

with the joint PDF of X and Y given by $f(x, y)$

$$f(x, y) = \begin{cases} \frac{1}{4} & (x, y) \in [-1, 1] \times [-1, 1] \\ 0 & \text{otherwise} \end{cases}.$$

We are interested in the mean of Q:

$$\bar{Q} = \frac{1}{4} \int_{-1}^{1} dx \int_{-1}^{1} dy \, q(x, y) = \pi. \tag{7.4}$$

The result of this integral is π because we are taking the area of a circle with radius 1.

To use a Monte Carlo estimate for this problem, we sample N values of x and y based on the joint distribution (in this case we can simply sample x and y from a uniform distribution between -1 and 1). We then evaluate Eq. (7.3) and average the results over all N.

In Fig. 7.1 the error in the estimate of $\bar{Q}$ is shown for different values of N. For each value of N, a single estimate is computed. Therefore, due to the randomness

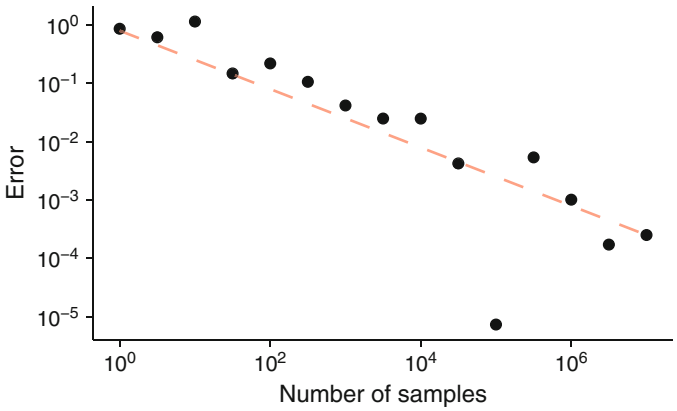

Fig. 7.1 The convergence of the error for the Monte Carlo estimate of the mean given by Eq. (7.4) at different numbers of samples, N. For each value of N, a single estimate was computed. The dashed line has a slope of $-1/2$

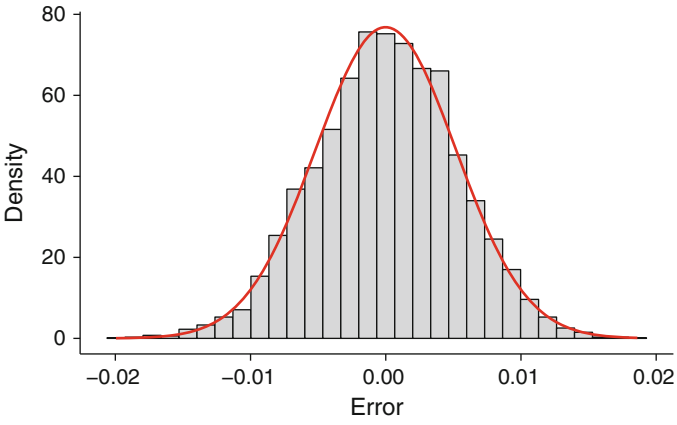

Fig. 7.2 The distribution of 5000 estimates of the mean using $N = 10^5$ given in Eq. (7.4). The bars are the histogram of the estimates, and the curve is the normal distribution with mean zero and standard deviation given by $\sigma_N/\sqrt{N}$

of the sampling, there is some "noise" in the convergence. The dashed line in this figure has a slope of $-1/2$, demonstrating that the error decreases proportional to $N^{-1/2}$.

We show the empirical distribution of estimates with $N = 10^5$ in Fig. 7.2. In this figure we show a histogram of 5000 estimates of $\bar{Q}$. Additionally, for each estimate we computed the sample variance. In the figure we include the PDF for a normal distribution with mean zero and a variance given by the mean of the sample variances divided by 10^5. This PDF agrees with the histogram, as predicted by the theory above.

7.1.1 Empirical Distributions

The estimation of moments of the distribution is an important use of Monte Carlo, but there are other quantities that we may be interested in that are not directly related to moments. For instance, we may be interested in the probability that the quantity of interest is above a certain limit. In this case we could take the fraction of the samples above that limit and quote that as the estimate. Care must be taken in this exercise because, if the true probability of exceeding the limit is 10^{-6}, then one would not expect to get a sample beyond the limit without taking 10^6 (or likely more) samples.

Additionally, one can produce an empirical CDF based on the samples. The empirical CDF for a random variable with N samples is computed as

$$F_N(t) = \frac{\#\text{ of samples less than } t}{N}. \tag{7.5}$$

The median can be estimated by computing the sample median. This statistic is very robust to the behavior extreme values in the tails of the distribution. This compares with the mean, which can be influenced by a single extreme value.

7.1.2 Maximum Likelihood Estimation

Consider a random variable that we believe to be described by a particular family of probability distributions, for example, a normal, gamma, or some other common distribution. This distribution will have some set of parameters to describe it; denote these as θ. Using Bayes' rule, the posterior distribution for these parameters, $\pi(\theta)$, conditional on drawing N samples of a random variable x, is

$$\pi(\theta|x_1, \ldots, x_N) \propto \prod_{n=1}^{N} f(x_n|\theta)\pi(\theta). \tag{7.6}$$

The maximum likelihood estimate assumes that the prior distribution on θ is uniform and then finds the values of θ that maximize the likelihood function, $\prod_{n=1}^{N} f(x_n|\theta)$.

To demonstrate this procedure, we will consider N samples from a distribution that we assume to be normal. We would like to use these samples to estimate the mean and variance of the underlying distribution. In this case the likelihood function is

$$\begin{aligned}\prod_{n=1}^{N} f(x_n|\theta) &= \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(\mu - x_n)^2}{2\sigma^2}\right] \\ &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \exp\left[-\sum_{n=1}^{N} \frac{(\mu - x_n)^2}{2\sigma^2}\right].\end{aligned}$$

To maximize this function, we take its derivative and set it to zero. Before taking the derivative, we use the trick of taking the logarithm first. We can do this because the likelihood is nonnegative:

$$\log \prod_{n=1}^{N} f(x_n|\theta) = -\frac{N}{2}\log\sigma^2 - \frac{N}{2}\log 2\pi - \sum_{n=1}^{N}\frac{(\mu - x_n)^2}{2\sigma^2}.$$

The derivative of this with respect to μ is

$$\frac{d}{d\mu}\log \prod_{n=1}^{N} f(x_n|\theta) = -\sum_{n=1}^{N}\frac{(\mu - x_n)}{\sigma^2} \tag{7.7}$$

The root of this derivative can be solved for μ as

$$\mu = \frac{1}{N}\sum_{n=1}^{N} x_n.$$

This is the standard estimate for the mean that we have used before.

For σ^2 we get the derivative

$$\frac{d}{d\sigma^2}\log \prod_{n=1}^{N} f(x_n|\theta) = -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{n=1}^{N}(\mu - x_n)^2. \tag{7.8}$$

Setting the derivative to 0 and solving for σ^2, we get the maximum likelihood estimate

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(\mu - x_n)^2,$$

where μ is computed as above. This is the estimate of the variance we have seen before, except it does not have Bessel's correction.

What this example indicates is that we can use the maximum likelihood estimator to estimate properties of a distribution. To use this we had to assume a form for the final distribution (a normal in the example). In the case of the normal distribution, we could do this by hand. In many cases, we may require numerical root finding to determine the value of θ. As we mentioned in Chap. 1, this selection of a distribution that we desire to fit to our data could be a source of epistemic uncertainty. The question of how our choice of distribution affects our conclusions needs consideration but is not always easy to quantify. For instance, for any set of samples, we could fit a normal distribution using the procedure detailed in the example. However, the resulting distribution may not match the histogram of the

data at all. At the very least, one should compare the behavior of the fit distribution to the samples collected.

7.1.3 Method of Moments

The method of moments is an alternative to maximum likelihood when one wants to find the parameters of a prescribed distribution from samples. In this case, we assume that there are K parameters, i.e., θ is a vector of length K. Using N samples from the random variable X, we then compute estimates of K moments from the samples

$$E_N[X^k] = \frac{1}{N}\sum_{n=1}^{N} x_n^k, \qquad k = 1, \ldots, K.$$

Then we equate these K moments to the same moments of the distribution we want to fit:

$$\begin{aligned} E_N[X] &= \int x f(x|\theta)\, dx, \\ &\vdots \\ E_N[X^K] &= \int x^K f(x|\theta)\, dx. \end{aligned} \tag{7.9}$$

The right-hand side of the system in (7.9) can be computed exactly for many common distributions and will be a function of θ only. Therefore, in principle we have a soluble system because we have K equations and K unknowns. The values of θ found then give us a distribution that matches the moments of our samples. The method of moments distribution will, in general, differ from a distribution fit using maximum likelihood. For many distributions, the method of moments can be easier to compute than the maximum likelihood estimate.

We will apply the method of moments to the example of considering N samples from a distribution we believe to be normal. In this case we have two parameters to fit, $\theta = (\mu, \sigma^2)$. Assume that we have computed, from our sample, the mean, which we will denote as μ_s here to make it clear that it is the sample mean, and second moment, $E[X^2]$. Performing the integrations in (7.9), we get the system

$$\begin{aligned} \mu_s &= \mu, \\ E_N[X^2] &= \mu^2 + \sigma^2. \end{aligned}$$

The first of these says that the estimate of the mean, μ, is the sample mean. The second gives

$$\sigma^2 = E[X^2] - \mu^2 = \frac{1}{N}\sum_{n=1}^{N}(\mu_s^2 - x_n^2).$$

These estimates are equivalent to the values from the maximum likelihood estimate. This will not always be the case, but it is a special property of the normal distribution.

To further demonstrate the method, we consider fitting a Gumbel distribution to some samples. The Gumbel distribution has support on the real line and has a PDF of

$$f(x|m,\beta) = \frac{1}{\beta}e^{-(z+e^{-z})}, \qquad \text{where} \quad z = \frac{x-m}{\beta}. \tag{7.10}$$

The two parameters of the distribution are $\theta = (m, \beta)$. The mean of the Gumbel distribution is

$$\mu = m + \beta\,\gamma,$$

where $\gamma \approx 0.5772$ is the Euler-Mascheroni constant. The second moment of the Gumbel distribution is

$$E[X^2] = \int_{-\infty}^{\infty} x^2 \frac{1}{\beta}e^{-(z+e^{-z})}\,dx = \beta\pi/\sqrt{6} + (m + \beta\,\gamma)^2.$$

Solving these two equations, we get

$$m = \mu - \frac{\sqrt{6}\gamma\sqrt{E_N[X^2] - \mu^2}}{\pi},$$

$$\beta = \frac{\sqrt{6}\sqrt{E_N[X^2] - \mu^2}}{\pi}.$$

We will draw 1000 samples from a Gumbel distribution with $m = 1$ and $\beta = 2$. Using the method of moments, we get the fit distribution shown in Fig. 7.3. As mentioned before, we could also fit a normal distribution to this data using these samples and the method of moments. This normal distribution is shown in the figure as well. Notice that the normal greatly overestimates the probability of low values and underestimates the prevalence of high values. Despite the fact that the method of moments works for both distributions, the choice of distribution matters greatly.

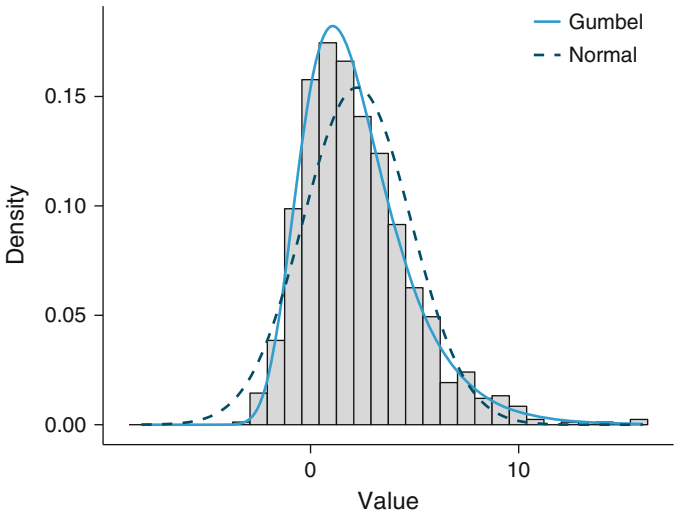

Fig. 7.3 The distribution of 1000 samples from a Gumbel distribution and a normal distribution fit to the data using the method of moments, and a Gumbel distribution fit with the method of moments

Both the method of moments and maximum likelihood have the feature that the choice of distribution matters. One can use Bayesian model selection (Carlin and Louis 2008) or other frequentist methods (Hastie et al. 2009) to help decide which distribution is the best fit. The problem of extreme values remains: without a large number of samples, our knowledge of the tails of the distribution will be limited or absent.

7.2 Design-Based Sampling

The previous section explored the use of random sampling to estimate properties of a distribution. In uncertainty quantification we are often interested in QoIs that have a large number of parameters. Also, quantities of interest are not always described by "nice" distributions: they may have non-smooth regions or cliffs in the value that arise from extreme values of the uncertain parameters. It is often the behavior at these cliffs or extreme points that we are most interested in. For this reason, we desire to make sure that our sampled parameters cover the range of possible values that could occur in the real system.

The process of selecting points to evaluate the QoI at is the problem of *designing* an experiment. The design of experiments is itself a subfield of statistics and entire volumes are devoted to its vagaries. Also, because the intricacies of designing experiments for computer experiments are very different than designing experiments

for pharmaceutical clinical trials or social science,[1] this exercise is called the design of *computer* experiments. The monograph by Santner et al. (2013) is dedicated to this topic.

In computer experiments we do not need replicates because generally computer simulations will give the same result given the same inputs, unless the code is stochastic in some way, such as codes that use Monte Carlo to estimate the outcome of a stochastic process. Therefore, we focus on filling the space of the inputs, or, in the parlance of experimental design, we seek a space-filling design.

Simple random sampling, like that we used in the previous section, is often not adequate because of the tendency of random samples to group near the mode of the distribution. Also, there is no guarantee that two samples will not be close together. In pseudo-Monte Carlo (or pseudo-sampling) based on an experimental design, we address this issue by imposing some structure on the sampling procedure but retain the stochastic nature of the process.

7.2.1 Stratified Sampling

Stratified sampling is an approach to improve the space-filling properties of an experimental design by dividing probability space into several regions and forcing the number of samples in a given region to be a certain number. It is most easily illustrated when sampling from a one-dimensional space.

As discussed in Sect. 2.5, we can sample from a random variable by drawing a random number uniformly distributed between 0 and 1 and then evaluating the inverse CDF of the RV at this random point. Therefore, if we have a means to construct a design on [0, 1], we have a design for any random variable.

In stratified sampling one begins with the number of samples desired, N, and divides the region [0, 1] into M strata, S_m, where

$$S_m = \left[\frac{m-1}{M}, \frac{m}{M}\right], \qquad m = 1, \ldots, M. \tag{7.11}$$

We then select $N_S = N/M$ random numbers distributed uniformly in each stratum. Clearly, if N cannot be exactly divided by M, some rounding will be necessary, and either the number of samples per stratum will not be equal or the number of samples will not equal N. For this purpose we recommend having N be some integer multiple of M.

[1] For some time, design of experiments for social science has consisted of running a trial until one gets the desired answer and then stopping (John et al. 2012). There are efforts to identify and correct this practice (Collaboration et al. 2015).

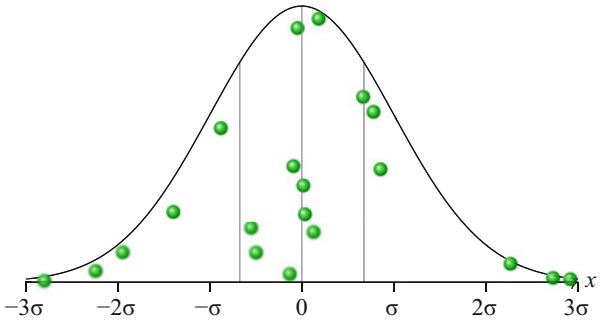

Fig. 7.4 A demonstration of stratified sampling for a standard normal distribution. There are four strata (denoted by solid vertical lines) with five points in each. The y-position of the points is set randomly

In each stratum we will have N_S uniform random variables, t_n. To get the samples from a random variable X, we evaluate

$$x_n = F_X(t_n).$$

Stratified sampling will assure that for M strata there are at least N/M samples in each of the quantiles $[0, M^{-1}]$, $[M^{-1}, 2M^{-1}]$, $\ldots$, $[(M-1)M^{-1}, 1]$. In other words, we are guaranteed to have some number of samples in the extreme values of the distribution with some measure of even distribution in between. For example, if $N = M = 100$, we know that one point will be in the bottom 1% of possible values and one point will be in the top 1% of possible values. This is not guaranteed with random sampling: taking 100 samples may not give any points at this extreme.

In Fig. 7.4 20 points divided into four strata (5 points per stratum) are shown for a standard normal distribution. The strata boundaries define the quartiles of the standard normal distribution. We can see that in each stratum, there are exactly five points: giving five samples per quartile in this case.

We can show that the variance in an estimate from stratified sampling will be lower than one from simple random sampling. As demonstrated by Santner et al. (2013), it is possible to show that the variance in an estimate $I_{N,\text{strat}} \approx E[g(\mathbf{X})]$ can be written as

$$\text{Var}(I_{N,\text{strat}}) = \sum_{m=1}^{M} \left(\frac{V_m^2}{n_m} \right) \sigma_m^2, \tag{7.12}$$

where V_m is the volume of the mth stratum, n_m is the number of samples per stratum, and σ_m^2 is the variance of $g(X)$ over the mth stratum. If the number of strata is equal to the total number of samples, i.e., $M = N$, the strata are all of the same size, $V_m = N^{-1}$ for p the size of $\mathbf{X}$, and the number of samples per stratum is $n_m = 1$, then Eq. (7.12) simplifies to

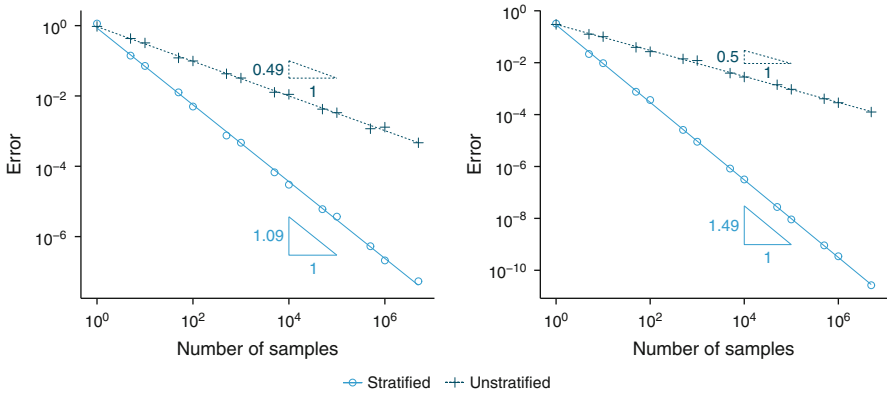

Fig. 7.5 The convergence of the standard deviation of the estimated mean of a standard normal (left) and a uniform distribution (right) using 50 replicates at each value of N with stratified sampling where $M = N$ and standard (unstratified) sampling

$$\mathrm{Var}(I_{N,\mathrm{strat}}) = \frac{1}{N^2} \sum_{m=1}^{N} \sigma_m^2. \tag{7.13}$$

We define $\hat{\sigma}^2 = \max_m \sigma_m^2$ so that

$$\mathrm{Var}(I_{N,\mathrm{strat}}) \leq \frac{1}{N} \hat{\sigma}^2.$$

The best possible value of $\hat{\sigma}^2$ can be shown to be $CN^{-2/p}$ (Carpentier and Munos 2012) (the value of C does depends on the gradient of $g(\mathbf{X})$). Therefore, we can say that the variance in the estimate from stratified sampling with equal-sized strata and a single sample per stratum is

$$CN^{-(1+\frac{2}{p})} \leq \mathrm{Var}(I_{N,\mathrm{strat}}) \leq \frac{1}{N} \hat{\sigma}^2. \tag{7.14}$$

Therefore, stratified sampling will be no worse than simple random sampling and could be much better.

A simple demonstration of stratified sampling can be seen by sampling from a distribution and computing the mean. As we discussed before, repeatedly doing this with simple random sampling will give an estimate of the mean that has a standard deviation of that scales as $N^{-1/2}$, where N is the number of samples. Doing this with stratified sampling, we see that the standard deviation of the estimate decreases faster. Results from a numerical experiment where the standard deviation of the mean is estimated using 50 replicates for each value of N are shown in Fig. 7.5. The figure shows the results for two underlying distributions: a standard normal and a uniform distribution. In the figure we see that stratified sampling with $N = M$

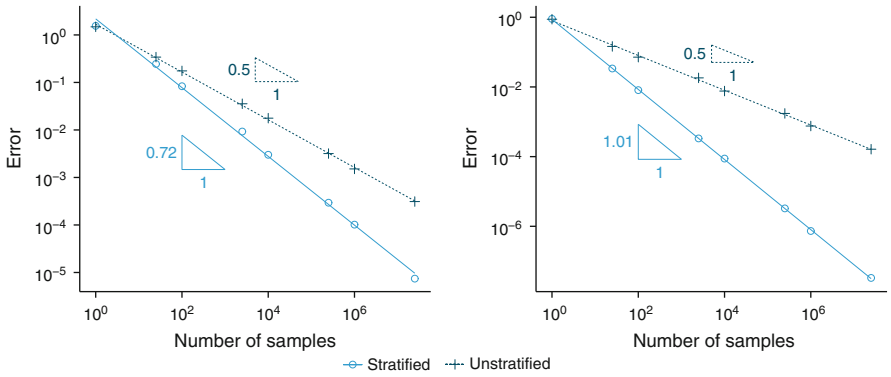

Fig. 7.6 The convergence of the standard deviation of the estimated mean of the 2-D distribution (left) given by Eq. (7.4) and the sum of two uniform random variables (right) using stratified sampling compared with standard (unstratified) sampling

yields estimates with a much lower standard deviation than unstratified sampling. Estimating the mean of the uniform distribution does give the theoretical best convergence for the standard deviation, $O(N^{-3/2})$, whereas the normal converges slightly slower, at about $O(N^{-1})$.

The effect of stratifying in multiple dimensions is demonstrated in Fig. 7.6 where the standard deviation of the mean of a QoI that is a function of two random variables, given in Eq. (7.4), is shown at different values of N. Once again, stratified sampling converges faster than simple random sampling, but the rate has decreased to be less than $O(N^{-1})$. For a simpler estimate, the sum of two uniform random variables, the standard deviation converges to zero at the theoretical rate of $O(N^{-1})$ given by Eq. (7.14).

One of the drawbacks of stratified sampling is that the number of samples required for a full stratification grows geometrically as the number of dimensions increases. For example, to have s strata per dimension will require s^d samples if d is the number of dimensions: this is the dreaded curse of dimensionality. When the number of dimensions is high, full stratification becomes impossible. This is one of the reasons to undertake a variable screening study using inexpensive sensitivity methods.

7.2.2 Latin Hypercube Designs

There is an approach to generating an experimental design using partial stratification that does not grow geometrically in the number of dimensions. This approach has a name that sounds like a device from science fiction: Latin hypercube sampling. The idea behind Latin hypercube sampling is to try to pick a design that fills the design space given a fixed number of samples.

The idea can be demonstrated using in 2-D using a Latin square.[2] The square has a side length of 1 and the divisions in each dimension correspond to the quantile of the variable to be sampled. If we desire N total samples, we divide the square into N^2 equally sized cells. Then in each row, permutations of the integers 1 through N are placed so that no column has an integer repeating. This is reminiscent of the puzzle game sudoku.

For $N = 4$, two possible examples of these permutations are

3	1	4	2
4	3	2	1
2	4	1	3
1	2	3	4

4	3	2	1
3	4	1	2
2	1	4	3
1	2	3	4

The next step is to pick a random integer between 1 and N. This integer then selects the N cells to generate a sample in. In our example, if 4 is the integer chosen, the cells chosen are

3	1	4	2
4	3	2	1
2	4	1	3
1	2	3	4

4	3	2	1
3	4	1	2
2	1	4	3
1	2	3	4

We would then pick a point, at random, in each of the shaded boxes to get our four samples. The design on the right picked the diagonal, which is not ideal because of the correlation between the dimensions. We will revisit this idea later.

To generalize the Latin square to a hypercube, we define a $\mathbf{X} = (X_1, \ldots, X_p)$ as a collection of p independent random variables. To generate N samples, we divide the domain of each X_j in N intervals. In total there are N^p such intervals. The intervals are defined by the $N + 1$ edges:

$$\left\{ F_j^{-1}(0), F_j^{-1}\left(\frac{1}{N}\right), F_j^{-1}\left(\frac{2}{N}\right), \ldots F_j^{-1}\left(\frac{N-1}{N}\right), F_j^{-1}(N) \right\}.$$

[2] The terminology hypercube comes from the fact that we use a square in p dimensions to formulate the design.

To choose which combinations of intervals get samples, we define a permutation matrix $\boldsymbol{\Pi}$ of size $N \times p$ with elements π_{ij} where the columns are p different, randomly selected permutations of the integers $\{1, 2, \ldots, N\}$. To generate the ith sample in dimension j, we evaluate

$$x_{ij} = F_j^{-1}\left(\frac{1}{N}(\pi_{ij} - 1 + u_{ij})\right), \tag{7.15}$$

where $u_{ij} \sim \mathscr{U}(0, 1)$. This makes $\mathbf{x}_i = (x_{i1}, \ldots, x_{iN})$. the ith sample of $\mathbf{X}$.

As an example we look at a case with $p = 3$ and $N = 4$. In this case a possible matrix $\boldsymbol{\Pi}$ is

$$\boldsymbol{\Pi} = \begin{pmatrix} 4 & 1 & 2 \\ 3 & 3 & 1 \\ 2 & 2 & 3 \\ 1 & 4 & 4 \end{pmatrix}.$$

The samples are then

$$x_{11} = F_1^{-1}\left(\frac{3 + u_{11}}{4}\right) \quad x_{12} = F_2^{-1}\left(\frac{u_{12}}{4}\right) \quad x_{13} = F_3^{-1}\left(\frac{1 + u_{13}}{4}\right),$$

$$x_{21} = F_1^{-1}\left(\frac{2 + u_{21}}{4}\right) \quad x_{22} = F_2^{-1}\left(\frac{2 + u_{22}}{4}\right) \quad x_{23} = F_3^{-1}\left(\frac{u_{23}}{4}\right),$$

$$x_{31} = F_1^{-1}\left(\frac{1 + u_{31}}{4}\right) \quad x_{32} = F_2^{-1}\left(\frac{1 + u_{32}}{4}\right) \quad x_{33} = F_3^{-1}\left(2 + \frac{u_{33}}{4}\right),$$

$$x_{41} = F_1^{-1}\left(\frac{u_{41}}{4}\right) \quad x_{42} = F_2^{-1}\left(\frac{3 + u_{42}}{4}\right) \quad x_{43} = F_3^{-1}\left(3 + \frac{u_{43}}{4}\right).$$

Notice that each of the four intervals in each dimension are only sampled once.

Latin hypercube designs can be shown to be an improvement on simple random sampling by looking at how the "main effects" of the function we are trying to estimate the expected value of. Say we are interested in $E[g(Q)]$ as estimated by Eq. (7.2). If the random variable space is p dimensional, there are p main effects defined by the function of x_p:

$$\begin{aligned} \alpha_j(x_j) = \int dx_1 \cdots \int dx_{j-1} \int dx_{j+1} \cdots \int dx_p \, (g(q(\mathbf{x})) \\ - E[g(Q)])\, f(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_p). \end{aligned} \tag{7.16}$$

The main effects give a measure of how much single dimension of the input space affects the behavior of $g(Q)$ about its mean. It can be shown (Stein 1987) that unless

the main effects are all zero, Latin hypercube will have a superior convergence of the error compared to simple random sampling. Indeed, the integral of the main effects squared gives the improvement of Latin hypercube sampling over simple random sampling.

7.2.3 Choosing a Latin Hypercube Design

We noted above that some designs generated by a Latin hypercube are better than others. Because the selection of intervals is chosen using random permutations, it is possible that the design does not fill the space optimally. Also, when the points are chosen inside the intervals, they could be close together due to the random placement. To address this we introduce the distance between any two points:

$$\rho_\ell(\mathbf{x}, \mathbf{y}) = \left(\sum_{j=1}^{p} |x_j - y_j|^\ell \right)^{1/\ell}. \tag{7.17}$$

For a given design, $\mathscr{X} = \{\mathbf{X}_1, \cdots, \mathbf{X}_N\}$, the minimum distance between two points can be defined as

$$\rho_\ell(\mathscr{X}) = \min_{1 \le i, j \le N} \rho_\ell(\mathbf{x}_i, \mathbf{x}_j). \tag{7.18}$$

Therefore, if we generate many designs, we can compare them based on this minimum distance and choose the design with maximum minimum distance between points. Such a design is called the minimax distance design. This method can be done in either the random variable space, X_j, or in quantile space $F_j(X_j)$ to deal with the fact that distance may have different meanings in each space (such as dimension 1 being a normal random variable and dimension 2 being uniform).

There are other ways to choose between designs based on maximizing the average distance between points, rather than the minimum distance. This and other distance considerations are discussed in Santner et al. (2013).

7.2.4 Orthogonal Arrays

Latin hypercube designs assure that a point in each of the N intervals is selected for each of the p dimensions in $\mathbf{X}$. We could extend this to ask, for example, could we create a design that selects every pair of intervals, i.e., if I project the design onto any 2-D plane, the sampling fills the space. This can be generalized to other groupings of intervals.

To construct designs that have this desired projection property, we can use an orthogonal array. An orthogonal array O of strength t on s intervals is an $N \times p$ matrix, where $N = \lambda s^t$ and the number of dimensions $p \geq t$ and has property that in every $N \times t$ submatrix of the orthogonal array the s^t, possible rows appear λ times. The parameter λ can be thought of as the number of replicates, so in computer simulations λ is typically set to 1.

To unpack the definition of an orthogonal array, it creates a design of N points such that when one projects into a t-dimensional space, every interval is covered. In this sense, when $t = 1$, we get a Latin hypercube design because each interval is chosen once. Additionally, orthogonal arrays of strength 2 are the basis for factorial experimental designs.

For an example we consider a four-dimensional space ($p = 4$), with three intervals in each dimension ($s = 3$) and a strength of 2 ($t = 2$). There will be $3^2 = 9$ samples in each pair of dimensions. An orthogonal array for this situation is

$$\mathbf{O} = \begin{pmatrix} 3 & 2 & 1 & 3 \\ 1 & 2 & 3 & 2 \\ 2 & 1 & 3 & 3 \\ 1 & 3 & 2 & 3 \\ 2 & 2 & 2 & 1 \\ 2 & 3 & 1 & 2 \\ 3 & 3 & 3 & 1 \\ 3 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Each row in this array gives an interval to pick a point in, just like the matrix $\boldsymbol{\Pi}$ did for a Latin hypercube. The samples corresponding to each entry in the matrix are generated using Eq. (7.15). From this example orthogonal array, we get the design shown in Fig. 7.7.

The generation of orthogonal arrays is not straightforward. For R, the package `DoE.base` will generate strength 2 orthogonal arrays with the `oa.design` function. Python has the `OApackage` for generating these arrays as well.

7.3 Quasi-Monte Carlo

Quasi-Monte Carlo dispenses with the notion of using random numbers in the sampling and uses sequences of seemingly random numbers. These sequences can be designed so that they are space filling and can be rapidly generated. The sequences of samples are often called low-discrepancy sequences because there is a measure of uniformity in how they fill the space, i.e., they do not leave large gaps.

The simplest low-discrepancy sequence is the van der Corput sequence. For a given base, b, this sequence takes the integers $n = 1, \ldots, N$ and for each,

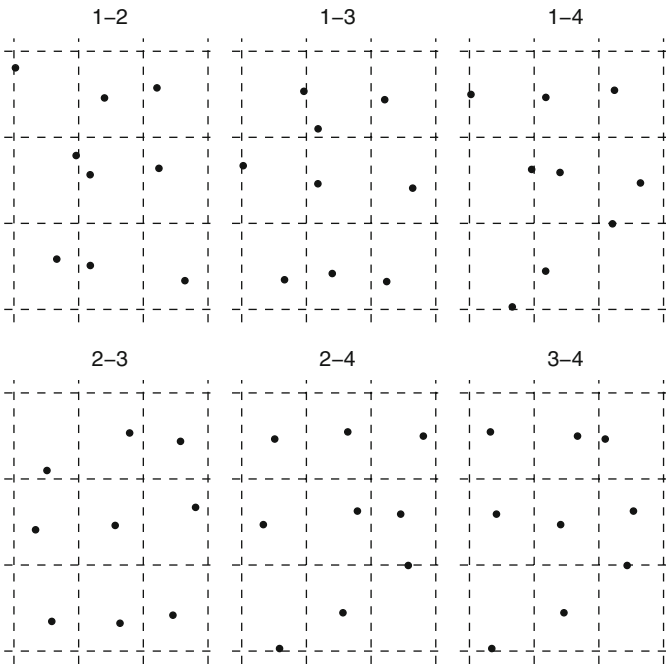

Fig. 7.7 A design generated by an orthogonal array on four dimensions, with three intervals per dimension and strength 2. The headings indicate the dimension shown on the x and y axis, respectively. Note that every possible 2-D projection fills the nine possible pairs of intervals

1. Writes n in base b,
2. Reflects that number about the ones place to create a rational number, and
3. Writes the resulting number as a decimal.

As an example consider $b = 2$ and $n = 2$. In base 2, $2 = (10)_2$, where the subscript denotes the base. Reflecting this number gives $(.01)_2$ in base 2 which is $2^{-2} = 0.25$. With $n = 3$ we have $3 = (11)_2$ and $(.11)_2 = 2^{-1} + 2^{-2} = \frac{3}{4}$. The van der Corput sequence in base 2 is

$$\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \frac{1}{16}, \frac{9}{16}, \frac{5}{16}, \frac{13}{16}, \frac{3}{16}, \frac{11}{16}, \frac{7}{16}, \frac{15}{16}, \ldots$$

The first eight points of the van der Corput sequence base 2 are shown in Fig. 7.8. Notice that the sequence moves to fill in the largest gap in the interval for each point added.

Using the van der Corput sequence, we can use the sequence points to be a uniform random number to use for sampling. Though the formula will generate numbers for any base b, the base must be prime to avoid repeated numbers. Also, we need to generalize the prescription to sample from a multidimensional distributions: if we use van der Corput with the same base in each dimension, we will only sample the diagonal.

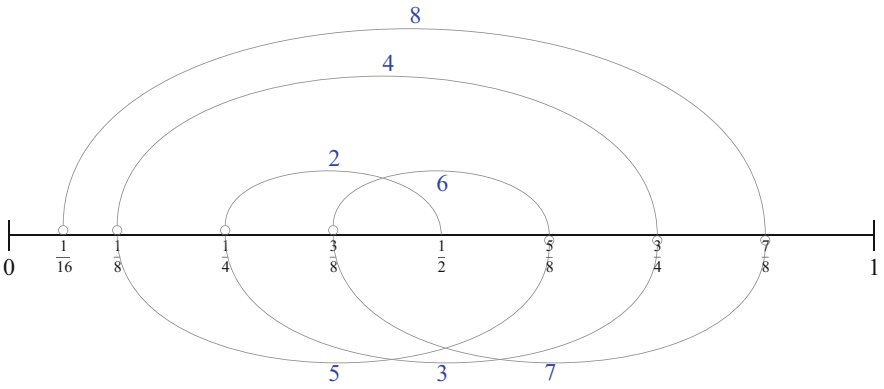

Fig. 7.8 The first eight points of the base 2 van der Corput sequence. The first point is $\frac{1}{2}$; the paths between subsequent points are labeled

7.3.1 Halton Sequences

The Halton sequence is the generalization of van der Corput sequences to multiple dimensions. Each dimension has a different prime base in the sequence. This effectively generalizes the van der Corput sequence to multiple dimensions and is simple to generate the points. However, there is a drawback in that when the prime number used for the base is large, the number of consecutive samples where the sequence has a monotonic behavior (i.e., sample $n + 1$ is greater than or less than n for many consecutive n) causes the Halton sequence to not behave in a seemingly random manner or to fill the space.

A demonstration of the behavior of Halton sequences for different numbers of dimensions is shown in Figs. 7.9 and 7.10. In the five-dimensional case in Fig. 7.9, the space is reasonably filled with a small correlation between the variables. However, when the dimension is increased to 40, Fig. 7.10, there is a clear correlation between certain variables, and there are large gaps of unfilled space.

For these reasons, Halton sequences are not suggested for input parameter spaces larger than about eight dimensions. There are alternatives to the Halton sequence that utilize van der Corput sequences. One possibility is the Faure sequence which reorders the van der Corput sequence (Faure 1982).

7.3.2 Sobol Sequences

Another common possible low-discrepancy sequence that can be used in quasi-Monte Carlo is the Sobol sequence (Sobol 1967). This sequence was designed to make integral estimates on the p-dimensional hypercube converge as quickly as possible. The details of the sequence require a background in number theory and primitive polynomials that would take us too far afield. We demonstrate in Figs. 7.11

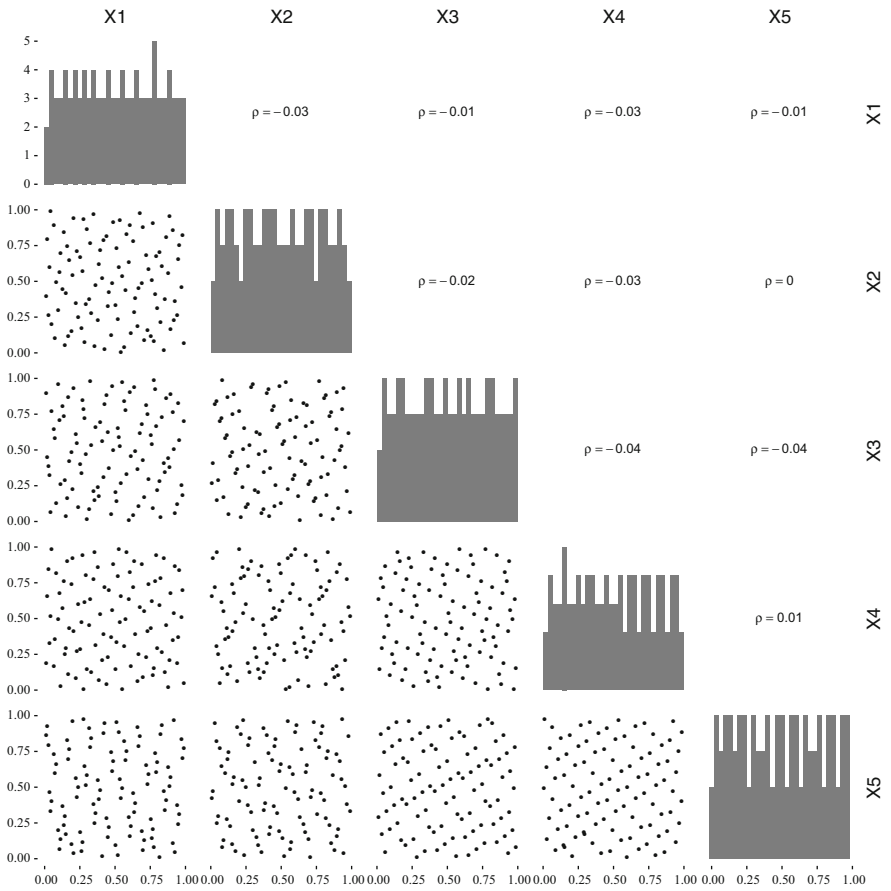

Fig. 7.9 The pairwise projections for the variables $X1$ through $X5$ in a 5-D Halton sequence with 100 points. The upper part of the diagram gives the Spearman correlation (ρ) between the two variables; the diagonal shows a histogram for each variable

and 7.12 that the performance for Sobol sequences is comparable in five dimensions, and slightly improved at 40 dimensions, relative to the Halton sequence. In 40 dimensions the Sobol sequence does not have the high correlation that the Halton sequence did, but there is a noticeable relation between variables, especially in $X11$ and $X12$.

7.3.3 Implementations of Low-Discrepancy Sequences

There are implementations of low-discrepancy sequences available for R in the `randtoolbox` library. This library includes Halton and Sobol sequences. For

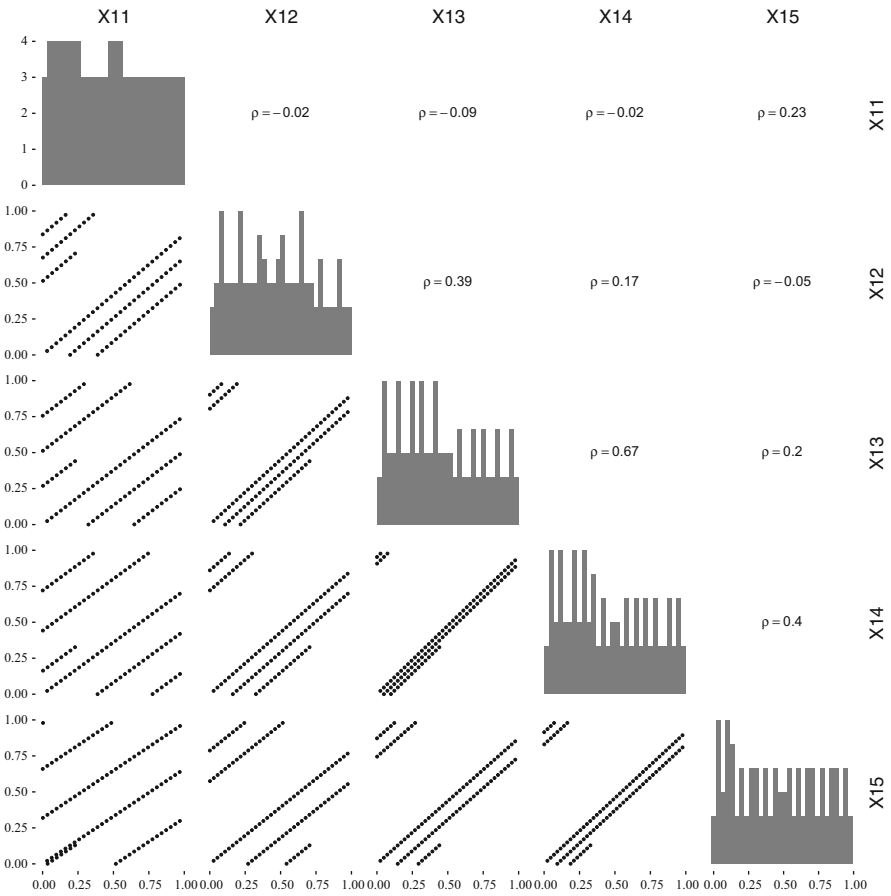

Fig. 7.10 The pairwise projections for the variables $X11$ through $X15$ in a 40-D Halton sequence with 100 points. The upper part of the diagram gives the Spearman correlation (ρ) between the two variables; the diagonal shows a histogram for each variable

Python there are separate libraries for Halton and Sobol sequences available. Many of these implementations are based on the work of Fox (1986) and Bratley et al. (1992).

7.4 Comparison of Techniques

To compare the different sampling techniques, we will turn to the advection-diffusion-reaction problem initially described in Sect. 4.3.1. In this case we will make the distributions of the five parameters non-normal but joined via a normal copula (c.f., Sect. 3.2.1). The distributions will have the same means as before, but

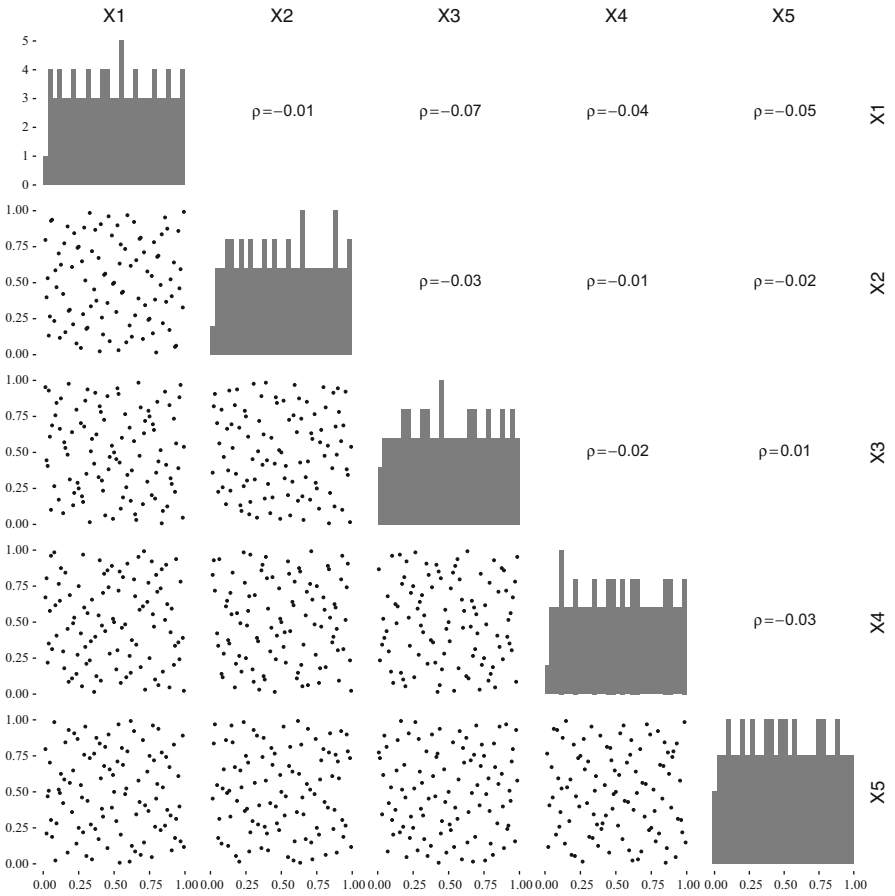

Fig. 7.11 The pairwise projections for the variables $X1$ through $X5$ in a 5-D Sobol sequence with 100 points. The upper part of the diagram gives the Spearman correlation (ρ) between the two variables; the diagonal shows a histogram for each variable

rather than each being normal, each parameter will be gamma distributed with the parameters chosen so that the standard deviation is 10% of the mean. Additionally, the normal copula uses the same correlation matrix as in Sect. 4.3.1.

The resulting distributions of the quantity of interest, the total reaction rate, are shown in Fig. 7.13 for different sampling techniques and numbers of samples. At a low number of samples, $N = 100$, none of the methods match the reference solution, but we do see that the quasi-Monte Carlo designs, namely, Halton and Sobol sampling, do seem to steadily improve as N increases. Simple random sampling (denoted as SRS on the plot) demonstrates the most variability when changing the number of points: as N increases the overall behavior seems to improve, but there are idiosyncratic spikes in the plot that behave randomly because the samples are random. The Latin hypercube samples (LHS) are in between SRS

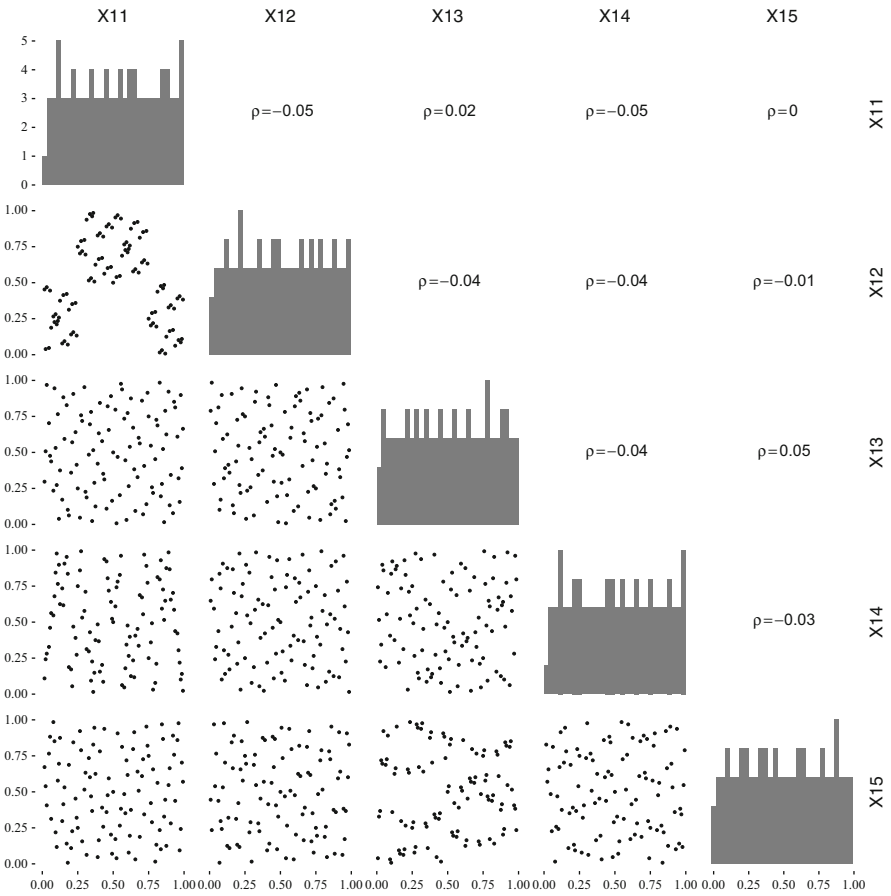

Fig. 7.12 The pairwise projections for the variables $X11$ through $X15$ in a 40-D Sobol sequence with 100 points. The upper part of the diagram gives the Spearman correlation (ρ) between the two variables; the diagonal shows a histogram for each variable

and quasi-Monte Carlo in these regards. The LHS solution does improve noticeably as N increases, in a similar manner to quasi-Monte Carlo, but there are still small artifacts from the random sampling inside the strata in LHS (Fig. 7.14).

To explore how increasing the dimension of the input space affects the performance of sampling methods, we return to the ADR problem where the value of κ was the result of a random process. In Sect. 4.3.2 we defined this problem and solved it with 2000 mesh cells. In this test we will use 40 mesh cells to get a 40-dimensional input space because that was the largest dimension available for the Sobol sampler available for Python. The problem sets that value for all the parameters at the mean of values, except for κ which is set as a Gaussian random process with known covariance function.

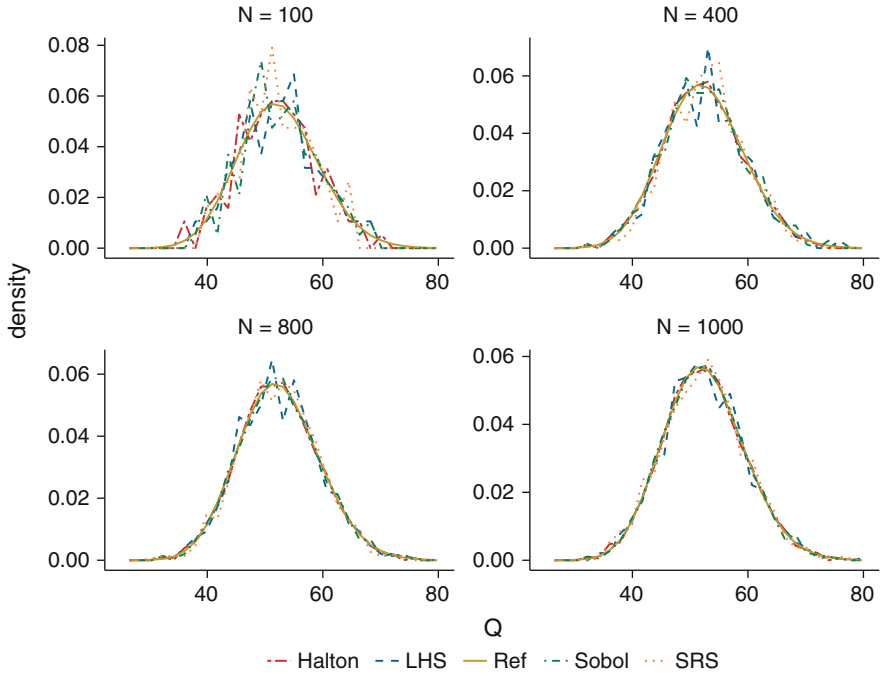

Fig. 7.13 Empirical distributions of the QoI from the ADR problem using different methods and number of samples. The reference distribution is a Latin hypercube design with 10^6 points

The resulting empirical distributions from running several different numbers of samples are compared to a reference Latin Hypercube result using 10^7 samples in Fig. 7.15. In these distributions we see that for a larger dimensional sampling space, the low-discrepancy sequences do not perform as well as the previous example. At $N = 100$ the Halton sequence has many more instances of low values of the QoI; the Sobol results show an artificially high peak to the left of the mode of the reference solution. At $N = 1000$ both Halton and Sobol results have a narrower and more peaked distribution than the other results. The Latin hypercube and simple random sampling results do not have these same artifacts, despite some expected errors at $N = 100$.

When we look at the moments of the QoI in Fig. 7.16, the Latin hypercube and simple random sampling results are superior to Halton and Sobol sampling for this high-dimensional problem. For the variance and higher moments, the Halton sequence results have a large error (in the thousands of percent for kurtosis and skew). They do, nevertheless, have a good rate of convergence, demonstrating that convergence is not necessarily as important as overall error.

The Sobol results, while better than Halton, are generally inferior to the random sampling techniques, except in the estimate of the mean where it is superior to SRS. For the kurtosis estimate, Sobol sampling appears to be giving a more accurate

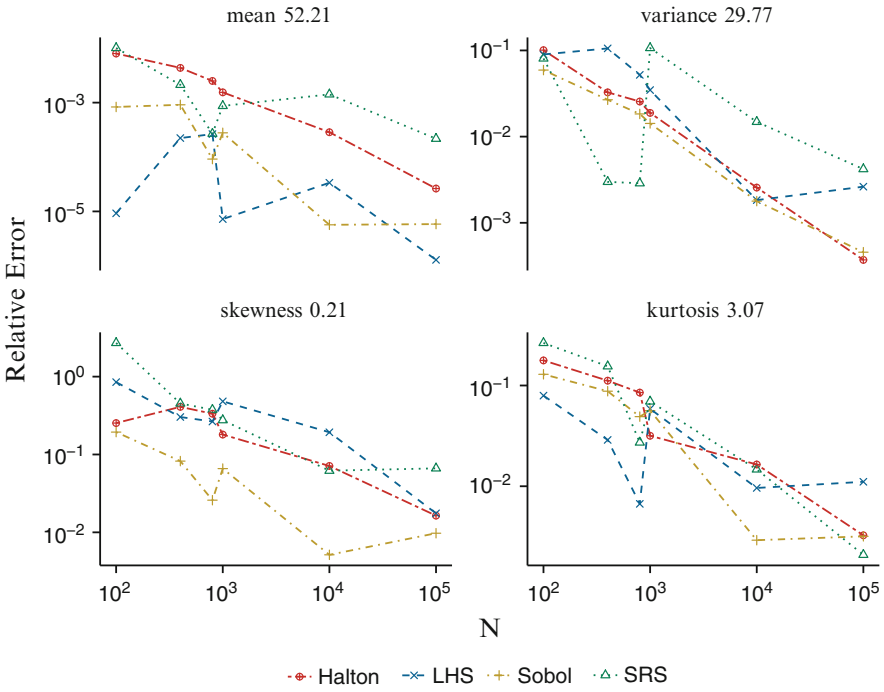

Fig. 7.14 Convergence of the moments of the QoI from the ADR problem using different methods and number of samples. The reference distribution is a Latin hypercube design with 10^6 points

estimate than LHS for N around 1000, but this appears to be an anomaly because as more points are added, the estimate does not improve, and, indeed, the Sobol estimate of kurtosis has an error of about 100% with $N = 10^5$.

These results demonstrate that as the dimensionality of the space gets bigger, the QMC approaches we have discussed may not be adequate for estimating the distributions of QoIs. When the dimensionality of the input space is smaller, as we saw when $p = 5$, these methods appear to be superior to the random and design-based sampling techniques. Also in every case we tested, Latin hypercube sampling was superior to simple random sampling, making it a method that should be used when possible over a pure Monte Carlo strategy.

7.5 Notes and References

Our discussion of sampling did not include the ideas of importance sampling, biased sampling techniques, or other specialized Monte Carlo variance reduction techniques. These techniques are often highly customized to the particular problem at hand and are, therefore, less amenable to a general prescription for uncertainty

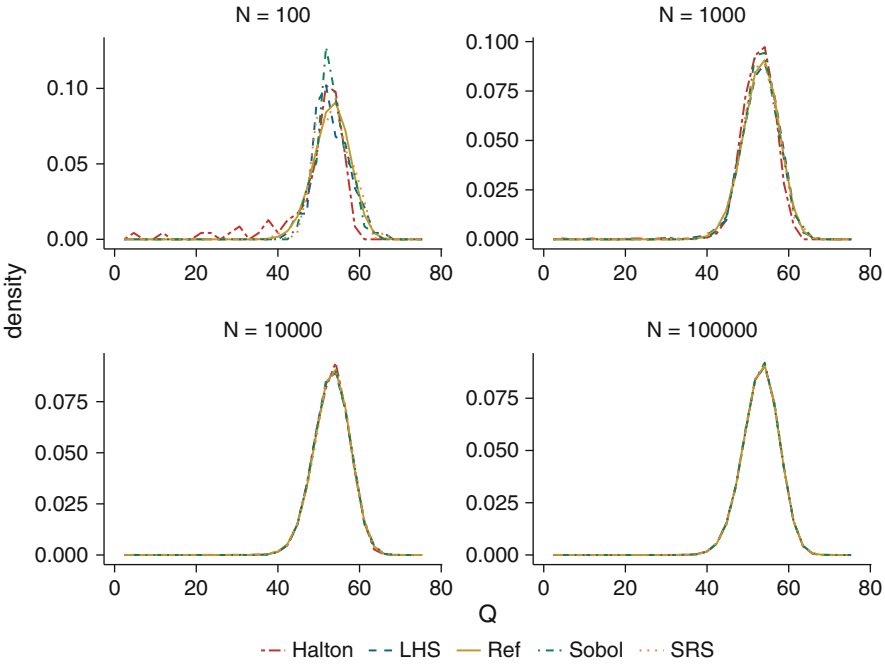

Fig. 7.15 Empirical distributions of the QoI from the random process ADR problem using different methods and number of samples. The reference distribution is a Latin hypercube design with 10^7 points

quantification purposes. The works of Robert and Casella (2013) and Kalos and Whitlock (2008) are appropriate references for these topics.

There also has been recent work on using low-resolution calculations in Monte Carlo estimates and correcting these low-resolution calculations with high-resolution calculations in a method known as multilevel Monte Carlo (MLMC) (Giles 2013; Cliffe et al. 2011). The basic idea is that the numerical calculation of a QoI at a low resolution Q_0 and at higher resolutions Q_ℓ for $\ell = 1, \ldots, L$ can be combined to form an estimate of the expected value of Q_L, the highest-resolution estimate, using the linearity of the expected value operator:

$$E[Q_L] = E[Q_0] + \sum_{\ell=1}^{L} E[Q_\ell - Q_{\ell-1}].$$

Then we use different Monte Carlo estimators for each expected value as

$$E[Q_0] = \frac{1}{N_0} \sum_{n=1}^{N_0} Q_{0,n}, \qquad E[Q_\ell - Q_{\ell-1}] = \frac{1}{N_\ell} \sum_{n=1}^{N_\ell} Q_{\ell,n} - Q_{\ell-1,n}.$$

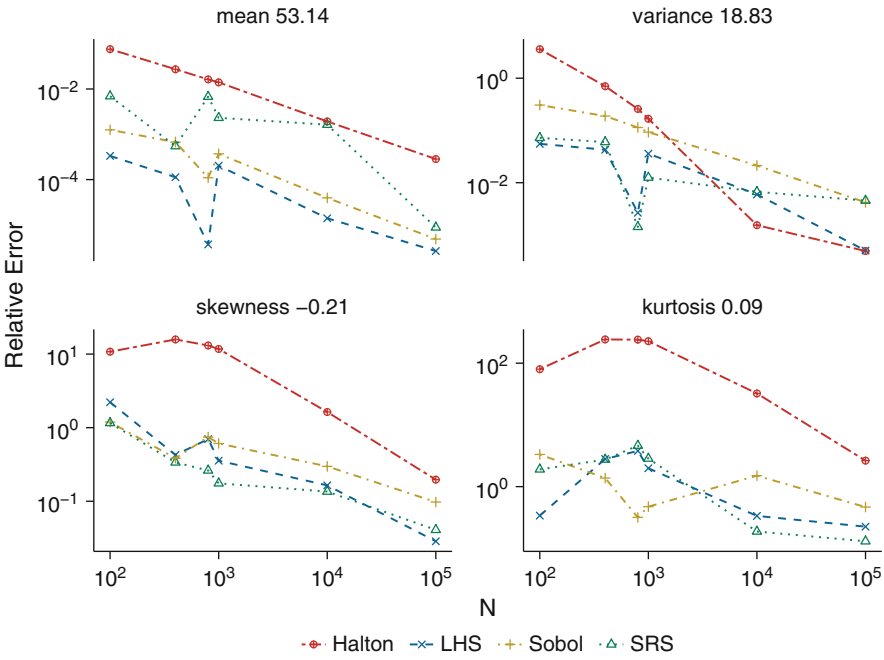

Fig. 7.16 Convergence of the moments of the QoI from the random process ADR problem using different methods and number of samples. The reference distribution is a Latin hypercube design with 10^7 points

If $N_L \leq N_{L-1} \leq \cdots \leq N_0$, then the estimate will be less costly than standard Monte Carlo provided that the numerical error and the variance in the estimators go to zero at an appropriate rate. Moreover, design of computer experiments and quasi-Monte Carlo can be used to improve the estimators.

There are many subtleties to MLMC. These include selecting the resolution for Q_0: if the calculation is too coarse, the errors in the estimate may be too large to give an effective estimate. Additionally, a series of estimates of the QoI at different resolutions may not be readily available. MLMC is an active area of research (Barth et al. 2011; Gunzburger et al. 2014; Collier et al. 2015).

7.6 Exercises

1. For the random variable $X \sim \mathcal{N}(0, 1)$, draw 50 samples and generate histograms using the following sampling techniques:

   (a) Simple random sampling,
   (b) Stratified sampling,

(c) A van der Corput sequence of base 2,
(d) A van der Corput sequence of base 3.

Additionally, compare the medians for each method.

2. Consider an experiment that tossed a coin 80 times with 33 heads and 47 tails. Use a maximum likelihood estimator and the method of moments to estimate the probability of getting heads assuming that the outcome is described by a binomial distribution.
3. Consider the Rosenbrock function: $f(x, y) = (1-x)^2 + 100(y-x^2)^2$. Assume that $x = 2t - 1$, where $T \sim \mathscr{B}(3, 2)$ and $y = 2s - 1$, where $S \sim \mathscr{B}(1.1, 2)$. Estimate the probability that $f(x, y)$ is less than 10 using

(a) Latin hypercube sampling using 50 points.
(b) A strength 2 orthogonal array on seven intervals (49 points).
(c) A Halton sequence using 50 points.
(d) Simple random sampling with 50 points.

Compare this with the probability you calculate using 10^5 random samples.

4. Consider the exponential integral function, $\mathrm{E}_n(x)$,

$$\mathrm{E}_n(x) = \int_1^\infty \frac{e^{-xt}}{t^n}\, dt.$$

This function is involved in the solution to many pure-absorbing radiative transfer problems. Use this function to solve the problem,

$$\mu \frac{\partial \psi}{\partial x} + \sigma \psi = 0,$$

$$\psi(0, \mu > 0) = \alpha, \psi(10, \mu < 0) = 0,$$

for the scalar intensity $\phi(x) = \int_{-1}^{1} \psi(x, \mu)\, d\mu$. Assume that σ and α are each independently gamma-distributed with mean 1 and variance 0.01. Using $N = 10, 100, 1000$, use Latin hypercube sampling, a Halton sequence, and simple random sampling to estimate the distribution, mean, and variance of $\phi(x)$ at $x = 1, 1.5, 3, 5$. Also, for each estimate, plot the mean value of ϕ as a function of x with error bars giving a 90% confidence interval (i.e., have error bars that show the range from the 5th to the 95th percentile at each x point). Compare your result to an 10^5 point design using simple random sampling.

Chapter 8
Reliability Methods for Estimating the Probability of Failure

> Dr. Peter Venkman: *You're gonna endanger us, you're gonna endanger our client—the nice lady, who paid us in advance, before she became a dog...*
>
> Dr. Egon Spengler: *Not necessarily. There's definitely a VERY SLIM chance we'll survive.*
>
> —from the film *Ghostbusters*

Reliability methods are a class of techniques that seek to answer the question of with what probability a QoI will cross some threshold value. The name for the methods comes from civil engineering where they were originally formulated to answer the question of when the amount of margin in the system is smaller than zero, i.e., the system fails. These methods typically try to answer this question using approximations to the distribution based on a minimal set of evaluations of the QoI.

Reliability methods will try to characterize the safety of the system using a single number, β, which is the probability of not failing, expressed as the number of standard deviations above the mean performance where the failure point of the system is. While it is a laudable goal to have a single metric to report to other stakeholders and decisionmakers, as we shall see, many details necessarily get obfuscated in doing so.

As mentioned, reliability methods try to estimate the system performance using a minimal number of QoI evaluations to infer system behavior: an endeavor that necessarily requires extrapolation from a few data points to an entire distribution. This contrasts with the previous chapter on sampling methods where we used actual samples from the distribution of the QoI to make statements about a distribution, at the cost of requiring many evaluations of the QoI. As a result of the fewer evaluations required in reliability analysis, it can be much faster than sampling. On the other hand, the simplifications made in these methods make it less robust than sampling techniques. The assumptions and approximations in a reliability calculation should be noted by a practitioner.

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0_8

8.1 First-Order Second-Moment (FOSM) Method

The simplest and least expensive type of reliability method involves extending the sensitivity analysis we have already completed to make statements about the values of the distribution. The first-order second-moment (FOSM) method uses first-order sensitivities to estimate the variance. Then, using the assumption that the value of the QoI at the mean of the inputs is the mean of the QoI, i.e.,

$$\overline{Q(\mathbf{X})} = Q(\bar{\mathbf{x}}). \tag{8.1}$$

An additional assumption is that the QoI is normal with a known mean and variance.

We use the covariance matrix for the inputs, along with the sensitivities, $\frac{\partial Q}{\partial X_i}$, to estimate the variance as (c.f. Eq. (4.11))

$$\mathrm{Var}(Q) \approx \frac{\partial Q}{\partial \mathbf{X}}^{\mathrm{T}} \Sigma \frac{\partial Q}{\partial \mathbf{X}}.$$

With the mean and variance in hand, we can then assume, without any justification at this point, that Q is normally distributed as

$$Q \sim \mathcal{N}\left(Q(\bar{\mathbf{x}}), \frac{\partial Q}{\partial \mathbf{X}}^{\mathrm{T}} \Sigma \frac{\partial Q}{\partial \mathbf{X}}\right). \tag{8.2}$$

This assumption will only be valid if the QoI is a linear function of the inputs *and* if the inputs are independent and normally distributed.

Reliability analysis typically rescales the QoI so that the point we are interested in, the so-called failure point, is expressed as a quantity, Z, such that failure occurs when $Z < 0$. Therefore, we use the failure value of the QoI, Q_{fail}, to define Z:

$$Z(\mathbf{X}) = Q_{\text{fail}} - Q(\mathbf{X}). \tag{8.3}$$

When $Q(\mathbf{X})$ exceeds the failure point, Z will be negative.

Some context is in order at this point. If we consider the example where the QoI is the load on a structure, and Q_{fail} is the load at which the structure collapses, then Z is the amount of margin in the system for a given realization, i.e., how much extra load-carrying capacity does the system have. Indeed, reliability methods are not confined to structural analysis. When one is interested in the QoI exceeding some threshold, a variable Z can be defined such that Z is negative whenever that threshold is exceeded.

Given that we have assumed that the QoI is normal in Eq. (8.3), Z will also be normally distributed and have a mean of $Q_{\text{fail}} - Q(\bar{\mathbf{X}})$. Therefore, the probability of failure is

$$P(Z < 0) = \Phi\left(\frac{0 - (Q_{\text{fail}} - Q(\bar{\mathbf{X}}))}{\sqrt{\frac{\partial Q}{\partial \mathbf{X}}^{\mathrm{T}} \Sigma \frac{\partial Q}{\partial \mathbf{X}}}}\right) = 1 - \Phi\left(\frac{Q_{\text{fail}} - Q(\bar{\mathbf{X}})}{\sqrt{\frac{\partial Q}{\partial \mathbf{X}}^{\mathrm{T}} \Sigma \frac{\partial Q}{\partial \mathbf{X}}}}\right), \tag{8.4}$$

where $\Phi(x)$ is the standard normal CDF. The probability of failure leads to the definition of the reliability index for the system. The reliability index, β, is defined as

$$\beta = \frac{Q_{\text{fail}} - Q(\bar{\mathbf{X}})}{\sqrt{\frac{\partial Q}{\partial \bar{\mathbf{X}}}^{\mathrm{T}} \Sigma \frac{\partial Q}{\partial \bar{\mathbf{X}}}}}. \tag{8.5}$$

This makes $1 - \Phi(\beta)$ the estimated probability of failure. β is simply the number of standard deviations above 0 the mean system performance is. A larger value of β indicates that the system is farther from the failure point at the nominal conditions. In other words, β indicates how many standard deviations of margin are available when the QoI is evaluated at the mean value of the inputs. Of course, there have been many assumptions that went into calculating β, and one should consider these when using β to make quantitative statements. On the other hand, even an approximate indicator like β can be quite useful when comparing two different systems in terms of reliability.

As a simple example, consider the QoI defined by the linear combination of independent normal random variables:

$$Q(x, y) = 2x + 0.5y, \qquad X \sim \mathcal{N}(5, 2), \quad Y \sim \mathcal{N}(3, 1).$$

The QoI will then be normally distributed with mean 11.5 and standard deviation of 4.03 as shown in Fig. 8.1. If the failure point is $Q_{\text{fail}} = 16.5$, then the reliability

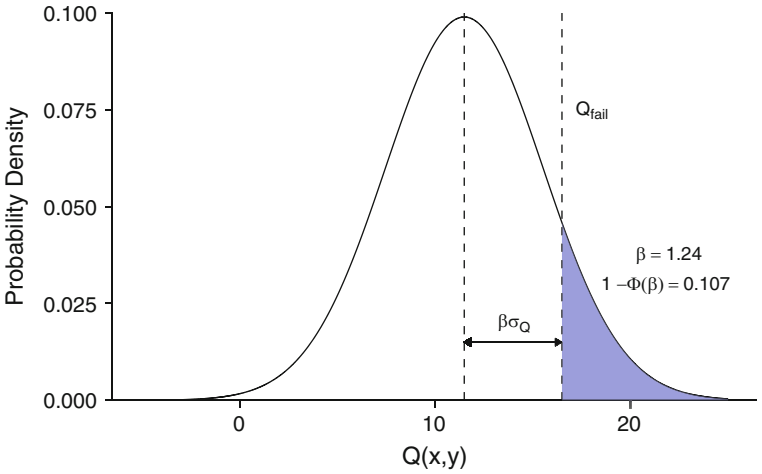

Fig. 8.1 Illustration of the reliability index for the QoI $Q(x, y) = 2x + 0.5y$, $X \sim \mathcal{N}(5, 2)$, $Y \sim \mathcal{N}(3, 1)$ with a failure point $Q_{\text{fail}} = 16.5$. The shaded area is the probability of failure, and $\beta\sigma_Q$ is the distance from the mean to the failure point

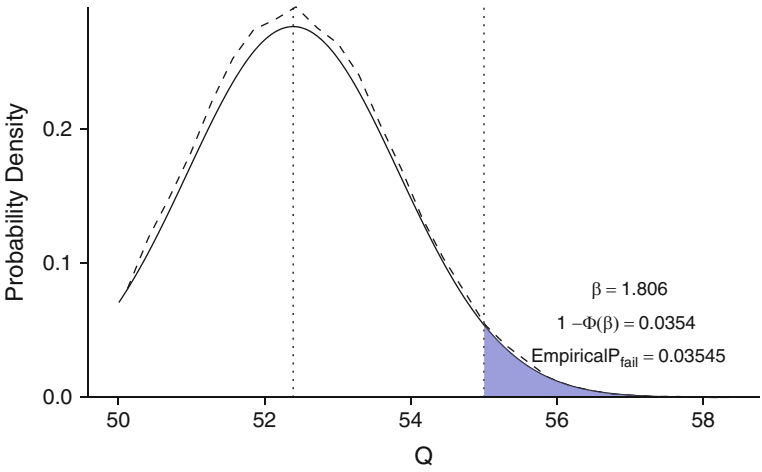

Fig. 8.2 Comparison of FOSM with a 4×10^4 sample Monte Carlo design to estimate the probability of failure in the ADR example with multivariate normal inputs. The solid line is the distribution fit by FOSM, and the dashed line is empirical probability density. The shaded region has an area of $1 - \Phi(\beta)$

index is $\beta = (16.5 - 11.5)/4.03 = 1.24$, and the probability of failure is 10.7%. In this example, all the assumptions of FOSM are satisfied (the input variables are normal and independent and the QoI is linear).

We can perform a similar analysis on a problem where the first-order second-moment analysis will be an approximation. Consider the advection-diffusion-reaction (ADR) problem from Sect. 4.3.1. In that example we used derivative approximations to estimate the variance of a quantity of interest that was the total reaction rate in the system. We estimated that the variance in the QoI was 2.0876; the response at the mean was 52.390. In this problem the inputs were normally distributed, but they were not independent. If we say that the failure point in the system is $Q_{\text{fail}} = 55$, using FOSM we get a reliability index of $\beta = (55 - 52.390)/\sqrt{2.0876} = 1.806$, for a probability of failure of 3.54%.

When we compare the FOSM result with a Monte Carlo result using random sampling and 4×10^4 samples, we get an estimate of the probability of failure of 3.545%. The empirical density from the samples and the normal distribution fit with FOSM are shown in Fig. 8.2. Not only do the probabilities of failure agree well, but the probability densities show good agreement as well. For this problem, the FOSM method required six evaluations of the QoI, and FOSM is nearly 10,000 times faster than using sampling to get effectively the same answer.

If we change the input distribution to be nonnormal, we would expect a larger discrepancy between FOSM and the results from sampling. To this end we reprise the ADR solutions from Sect. 7.4. In that example the five input parameters were each a gamma random variable with mean at the nominal value and a standard

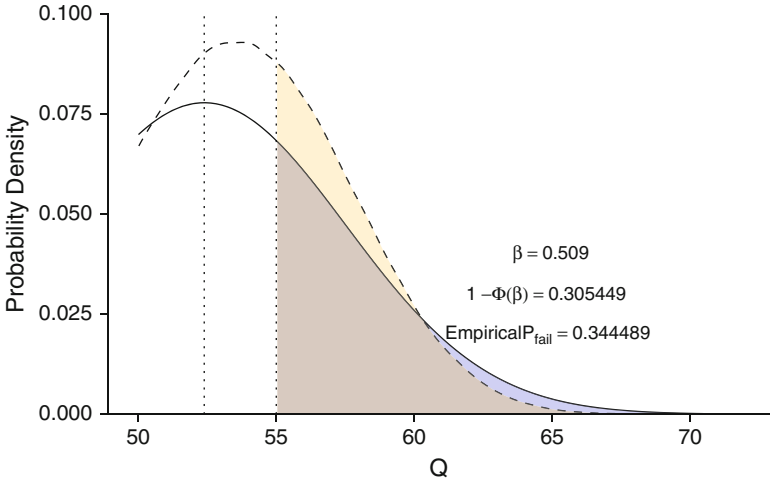

Fig. 8.3 Comparison of FOSM with a 10^6 sample Monte Carlo design to estimate the probability of failure for the ADR example where the inputs are gamma random variables joined by a normal copula. The solid line is the distribution fit by FOSM, and the dashed line is empirical probability density

deviation that is 10% of the mean. These variables are joined via a normal copula where the correlation matrix is given by Eq. (4.15).

In Fig. 8.3 the results from a Monte Carlo estimate using 10^6 samples and FOSM are shown. While the FOSM estimate for the probability of failure is close to that estimated via Monte Carlo (30.55% for FOSM and 34.45% for MC), it appears to be getting the right answer for the wrong reasons. FOSM predicts that values of Q much larger than the failure point are more probable. Also, the mode of the empirical distribution from Monte Carlo is not the value of $Q(\bar{\mathbf{x}})$, as assumed in FOSM.

To demonstrate that the type of distribution used makes significant difference in the results if FOSM, we change the distribution of κ_h to be a binomial distribution with PDF.

$$f(\kappa_\mathrm{h}) = 0.995\delta(\kappa_\mathrm{h} - 1.98582) + 0.005\delta(\kappa_\mathrm{h} - 4.82135).$$

This distribution has the same mean and standard deviation as that used previously but obviously has a much different character. As before, the parameters are joined by a normal copula. The covariance matrix overall is different and leads to an estimate of the variance of the QoI that is larger than before. Nevertheless, in this instance FOSM gives an estimated probability of failure 32 times smaller than observed empirically if Q_fail is set to 75, as shown in Fig. 8.4. This demonstrates that the accuracy of FOSM is sensitive to the underlying distributions.

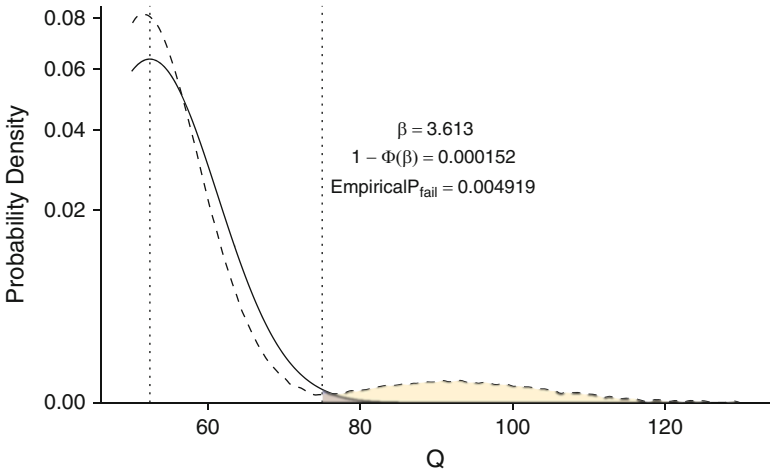

Fig. 8.4 Comparison of FOSM with a 10^6 sample Monte Carlo design to estimate the probability of failure for the ADR example where the input for κ_h is a binomial distribution with the same mean and variance as that in the previous example. Note that the vertical axis is proportional to the square root of the probability density. The solid line is the distribution fit by FOSM, and the dashed line is empirical probability density

8.2 Advanced First-Order Second-Moment Methods

One of the drawbacks to FOSM as it was formulated in the previous section is that it is independent of the underlying distributions (except through the mean and variance, which can be the same for very different distributions). Moreover, relationships between variables are not necessarily included, except in how they influence the estimate of the variance of the QoI. Advanced FOSM methods add in these effects to the estimate of the failure probability *to a degree*. These ideas are based on the Hasofer-Lind method as modified by Rackwitz and Flessler (1978).

The goal in advanced FOSM (AFOSM) is to identify the nearest point on the failure surface to the nominal value. That is, if the designed, nominal behavior of the system occurs at $\mathbf{X}_0$, we want to find the most likely point on the failure surface, called the most probable failure point. The distance from the nominal point to the failure surface will become β, and from this we estimate the probability of failure as before.

To find this failure point, we standardize the coordinate system of the inputs so that they have the same variance in "equivalent" normal values. In this new coordinate system, the nearest point on the failure surface—the set of points where $Q(\mathbf{X})$ is equal to Q_{fail}—is a distance β from the design point. This distance is β because it is the distance to the failure point in standard normal coordinates. In two dimensions the value of β defines an ellipse as

$$(\mathbf{X} - \boldsymbol{\mu})^{\mathrm{T}} \Sigma^{-1} (\mathbf{X} - \boldsymbol{\mu}) = \beta^2.$$

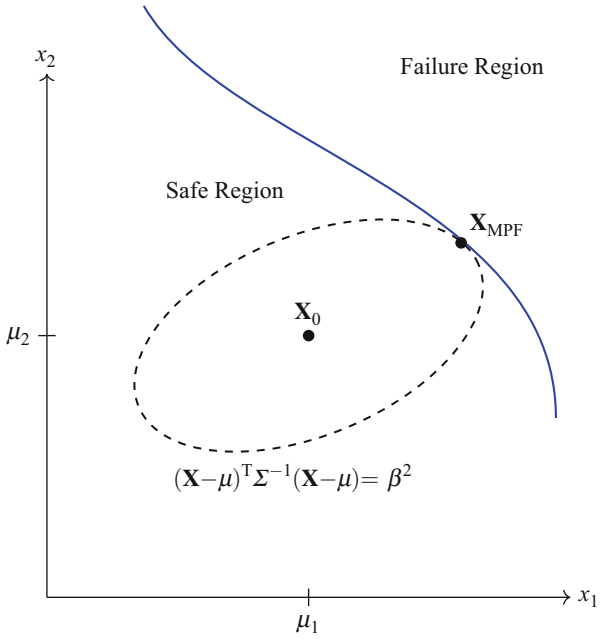

Fig. 8.5 Illustration of the advanced first-order second-moment method. The ellipse centered at the design point, $\mathbf{X}_0$, is the smallest circle in a rescaled coordinate system that touches the failure surface. The point where they touch is the most probable failure point, $\mathbf{X}_{\text{MFP}}$

This ellipse and the failure surface are illustrated in Fig. 8.5.

To use AFOSM we need to determine an equivalent normal variable for each of the inputs. This will allow us to use the standard normal distribution to estimate the probability of failure. For each variable we need to determine the mean and standard deviation for this equivalent normal. To do this we equate the distribution of each input to a normal distribution at the mean of the distribution using the CDF and PDF at some point x_i:

$$\Phi\left(\frac{x_i - \mu_i'}{\sigma_i'}\right) = F_{X_i}(x_i), \tag{8.6}$$

$$\frac{1}{\sigma_i'}\phi\left(\frac{x_i - \mu_i'}{\sigma_i'}\right) = f_{X_i}(x_i). \tag{8.7}$$

Solving these equations for μ_i' and σ_i' we get

$$\sigma_i' = \frac{\phi(\Phi^{-1}(F_{X_i}(x_i)))}{f_{X_i}(x_i)}, \tag{8.8}$$

$$\mu_i' = x_i - \Phi^{-1}(F_{X_i}(x_i))\sigma_i'. \tag{8.9}$$

In these relations we have used the fact that a random variable $X \sim \mathcal{N}(\mu, \sigma^2)$ has a PDF of

$$f(x) = \frac{1}{\sigma}\phi\left(\frac{x-\mu}{\sigma}\right).$$

It has been noted by Rackwitz and Flessler (1978) that if the original distribution is skewed, then one can match μ_i' to the median, $\mu_i' = F_{X_i}^{-1}(0.5)$, and then set the σ_i' using Eq. (8.6).

With these equivalent normal variables, we can infer a multivariate normal distribution by using the correlation matrix, $\mathbf{R}$, of the inputs. The vector of variables, $\mathbf{Y}$, defined by

$$Y_i = \frac{x_i - \mu_i'}{\sigma_i'},$$

with correlation matrix $\mathbf{R}$, will be treated as a multivariate normal with zero mean, unit variance, and known correlation.

What we want to do is find the nearest point to the failure surface $Z(\mathbf{X}) = Q_{\text{fail}} - Q(\mathbf{X}) = 0$ when measured in terms of $\mathbf{Y}(\mathbf{X})$. In other words, we want to minimize

$$\beta \equiv \min_{Z(\mathbf{X})=0} \sqrt{\mathbf{Y}^{\mathbf{T}}(\mathbf{X})\mathbf{R}^{-1}\mathbf{Y}(\mathbf{X})}. \tag{8.10}$$

Finding this minimum will give the nearest point to the failure surface, relative to the nominal system performance, where distance is measured in a normalized coordinate system. This minimum is called the most probable point of failure.

Finding this minimum will require an optimization procedure. Using Lagrange multipliers, minimizing the function

$$g(\mathbf{X}, \lambda) = \frac{1}{2}\mathbf{Y}^{\mathrm{T}}(\mathbf{X})\mathbf{R}^{-1}\mathbf{Y}(\mathbf{X}) - \lambda(Q_{\text{fail}} - Q(\mathbf{X})), \tag{8.11}$$

will find the minimum β on the failure surface.

Using this objective function, we will find a minimum using an iteration procedure. We start with a point, $\mathbf{X}$, and its mapping to the equivalent normals, $\mathbf{Y}(\mathbf{X})$. We then seek to find a point, $\hat{\mathbf{X}}$ and the associated $\hat{\mathbf{Y}} = \mathbf{Y}(\hat{\mathbf{X}})$ that is on the failure surface, with a small value of β. Therefore, we take the derivative of Eq. (8.11) with respect to $\hat{\mathbf{Y}}$ and set it to zero. After some manipulation, we get

$$\hat{\mathbf{Y}} = \lambda \mathbf{R} \nabla_{\mathbf{Y}}^{\mathrm{T}} Q, \tag{8.12}$$

where, using the chain rule, we evaluate the derivative of Q at $\mathbf{Y}$ as

$$\nabla_{\mathbf{Y}} Q(\mathbf{X}) = \left(\frac{\partial Q}{\partial Y_1}, \ldots, \frac{\partial Q}{\partial Y_p}\right) = \left(\sigma_1' \frac{\partial Q}{\partial X_1}, \ldots, \sigma_p' \frac{\partial Q}{\partial X_p}\right) \approx \nabla_{\hat{\mathbf{Y}}} Q.$$

To approximate the function, we use a first-order Taylor expansion and approximate a point on the failure surface:

$$Q(\mathbf{X}) + \nabla_{\mathbf{Y}} Q \left(\hat{\mathbf{Y}} - \mathbf{Y} \right) = Q_{\text{fail}}.$$

Using Eq. (8.12), this becomes

$$Q(\mathbf{X}) + \nabla_{\mathbf{Y}} Q \left(\lambda \mathbf{R} \nabla_{\mathbf{Y}}^{\mathrm{T}} Q - \mathbf{Y} \right) = Q_{\text{fail}}. \tag{8.13}$$

We can solve this equation for the Lagrange multiplier to get

$$\lambda = \frac{Q_{\text{fail}} - Q(\mathbf{X}) + \nabla_{\mathbf{Y}} Q \mathbf{Y}}{\nabla_{\mathbf{Y}} Q \mathbf{R} \nabla_{\mathbf{Y}}^{\mathrm{T}} Q}. \tag{8.14}$$

Therefore, using Eq. (8.12), the approximate value of β is

$$\beta = \frac{Q_{\text{fail}} - Q(\mathbf{X}) + \nabla_{\mathbf{Y}} Q \mathbf{Y}}{\sqrt{\nabla_{\mathbf{Y}} Q \mathbf{R} \nabla_{\mathbf{Y}}^{\mathrm{T}} Q}}. \tag{8.15}$$

This result leads to the iteration procedure for determining the β shown in Algorithm 8.1. Each iteration of the algorithm requires the calculation of the value of the QoI at a point and the local derivatives at that point. Therefore, it will require $p + 1$ QoI evaluations per iteration. For this reason a good initial guess for the most probable failure point is in order, if possible.

Algorithm 8.1 Algorithm for finding β and most probable failure point using AFOSM

0. Begin with an initial value for the most probable failure point, $\mathbf{X}_0$, and set $\ell = 0$.
1. Determine σ_i', μ_i' using the value of $\mathbf{X}_\ell$. Compute $\mathbf{Y}_\ell$.
2. Compute the derivatives of $Q(\mathbf{X})$ at point $\mathbf{X}_\ell$ to form $\nabla_{\mathbf{Y}} Q$.
3. Evaluate λ using the formula

$$\lambda = \frac{Q_{\text{fail}} - Q(\mathbf{X}_\ell) + \nabla_{\mathbf{Y}_\ell} Q \mathbf{Y}_\ell}{\nabla_{\mathbf{Y}_\ell} Q \mathbf{R} \nabla_{\mathbf{Y}_\ell}^{\mathrm{T}} Q}.$$

4. Compute $\mathbf{Y}_{\ell+1} = \lambda \mathbf{R} \nabla_{\mathbf{Y}_\ell}^{\mathrm{T}} Q$, and $\beta_{\ell+1} = \sqrt{\mathbf{Y}_{\ell+1}^{\mathbf{T}} \mathbf{R}^{-1} \mathbf{Y}_{\ell+1}}$
5. Check for convergence, i.e., is $|\beta_{\ell+1} - \beta_\ell| < \delta$, and $|Q(\mathbf{X}_{\ell+1}) - Q_{\text{fail}}| < \epsilon$.
6. If not converged, set $\ell \to \ell + 1$ and go to step 1.

As a demonstration of the AFOSM method, we will apply it to the QoI

$$Q(\mathbf{X}) = 2x_1^3 + 10x_1x_2 + x_1 + 3x_2^3 + x_2,$$

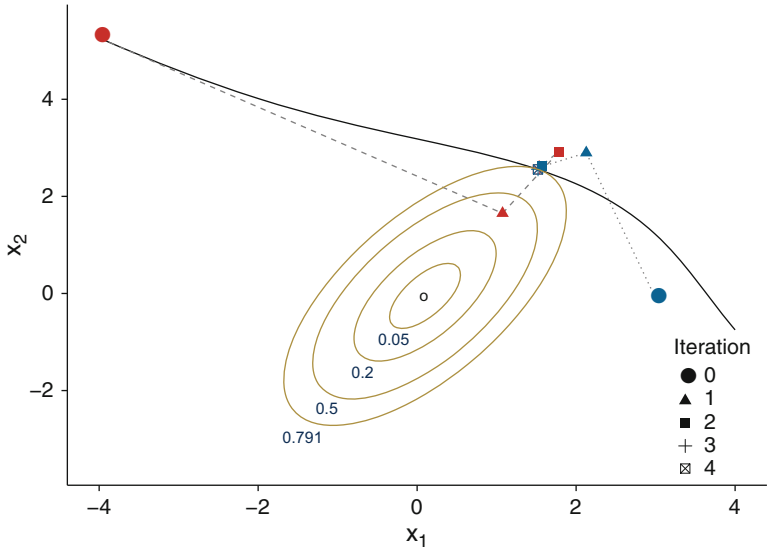

Fig. 8.6 Plot of the convergence of the AFOSM method to the most probable point of failure for several starting points. The solid curve is the failure surface, below which $Q(\mathbf{X}) < Q_{\text{fail}}$. Ellipses of different of magnitudes of $\mathbf{Y}^{\mathbf{T}}(\mathbf{X})\mathbf{R}^{-1}\mathbf{Y}(\mathbf{X})$ are drawn to show that the most probable failure point touches such an ellipse. AFOSM computes $\beta^2 \approx 0.791$

where the input is a multivariate normal with mean vector $(0.1, -0.05)$; the covariance and correlation matrices are

$$\Sigma = \begin{pmatrix} 4 & 3.9 \\ 3.9 & 9 \end{pmatrix}, \qquad \mathbf{R} = \begin{pmatrix} 1 & 0.65 \\ 0.65 & 1 \end{pmatrix},$$

which implies $\sigma_1 = 2$, and $\sigma_2 = 3$. Because the distributions are normal, the value of σ_i' and μ_i' are constant over the iterations. The value of Q_{fail} used in this example is 100. The value of the gradient for this problem is

$$\nabla_{\mathbf{X}} Q = \left(6x_1^2 + 10x_2 + 1, 10x_1 + 9x_2^2 + 1\right),$$

and

$$\nabla_{\mathbf{Y}} Q = \left(12x_1^2 + 20x_2 + 2, 30x_1 + 27x_2^2 + 3\right).$$

The results from applying AFOSM to this problem are shown in Fig. 8.6. The results for two different starting points are shown. In these results we can see that during the iteration procedure, the iterations do not necessarily stay on the failure surface if it starts there (see iterations that start at the top left). This is due to the

linear approximation of $Q(\mathbf{X})$. In the results we see that the value of β computed by the method is indeed the value of $\beta^2 = \mathbf{Y}^{\mathbf{T}}(\mathbf{X})\mathbf{R}^{-1}\mathbf{Y}(\mathbf{X})$ such that the ellipse touches the failure surface.

For this problem, using $\beta = 0.889$, the estimated failure probability is $1 - \Phi(\beta) = 0.187$. A 10^6 Monte Carlo sample gives 0.202 for the failure probability, a relative difference of about 7%. Using basic FOSM, we get that the probability of failure is functionally zero, because the estimate of β from Eq. (8.5) is 14.6. For this problem, AFOSM is necessary to get a reasonable approximation of the failure rate because the nominal design point $\mathbf{X} = (0.1, 0 - 0.5)$ has a value much smaller than Q_{fail}. In such a problem, it is necessary to include the interactions and nonlinearities in the QoI to get a good estimate of the behavior of the probability of failure.

To demonstrate that AFOSM will work when the input distributions are not normal, we change the problem to have x_1 and x_2 distributed by independent, i.e., $\mathbf{R} = \mathbf{I}$, Gumbel distributions (see Eq. (7.10) for the PDF of this distribution) with the same mean and standard deviation as that used in the previous example. Because the Gumbel distribution is skewed, we use the median of the distributions for the value of μ_i' for the equivalent normal distributions and then evaluate Eq. (8.6) to get σ_i'. The value of σ_i' will change each iteration in this case, resulting in a change in the formula for $\nabla_{\mathbf{Y}} Q$ each iteration. The results from applying AFOSM to this problem are shown in Fig. 8.7. Notice that when the underlying distribution is nonnormal, the surfaces of equal β^2 are no longer ellipses. Additionally, the mean of the inputs is no longer located at $\beta = 0$. As before, despite different starting points, the method converges to the same point, with a value of $\beta = 1.125$. For this value of β, we infer a failure probability of 0.130, compared with 10^6 Monte Carlo samples giving 0.169. For this problem, as before, the value of β from FOSM is much too large at 14.4. Therefore, while the approximation of AFOSM is not perfect (it underestimated the failure probability), it is much better than extrapolating from $Q(\bar{\mathbf{X}})$ using the gradient.

This example demonstrates that AFOSM can be a large improvement over basic FOSM. The improvement comes at a price in terms of more function evaluations. While FOSM requires only $N + 1$ evaluations of the QoI, AFOSM requires $N + 1$ function evaluations per iteration. Despite this increase AFOSM is still relatively inexpensive relative to sampling-based methods. Nevertheless, it is still an approximate method, and forgetting that AFOSM is based on normal distributions is a statistical Pelagian error.[1]

[1]The founding assumption of normal distributions could be called the "original sin" of AFOSM. The Pelagian heresy, named for the fourth-century British theologian Pelagius, rejected the notion of original sin, among other things. Therefore, ignoring the ramifications of the normal assumption could be seen as forgetting about this original sin.

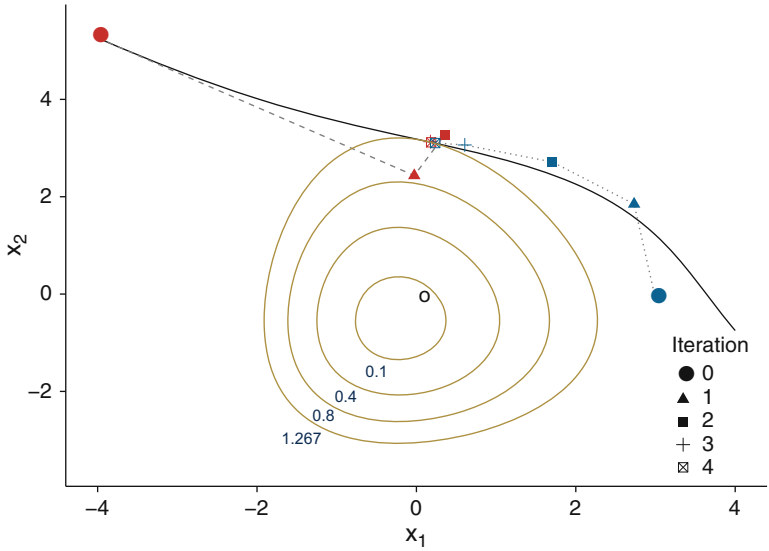

Fig. 8.7 Plot of the convergence of the AFOSM method to the most probable point of failure for several starting points where the distribution of x_1 and x_2 are independent Gumbel distributions with the same mean and standard deviations as the previous example. The solid curve is the failure surface, below which $Q(\mathbf{X}) < Q_{\text{fail}}$. Surfaces of different of magnitudes of $\mathbf{Y}^{\mathbf{T}}(\mathbf{X})\mathbf{Y}(\mathbf{X})$ are drawn to show that the most probable failure point touches such a surface. AFOSM computes $\beta^2 \approx 1.267$. The black circle indicates the mean of the inputs

8.3 Higher-Order Approaches

It is possible to use an estimate of the second derivative of the QoI to improve on the reliability methods we have discussed so far. These estimates can be improvements over those from the methods we have already discussed. The estimate of second derivatives (and the cross-derivative terms) in problems with a modest number of inputs can be cost prohibitive (as we saw in Chap. 4). Therefore, rather than discuss higher-order reliability methods, we will use more general approximations to the QoI in the form of polynomial chaos expansions in the next chapter.

8.4 Notes and References

Many of the topics in this section are presented in the book on reliability analysis by Haldar and Mahadevan (2000). Also, the review by Bastidas-Arteaga and Soubra (2006) is a useful reference. The references in these two works are useful because much of the reliability analysis literature is contained in domain-specific publications.

8.5 Exercises

1. Repeat the example in Sect. 8.2 where the distributions of x_1 and x_2 are Gumbel distributions with the same mean and standard deviation used previously. Use a Frank Copula with $\theta = 0, 1, 5, 10, 20$ to join the input parameter distributions. How does the most probable failure point change with the changes in the copula?
2. Consider the Rosenbrock function: $f(x, y) = (1-x)^2 + 100(y-x^2)^2$. Assume that $x = 2t - 1$, where $T \sim \mathscr{B}(3, 2)$ and $y = 2s - 1$, where $S \sim \mathscr{B}(1.1, 2)$. Estimate β and the probability that $f(x, y)$ is less than 10 using

   (a) FOSM
   (b) Advanced FOSM

3. Using a discretization of your choice, solve the equation

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = D\frac{\partial^2 u}{\partial x^2} - \omega u,$$

   for $u(x, t)$ on the spatial domain $x \in [0, 10]$ with periodic boundary conditions $u(0^-) = u(10^+)$ and initial conditions

$$u(x, 0) = \begin{cases} 1 & x \in [0, 2.5] \\ 0 & \text{otherwise} \end{cases}.$$

   Use the solution to compute the total reactions

$$\int_5^6 dx \int_0^5 dt\, \omega u(x, t).$$

   Compute the probability that this quantity of interest is greater than 0.035 using FOSM and AFOSM using the following distributions:

   (a) $\mu_v = 0.5, \sigma_v = 0.1$,
   (b) $\mu_D = 0.125, \sigma_D = 0.03$,
   (c) $\mu_\omega = 0.1, \sigma_\omega = 0.05$,

   How do these results change with changes in Δx and Δt?

Chapter 9
Stochastic Projection and Collocation

This is gonna be a total cluster cuss for everybody.

—from the film *Fantastic Mr. Fox*

An alternative approach to the sampling and reliability approaches previously discussed, and the one that we will consider in some detail here is to write the quantity of interest as an expansion in orthogonal polynomials. In particular we will pick the orthogonal polynomials so that the weighting function in the orthogonality condition "matches" the distribution of the parameters. Table 9.1 shows four common distributions and the matching orthogonal polynomial for each. To compute the integrals in the expansion, we will use a collocation procedure and Gauss quadrature. In the process we will encounter many classic approximation techniques and have to review a host of statistics, special functions, and quadrature techniques.

This approach is known as stochastic spectral projection, but the expansions we use are called polynomial chaos expansions. The name spectral is applied because if the function being approximated is smooth, the error in the expansion decays exponentially in the number of expansion coefficients. Therefore, if the quantity of interest is a smooth function of the random variables, then we expect the expansion to be accurate with only a few terms. A benefit of spectral projection is that, like Monte Carlo, it is a nonintrusive method: existing codes and methods can be applied out of the box. The approach does suffer from the curse of dimensionality in that the number of terms in the expansion explodes as the dimension of the random variable space increases. Later we will discuss approaches to mitigate this, using sparse grids and compressed sensing techniques.

The first part of this chapter (Sects. 9.1–9.7) deals with these methods for quantities of interest. Then in Sect. 9.8 we discuss how these ideas can be applied to random processes. We begin with a discussion of applying projection techniques to

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-319-99525-0_9) contains supplementary material, which is available to authorized users.

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0_9

Table 9.1 The orthogonal polynomials and support corresponding to the different families of input random variables

Input distribution	Orthogonal polynomial	Support
Normal	Hermite	$(-\infty, \infty)$
Uniform	Legendre	$[a, b]$
Beta	Jacobi	$[a, b]$
Gamma	Laguerre	$[0, \infty)$

single random variables, before embarking on multivariate expansions and the ideas of sparse quadrature. A natural starting point is a QoI that is a function of single, standard normally distributed random variable.

The quote used at the beginning of the chapter is related to the way many students and instructors find this subject. For the students much of the notation and the various competing definitions for basis functions make the application of these methods precarious. For the instructor the task of giving students adequate coverage of the topic is difficult without spending large amounts of time defining special functions and quadrature rules and writing out multidimensional expansions. This chapter seeks to give adequately explained and detailed projection techniques with fully worked examples to make the topic readily digestible and applicable to real-world problems.

9.1 Hermite Expansions for Normally Distributed Parameters

The Hermite polynomials,[1] $He_n(x)$, are a set of orthogonal polynomials that form a basis for square-integrable functions on the real line with weight,

$$w(x) = e^{-x^2/2},$$

and inner product

$$\langle g(x), h(x) \rangle = \int_{-\infty}^{\infty} g(x)h(x)\, e^{-\frac{x^2}{2}}\, \mathrm{d}x,$$

i.e., the polynomials form an orthogonal basis for $L^2(\mathbb{R}, w(x)\, dx)$. The Hermite polynomials are defined as

$$He_n(x) = (-1)^n e^{\frac{x^2}{2}} \frac{d^n}{dx^n} e^{-\frac{x^2}{2}}. \tag{9.1}$$

[1] There are two definitions of the Hermite polynomials that are scalings of each other. We use the "probabilist" version of the functions because of similarities with the standard normal distribution in the weighting function. The "physicist" version of the polynomials is slightly different and forms a natural expression of the quantum harmonic oscillator.

The first few Hermite polynomials are

$$\begin{aligned} He_0(x) &= 1, \\ He_1(x) &= x, \\ He_2(x) &= x^2 - 1, \\ He_3(x) &= x^3 - 3x, \\ He_4(x) &= x^4 - 6x^2 + 3, \\ He_5(x) &= x^5 - 10x^3 + 15x. \end{aligned}$$

The orthogonality relation for the Hermite polynomials is

$$\int_{-\infty}^{\infty} He_m(x) He_n(x)\, e^{-\frac{x^2}{2}}\, \mathrm{d}x = \sqrt{2\pi}\, n! \delta_{nm}. \tag{9.2}$$

The expansion of a function in terms of Hermite polynomials is written as

$$g(x) = \sum_{n=0}^{\infty} c_n He_n(x), \tag{9.3}$$

where the expansion constants are given by

$$c_n = \frac{\langle g(x), He_n(x) \rangle}{\sqrt{2\pi}\, n!}. \tag{9.4}$$

9.1.1 Hermite Expansion of a Function of a Standard Normal Random Variable

Consider a function $g(x)$ where $x \sim \mathcal{N}(0, 1)$. The value of the function is also a random variable that we will call $G \sim g(x)$. If we compute the zeroth order constant in the Hermite expansion of this function, we get

$$c_0 = \int_{-\infty}^{\infty} \frac{g(x)}{\sqrt{2\pi}}\, e^{-\frac{x^2}{2}}\, dx = E[G] = \bar{g}. \tag{9.5}$$

In other words, the constant c_0 in the expansion is the mean of the random variable G.

Recall that the variance of G is given by $E[G^2] - E[G]^2$, which is equal to

$$\begin{aligned}\text{Var}(G) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \left(\sum_{n=0}^{\infty} c_n He_n(x) \right)^2 e^{-\frac{x^2}{2}} \, dx - c_0^2 \\ &= \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{\infty} c_n^2 \langle He_n(x), He_n(x) \rangle - c_0^2 \\ &= \sum_{n=1}^{\infty} n! c_n^2.\end{aligned} \tag{9.6}$$

Here we have used the orthogonality of the Hermite polynomials to get the second equation, followed by the value of the integral in Eq. (9.2) to get the final result.

As an example, let us consider the function $g(x) = \cos(x)$. In this case we can directly compute the expansion coefficients:

$$c_n = \frac{1}{\sqrt{2\pi} n!} \int_{-\infty}^{\infty} \cos(x) He_n(x) e^{-x^2/2} \, dx = \begin{cases} 0 & n \text{ odd} \\ (-1)^{\frac{n}{2}} \frac{e^{-1/2}}{n!} & n \text{ even} \end{cases}. \tag{9.7}$$

This makes the approximation to the function

$$\cos(x) = e^{-\frac{1}{2}} \sum_{n \text{ even}} (-1)^{\frac{n}{2}} \frac{He_n(x)}{n!}, \qquad x \sim \mathcal{N}(0, 1). \tag{9.8}$$

This implies that the mean of $g(x)$ is $e^{-1/2}$ and that the variance is

$$\text{Var}(G) = e^{-1} \sum_{n \text{ even}, \, n > 1} \frac{1}{n!} = e^{-1}(\cosh(1) - 1) \approx 0.19978820.$$

We can get a baseline for comparison between the expansion and the actual distribution of G by sampling a value for x from a standard normal and then evaluating $g(x)$. The resulting distribution is a Monte Carlo approximation to the true distribution of G. We then can compare that to the values obtained by sampling x and then evaluating the expansion in Eq. (9.8) with different orders of expansion.[2] These results are shown in Fig. 9.1.

In these results we see the improvement obtained as we go to higher order expansions. The zeroth-order expansion only gives a value of the mean, and there is a large improvement in going to the second-order expansion. There is a noticeable

[2] For a function that is expensive to evaluate, we may not be able to estimate the distribution using Monte Carlo. Nevertheless, sampling x and then evaluating a polynomial of x is basically free.

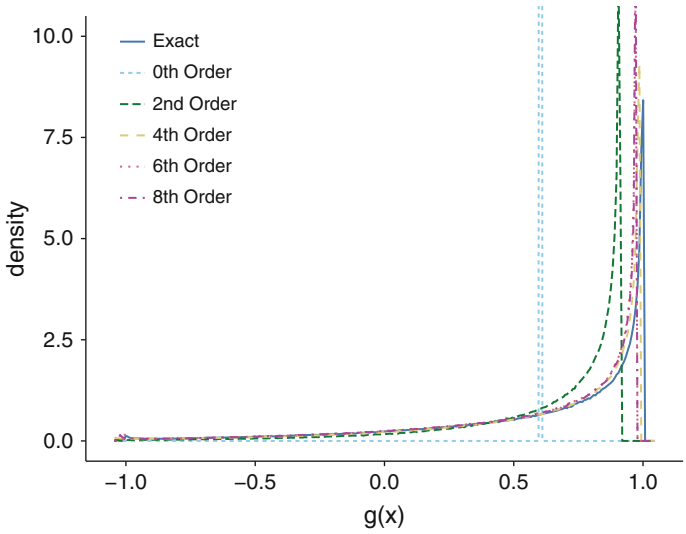

Fig. 9.1 PDF of the random variable $g(x) = \cos(x)$, where $x \sim \mathcal{N}(0, 1)$, and various approximations. This figure was generated from 10^6 samples of x that were used to evaluate $g(x)$ and the various approximations

Table 9.2 The convergence of $\mathrm{Var}(G)$ for $g(x) = \cos(x)$, where $x \sim \mathcal{N}(0, 1)$

Order	Variance
0	0
2	0.183939721
4	0.199268031
6	0.199778974
8	0.199788098
∞	0.199788200

difference between fourth- and second-order expansions, though beyond that, there is little difference in the figure. We can track improvement in the higher-order expansions by looking at the convergence of the variance. In Table 9.2 we show that adding more terms to the expansion does improve the estimate of the variance, though modestly beyond the second-order expansion. The values in this table were computed using `Mathematica`.

9.1.2 *Hermite Expansion of a Function of a General Normal Random Variable*

If the random variable is normal, but not standard normal, then we need to change the procedure a bit. We say that $g(x)$ is a function of the random variable $x \sim \mathcal{N}(\mu, \sigma^2)$. In this case we will change variables to express the function as $g(Z)$

where Z is a standard normal random variable. When Z is a standardized version of x, we can relate the expectation of a function of x to a function of Z as

$$E[g(x)] = E[g(\mu + \sigma Z)]. \tag{9.9}$$

We can check this in the formula for the mean

$$E[x] = E[\mu + \sigma Z] = \mu + \sigma E[Z].$$

Therefore, in this case

$$c_n = \frac{\langle g(\mu + \sigma z), He_n(z) \rangle}{\sqrt{2\pi} n!}. \tag{9.10}$$

The bounds of the inner product's integration are not affected because they are infinite; this may not be the case when we have bounded random variables.

Going back to our example from before where $g(x) = \cos(x)$, we now say that $x \sim \mathcal{N}(\mu = 0.5, \sigma^2 = 4)$. Evaluating the integrals for the coefficients in Eq. (9.10) gives the following expansion, to fifth order,

$$\begin{aligned} \cos(x) \approx\ & e^{-2} \left(1 - 2He_2(z) + \frac{2}{3} He_4(z)\right) \cos\left(\frac{1}{2}\right) \\ & + e^{-2} \left(2He_1(z) + \frac{4}{3} He_3(z) - \frac{4}{15} He_5(z)\right) \sin\left(\frac{1}{2}\right) \end{aligned} \tag{9.11}$$

The mean is

$$\bar{g} = e^{-2} \cos\left(\frac{1}{2}\right) \approx 0.1187678845769458,$$

and the variance is

$$\mathrm{Var}(G) = \frac{\left(e^4 - 1\right)\left(e^4 - \cos(1)\right)}{2e^8} \approx 0.48598481520881144144.$$

The distributions produced by various approximations to G are shown in Fig. 9.2. The exact distribution is more difficult to capture with a polynomial expansion, partly because of the non-smoothness in the solution at ± 1. By the sixth-order expansion, the overall shape of the distribution is correct, though the peaks are not in the correct place. Note that in all of these curves, the mean is the same. Also, notice that even though the minimum value of $g(x)$ is -1, the expansion can give a nonzero probability of getting a value less than -1.

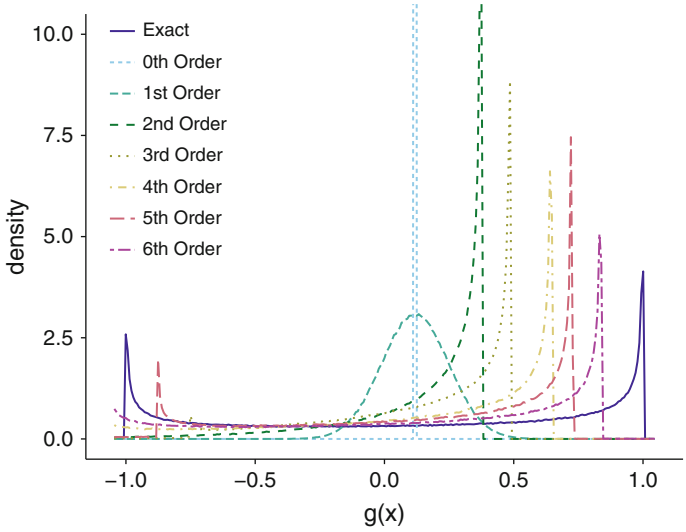

Fig. 9.2 PDF of the random variable $g(x) = \cos(x)$, where $x \sim \mathcal{N}(\mu = 0.5, \sigma^2 = 4)$, and various approximations. This figure was generated from 10^6 samples of x that were used to evaluate $g(x)$ and the various approximations

Table 9.3 The convergence of $\mathrm{Var}(G)$ for $g(x) = \cos(x)$, where $x \sim \mathcal{N}(\mu = 0.5, \sigma^2 = 4)$

Order	Variance
0	0
1	0.016807404
2	0.128990805
3	0.173419006
4	0.329091747
5	0.380416942
6	0.458346473
∞	0.485984815

In this example the variance also takes longer to converge. In Table 9.3, we see that even the sixth-order expansion only has 1 digit correct.

9.1.3 Gauss-Hermite Quadrature

Recall that our ultimate goal is to use polynomial expansions to provide information about the distribution of output quantities from a computer simulation. To that end we will need to estimate the coefficients in the Hermite expansion. If we use a quadrature rule to estimate the integrals in these coefficients, then we would like a quadrature rule to require as few evaluations of the integrand as possible, because each evaluation requires running a new simulation at a different point in input space.

Table 9.4 The non-negative abscissas and weights for Gauss-Hermite quadrature up to order 6

n	$\lvert x_i \rvert$	w_i
1	0	$\sqrt{\pi}$
2	$\frac{1}{\sqrt{2}}$	$\frac{1}{2}\sqrt{\pi}$
3	0	$\frac{2}{3}\sqrt{\pi}$
	$\frac{1}{2}\sqrt{6}$	$\frac{1}{6}\sqrt{\pi}$
4	0.524647623275290	0.804914090005514
	1.65060123885785	0.0813552017779922
5	0	0.945308720482942
	0.958572464613819	0.3936193231522404
	2.02018270456086	0.01995326880748209
6	0.436077411927617	0.7246295952243919
	1.335849074013697	0.1570673203228565
	2.350604973674492	0.004530009905508858

The most common way to approximate the required integrals is to use Gauss-Hermite quadrature, which is a Gauss quadrature rule for computing integrals of the form

$$\int_{-\infty}^{\infty} f(x)e^{-x^2}\,dx \approx \sum_{i=1}^{n} w_i f(x_i), \tag{9.12}$$

where the abscissas, x_i, are given by the n roots of $He_n(x)$, and the weights are given by

$$w_i = \frac{\sqrt{\pi}n!}{n^2\left(He_{n-1}\left(\sqrt{2}x_i\right)\right)^2}. \tag{9.13}$$

Gauss quadratures are defined so that the maximum degree polynomial is exactly integrated given a specified number of function evaluations. In particular, a Gauss quadrature rule using n points will exactly integrate a polynomial of degree $2n-1$. That this is possible can be seen by noting that a polynomial of degree $2n-1$ has $2n$ coefficients and an n point quadrature rule has $2n$ degrees of freedom: n points and n weights. To determine the quadrature points and weights, one can use a variety of computational techniques, such as the Golub and Welsch algorithm. See Townsend (2015) for an interesting discussion of the history of algorithms for computing the quadrature rules.

The values for the weights and abscissas given up to $n = 6$ are given in Table 9.4. Note that the points are symmetric about the origin, so we only give the magnitude of the abscissas.

This quadrature set has the standard features of Gauss quadrature. The rule will be exact when $f(x)$ is a polynomial of degree $2n-1$ or less.

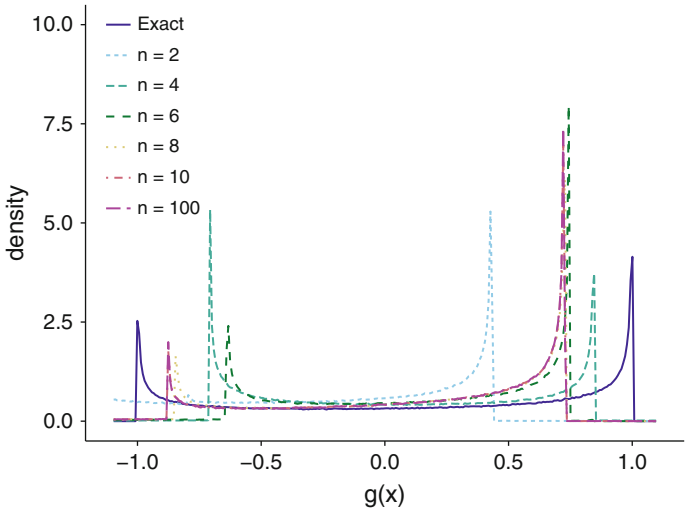

Fig. 9.3 PDF of the random variable $g(x) = \cos(x)$, where $x \sim \mathcal{N}(\mu = 0.5, \sigma^2 = 4)$ using a fifth-order Hermite expansion with various Gauss-Hermite quadrature rules to approximate the coefficients. This figure was generated from 10^5 samples of x that were used to evaluate $g(x)$ and the various approximations

There is a slight issue in Gauss-Hermite quadrature in that it uses the a weight function of $\exp\left(-x^2\right)$, rather than $\exp\left(-x^2/2\right)$ that we used in our inner product definition.[3] Therefore, we need to make the change of variable $x \to x'/\sqrt{2}$. This makes the approximation to the inner product

$$\langle g(x), He_m(x) \rangle \approx \sqrt{2} \sum_{i=1}^{n} w_i g(\sqrt{2} x_i). \tag{9.14}$$

We can use our previous example of $g(x) = \cos(x)$, where $x \sim \mathcal{N}(\mu = 0.5, \sigma^2 = 4)$, as a test of estimating the inner products using Gauss-Hermite quadrature rules. In Fig. 9.3, the distribution, as approximated by a fifth-order Hermite expansion, is computed using Gauss-Hermite quadratures of different values of n. For this expansion, we need at least eight quadrature points to get an accurate estimate of the coefficients. We can see the convergence in the coefficients with the number of quadrature points in Table 9.5. Here we see that to estimate the mean, c_0, with two digits of accuracy, we need $n = 6$, whereas the c_5 term needs $n = 9$ to get that many digits of accuracy.

[3] We could have defined our Gaussian quadrature rule to have the same weight function as we used in our expansion. Nevertheless, most readily accessible tabulations of Hermite quadrature use the weighting function used herein.

Table 9.5 The convergence of the first six coefficients in the Hermite polynomial expansion $g(x) = \cos(x)$, where $x \sim \mathcal{N}(\mu = 0.5, \sigma^2 = 4)$ as estimated by different Gauss-Hermite quadrature rules

n	c_0	c_1	c_2	c_3	c_4	c_5
2	−0.365203	−0.435940	−0.000000	0.145313	0.030434	−0.021797
3	0.307609	0.087730	−0.569973	−0.000000	0.142493	−0.004386
4	0.065646	−0.219271	−0.023343	0.173281	0.000000	−0.034656
5	0.130446	−0.103803	−0.322800	0.037629	0.141446	0.000000
6	0.116662	−0.135589	−0.213171	0.104748	0.048382	−0.028531
7	0.119090	−0.128702	−0.242956	0.081489	0.089843	−0.012370
8	0.118725	−0.129931	−0.236549	0.087602	0.076377	−0.018886
9	0.118773	−0.129744	−0.237688	0.086315	0.079768	−0.016907
10	0.118767	−0.129769	−0.237515	0.086541	0.079075	−0.017382
100	0.118768	−0.129766	−0.237536	0.086511	0.079179	−0.017302

9.2 Generalized Polynomial Chaos

When the input parameter is not normally distributed, we need a different polynomial expansion to approximate the mapping from input parameter to output random variable. We will cover three such cases, as enumerated in Table 9.1. First, we tackle uniform random variables.

9.2.1 Uniform Random Variables: Legendre Polynomials

Consider a random variable x that is uniformly distributed in the range $[a, b]$. In this case we write $x \sim \mathcal{U}[a, b]$.[4] Additionally, the PDF of x is

$$f(x|a, b) = \begin{cases} \frac{1}{b-a} & x \in [a, b] \\ 0 & \text{otherwise} \end{cases}. \tag{9.15}$$

The mean of a uniform distribution is $(a + b)/2$, and the variance is $(b - a)^2/12$.

As with normal random variables, it is useful to convert general uniform random variables to a standardized random variable.[5] In this case, we map the interval $[a, b]$ to $[-1, 1]$ to correspond with the support with the standard definition of Legendre polynomials. In particular, if $Z \sim \mathcal{U}[-1, 1]$, then

$$x = \frac{b-a}{2} z + \frac{a+b}{2}, \tag{9.16}$$

[4] $\mathcal{U}(a, b)$ denotes a uniform distribution between a and b.

[5] For statisticians it is more common to think of a standard uniform random variable as having the range $[0, 1]$. However, defining the standard to be symmetric about the origin makes for easier algebra down the road. This will also be the case with beta-distributed random variables later.

Table 9.6 The first ten Legendre polynomials

n	$P_n(x)$
0	1
1	x
2	$\frac{1}{2}(3x^2-1)$
3	$\frac{1}{2}(5x^3-3x \quad)$
4	$\frac{1}{8}(35x^4-30x^2+3)$
5	$\frac{1}{8}(63x^5-70x^3+15x)$
6	$\frac{1}{16}(231x^6-315x^4+105x^2-5)$
7	$\frac{1}{16}(429x^7-693x^5+315x^3-35x)$
8	$\frac{1}{128}(6435x^8-12012x^6+6930x^4-1260x^2+35)$
9	$\frac{1}{128}(12155x^9-25740x^7+18018x^5-4620x^3+315x)$
10	$\frac{1}{256}(46189x^{10}-109395x^8+90090x^6-30030x^4+3465x^2-63)$

and

$$z = \frac{a+b-2x}{a-b}. \tag{9.17}$$

Therefore, the expectation operator on a uniform random variable transforms to

$$E[g(x)] = \frac{1}{b-a}\int_a^b g(x)\,dx = \frac{1}{2}\int_{-1}^{1} g\left(\frac{b-a}{2}z + \frac{a+b}{2}\right)dz. \tag{9.18}$$

For a function on the range $[-1, 1]$, the Legendre polynomials form an orthogonal basis. The Legendre polynomials are defined as

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}\left[(x^2-1)^n\right]. \tag{9.19}$$

The first ten of these polynomials are given in Table 9.6.

The orthogonality relation for Legendre polynomials is written as

$$\int_{-1}^{1} P_n(x)P_{n'}(x)\,dx = \frac{2}{2n+1}\delta_{nn'}. \tag{9.20}$$

The expansion of a square-integrable function on the interval $[a, b]$ in Legendre polynomials is then

$$g(x) = \sum_{n=0}^{\infty} c_n P_n \left(\frac{a+b-2x}{a-b} \right), \qquad x \in [a,b], \tag{9.21}$$

where c_n is defined by

$$c_n = \frac{2n+1}{2} \int_{-1}^{1} g\left(\frac{b-a}{2} z + \frac{a+b}{2} \right) P_n(z)\, dz. \tag{9.22}$$

As before, c_0 will be the mean of the random variable $G \sim g(x)$:

$$\begin{aligned} c_0 &= \frac{1}{2} \int_{-1}^{1} g\left(\frac{b-a}{2} z + \frac{a+b}{2} \right) dz \\ &= \frac{1}{b-a} \int_{a}^{b} g\,(x)\; dx \\ &= E[G]. \end{aligned} \tag{9.23}$$

Additionally, the variance of the G is equivalent to the sum of the squares of the coefficients with $n \geq 1$:

$$\begin{aligned} \mathrm{Var}(G) &= \frac{1}{2} \int_{-1}^{1} \left(\sum_{n=0}^{\infty} c_n P_n\,(z) \right)^2 dz - c_0^2 \\ &= \sum_{n=1}^{\infty} \frac{c_n^2}{2n+1}. \end{aligned} \tag{9.24}$$

As a demonstration of the Legendre expansion, we will once again turn to the function $g(x) = \cos(x)$. This time, however, $x \sim \mathscr{U}(0, 2\pi)$. In this case we get

$$c_n = \frac{2n+1}{2} \int_{-1}^{1} \cos\left(\pi z + \pi\right) P_n(z)\, dz = -\frac{2n+1}{2} \int_{-1}^{1} \cos\left(\pi z\right) P_n(z)\, dz. \tag{9.25}$$

This makes the expansion, through sixth-order

$$\begin{aligned} \cos(x) \;\approx\; & \frac{15}{\pi^2} P_2\,(z) + \frac{45\left(4\pi^2 - 42\right)}{2\pi^4} P_4(z) \\ & + \frac{273\left(7920 - 960\pi^2 + 16\pi^4\right)}{16\pi^6} P_6(z) \quad x \sim \mathscr{U}(0, 2\pi), \end{aligned} \tag{9.26}$$

Table 9.7 The convergence of Var(G) for $g(x) = \cos(x)$, where $x \sim \mathcal{U}(0, 2\pi)$

Order	Variance
0	0
2	0.461969
4	0.499663
6	0.499999
8	0.500000
∞	0.500000

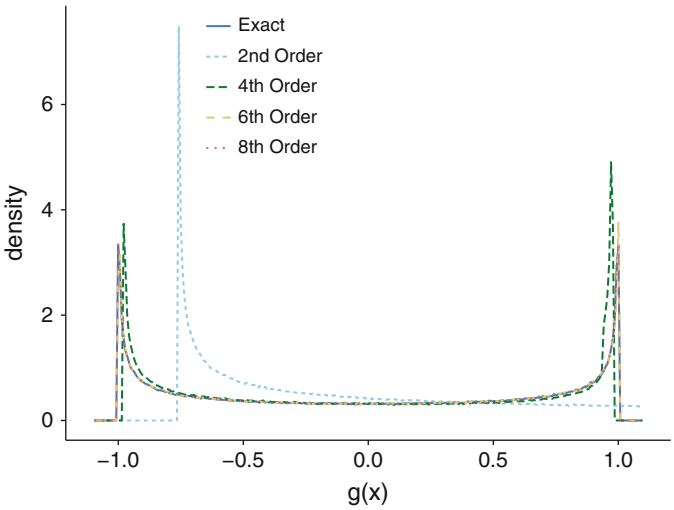

Fig. 9.4 PDF of the random variable $g(x) = \cos(x)$, where $x \sim \mathcal{U}(0, 2\pi)$, and various approximations. This figure was generated from 10^6 samples of x that were used to evaluate $g(x)$ and the various approximations

and z is related to x via Eq. (9.17). The variance of this function is given by

$$\text{Var}(G) = \frac{1}{2\pi} \int_0^{2\pi} \cos^2(x)\, dx = \frac{1}{2}. \tag{9.27}$$

The convergence of the variance estimate is given in Table 9.7.

The convergence of the approximation to G as a function of the order of the Legendre expansion is shown in Fig. 9.4. In this case, the approximation converges rather quickly: the eighth-order expansion is indistinguishable from the exact distribution.

9.2.2 Gauss-Legendre Quadrature

For estimating the coefficients in a Legendre expansion, Gauss-Legendre quadrature is a natural choice. Gauss-Legendre quadrature approximately integrates functions on the range $[-1, 1]$ as

$$\int_{-1}^{1} f(z)\, dz \approx \sum_{i=1}^{n} w_i f(z_i), \tag{9.28}$$

where the z_i are the roots of P_n, and the weights are given by

$$w_i = \frac{2}{\left(1 - z_i^2\right)\left[P_n'(z_i)\right]^2}. \tag{9.29}$$

Gauss-Legendre quadrature integrates polynomials of degree $2n - 1$ exactly (Table 9.8).

We can use our previous example of $g(x) = \cos(x)$, where $x \sim \mathcal{U}(0, 2\pi)$, as a test of estimating the inner products using Gauss-Legendre quadrature rules. In Fig. 9.5, the distribution, as approximated by a fifth-order Legendre expansion, is computed using Gauss-Legendre quadratures of different values of n. We can see the convergence in the coefficients with the number of quadrature points in Table 9.9. Here we see that to estimate the mean, c_0, with two digits of accuracy, we need $n = 4$, whereas the c_4 term needs $n = 7$ to get that many digits of accuracy.

Table 9.8 The non-negative abscissas and weights for Gauss-Legendre quadrature up to order 6

n	$\lvert x_i \rvert$	w_i
1	0	2
2	$\frac{1}{\sqrt{3}}$	1
3	0	$\frac{8}{9}$
	$\sqrt{\frac{3}{5}}$	$\frac{5}{9}$
4	0.3399810436	0.652145155
	0.8611363116	0.347854845
5	0	0.568888889
	0.5384693101	0.47862867
	0.9061798459	0.2369268851
6	0.2386191860	0.467913935
	0.6612093865	0.360761573
	0.9324695142	0.171324492

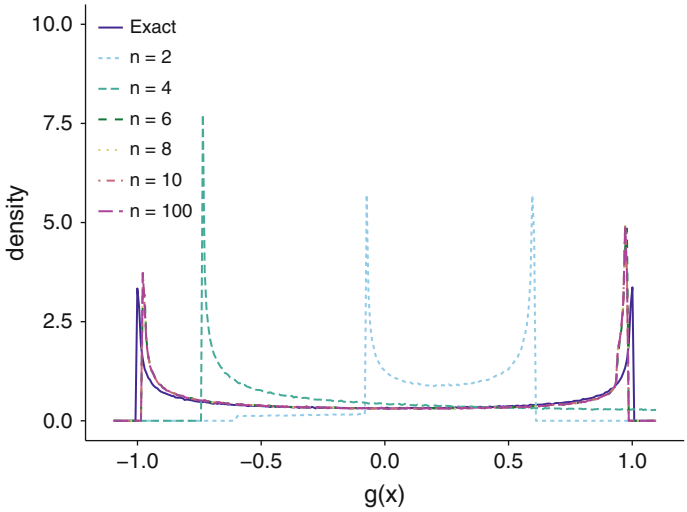

Fig. 9.5 PDF of the random variable $g(x) = \cos(x)$, where $x \sim \mathscr{U}(0, 2\pi)$ using a fifth-order Legendre expansion with various Gauss-Legendre quadrature rules to approximate the coefficients. This figure was generated from 10^6 samples of x that were used to evaluate $g(x)$ and the various approximations

Table 9.9 The convergence of the first six coefficients in the Legendre polynomial expansion $g(x) = \cos(x)$, where $x \sim \mathscr{U}(0, 2\pi)$ as estimated by Gauss-Legendre quadrature rules using different values of n

n	c_0	c_1	c_2	c_3	c_4	c_5
2	0.240619	0.000000	0.000000	0.000000	−0.842165	0.000000
3	−0.022454	0.000000	1.955092	0.000000	−2.639374	0.000000
4	0.001068	0.000000	1.478399	0.000000	−0.000000	0.000000
5	−0.000031	0.000000	1.521801	0.000000	−0.637516	0.000000
6	0.000001	0.000000	1.519760	0.000000	−0.579819	0.000000
7	0.000000	0.000000	1.519819	0.000000	−0.582523	0.000000
8	0.000000	0.000000	1.519818	0.000000	−0.582445	0.000000
9	0.000000	0.000000	1.519818	0.000000	−0.582447	0.000000
10	0.000000	0.000000	1.519818	0.000000	−0.582447	0.000000
100	0.000000	0.000000	1.519818	0.000000	−0.582447	0.000000

9.2.3 Beta Random Variables: Jacobi Polynomials

A random variable that takes on a value in the range, $[-1, 1]$, can often be described by a beta distribution.[6] A random variable Z that is beta-distributed is written as

[6]The definition of the beta distribution used here is not the typical statistician's distribution. That distribution has support on $[0, 1]$ and uses parameters α' and β' that are equal to $\alpha' = \alpha + 1$,

$Z \sim \mathscr{B}(\alpha, \beta)$, where $\alpha > -1$ and $\beta > -1$ are parameters. The PDF for Z is given by

$$f(z) = \frac{2^{-(\alpha+\beta+1)}}{\alpha+\beta+1} \frac{\Gamma(\alpha+1)+\Gamma(\beta+1)}{\Gamma(\alpha+\beta+1)} (1+z)^{\beta}(1-z)^{\alpha} \qquad z \in [-1, 1]. \tag{9.30}$$

The reason that is sometimes called a beta distribution is that the PDF can be expressed in terms of the beta function, $\mathrm{B}(\alpha, \beta)$

$$\mathrm{B}(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}, \tag{9.31}$$

as

$$f(z) = \frac{2^{-(\alpha+\beta+1)}}{\mathrm{B}(\alpha+1, \beta+1)} (1+z)^{\beta}(1-z)^{\alpha} \qquad z \in [-1, 1]. \tag{9.32}$$

There is some subtlety regarding the support of z. If α or β is less than 0, then one or both of the endpoints is excluded due to a singularity. The PDF for various values of α and β is shown in Fig. 9.6.

As before, we can scale the distribution to a general range $x \in [a, b]$ using Eqs. (9.16) and (9.17). The expectation operator in this case is given by

$$E[g(x)] = \int_{-1}^{1} g\left(\frac{b-a}{2}z + \frac{a+b}{2}\right) \frac{2^{-(\alpha+\beta+1)}(1+z)^{\beta}(1-z)^{\alpha}}{\mathrm{B}(\alpha+1, \beta+1)} \, dz. \tag{9.33}$$

From this we get following for a beta distribution on the range $[a, b]$:

$$\bar{x} = \frac{(\alpha+1)a + (\beta+1)b}{\alpha+\beta+2}, \qquad \mathrm{Var}(x) = \frac{(\alpha+1)(\beta+1)(a-b)^2}{(\alpha+\beta+2)^2(\alpha+\beta+3)}. \tag{9.34}$$

The Jacobi polynomials, $P_n^{(\alpha,\beta)}(z)$, are orthogonal polynomials under the weight $(1-z)^{\alpha}(1+z)^{\beta}$ for the interval $z \in [-1, 1]$. These polynomials can be defined in several ways, including the Rodrigues-type formula:

$$P_n^{(\alpha,\beta)}(z) = \frac{(-1)^n}{2^n n!}(1-z)^{-\alpha}(1+z)^{-\beta} \frac{d^n}{dz^n}\left\{(1-z)^{\alpha}(1+z)^{\beta}\left(1-z^2\right)^n\right\}. \tag{9.35}$$

The general form of these polynomials is given up to order 3 in Table 9.10. Note that when $\alpha = \beta = 0$, these polynomials are the Legendre polynomials.

---

and $\beta' = \beta + 1$. As we will see the definition in Eq. (9.30) is well-suited to expansion in Jacobi polynomials.

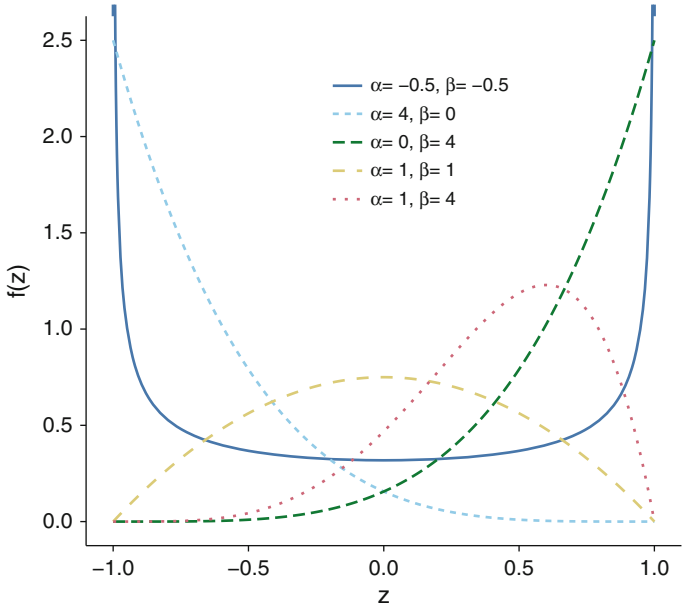

Fig. 9.6 PDF $Z \sim \mathscr{B}(\alpha, \beta)$ for several values of α and β. Note that when $\alpha = \beta$ the distribution is symmetric about $\frac{1}{2}$, and swapping α and β creates mirror images

Table 9.10 The first three Jacobi polynomials

n	$P_n^{(\alpha,\beta)}(z)$
0	1
1	$\frac{1}{2}(\alpha - \beta + z(\alpha + \beta + 2))$
2	$\frac{1}{2}(\alpha+1)(\alpha+2) + \frac{1}{8}(z-1)^2(\alpha+\beta+3)(\alpha+\beta+4) + \frac{1}{2}(z-1)(\alpha+2)(\alpha+\beta+3)$
3	$\frac{1}{6}(\alpha+1)(\alpha+2)(\alpha+3) + \frac{1}{48}(z-1)^3(\alpha+\beta+4)(\alpha+\beta+5)(\alpha+\beta+6)$ $+\frac{1}{8}(z-1)^2(\alpha+3)(\alpha+\beta+4)(\alpha+\beta+5) + \frac{1}{4}(z-1)(\alpha+2)(\alpha+3)(\alpha+\beta+4)$

These polynomials have the, somewhat ugly, orthogonality relation

$$\langle P_m^{(\alpha,\beta)}(z) P_n^{(\alpha,\beta)}(z) \rangle = \frac{2^{\alpha+\beta+1}}{2n+\alpha+\beta+1} \times \frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{\Gamma(n+\alpha+\beta+1)n!}\delta_{nm}, \quad \alpha, \beta > -1. \tag{9.36}$$

where

$$\langle g(z), h(z) \rangle = \int_{-1}^{1} (1-z)^{\alpha}(1+z)^{\beta} g(z)h(z)\, dz. \tag{9.37}$$

Note that if $n = 0$, then we can use the identity $\Gamma(z+1) = z\Gamma(z)$ to get the normalization constant used in the PDF for the beta distribution:

$$\frac{2^{\alpha+\beta+1}}{\alpha+\beta+1}\frac{\Gamma(\alpha+1)\Gamma(\beta+1)}{\Gamma(\alpha+\beta+1)} = 2^{\alpha+\beta+1}\mathrm{B}(\alpha+1,\beta+1).$$

A function that is square-integrable with respect to the inner product in Eq. (9.37) can be written as

$$g(x) = \sum_{n=0}^{\infty} c_n P_n^{(\alpha,\beta)}\left(\frac{a+b-2x}{a-b}\right), \qquad x \in [a,b], \tag{9.38}$$

where the constant is defined as

$$c_n = \langle P_n^{(\alpha,\beta)}(z) P_n^{(\alpha,\beta)}(z)\rangle^{-1} \int_{-1}^{1} g\left(\frac{b-a}{2}z + \frac{a+b}{2}\right) P_n^{(\alpha,\beta)}(z)(1-z)^{\alpha}(1+z)^{\beta}\, dz. \tag{9.39}$$

It is worthwhile to look at c_0. This formula gives us that, as before, c_0 is the mean (expected value) of $G \sim g(x)$:

$$c_0 = \frac{2^{-(\alpha+\beta+1)}}{\mathrm{B}(\alpha+1,\beta+1)} \int_{-1}^{1} g\left(\frac{b-a}{2}z + \frac{a+b}{2}\right)(1-z)^{\alpha}(1+z)^{\beta}\, dz = E[g(x)]. \tag{9.40}$$

Also, by construction the variance in $g(x)$ is the sum of the squares of the c_n for $n > 0$:

$$\mathrm{Var}(G) = E[g^2(x)] - (E[g(x)])^2 = \frac{2^{-(\alpha+\beta+1)}}{\mathrm{B}(\alpha+1,\beta+1)} \sum_{n=1}^{\infty} c_n^2 \langle P_n^{(\alpha,\beta)}(z) P_n^{(\alpha,\beta)}(z)\rangle. \tag{9.41}$$

As a test of this expansion, we will consider $g(x) = \cos(x)$, where $x \in [0, 2\pi]$ and x are derived from a standard beta random variable $Z \sim \mathscr{B}(4,1)$. The density plot from 10^6 samples of this distribution is shown in Fig. 9.7.

In this case we get

$$c_n = \langle P_n^{(\alpha,\beta)}(z) P_n^{(\alpha,\beta)}(z)\rangle^{-1} \int_{-1}^{1} \cos(\pi z + \pi)\, P_n^{(4,1)}(z)\, dz. \tag{9.42}$$

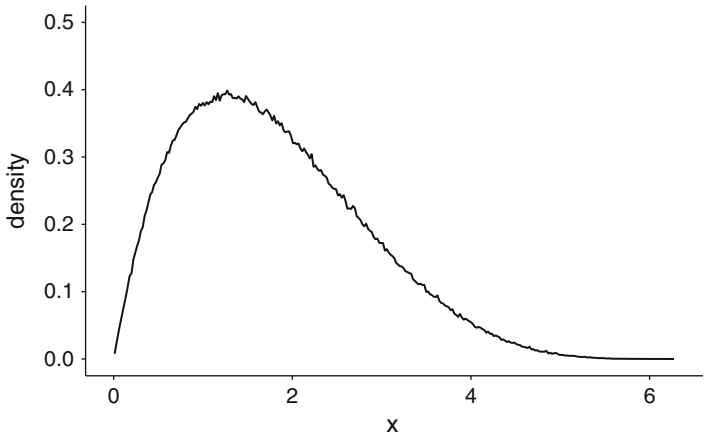

Fig. 9.7 Density plot of 10^6 samples of $x = \pi z + \pi$ where $Z \sim \mathscr{B}(4, 1)$. These samples were used to generate the results in Fig. 9.8

There is not a tidy formula for the coefficients, but we can calculate them (with the help of Mathematica). The mean value of $G \sim \cos(x)$ is

$$c_0 = -\frac{15\left(\pi^2 - 9\right)}{2\pi^4} \approx -0.0669551. \tag{9.43}$$

The expansion, through third-order is

$$\begin{aligned}\cos(x) \approx & -\frac{15\left(\pi^2 - 9\right)}{2\pi^4} + \frac{6\left(315 - 60\pi^2 + 2\pi^4\right)}{\pi^6} P_1^{(4,1)}(z) \\ & -\frac{35\left(630 - 75\pi^2 + \pi^4\right)}{2\pi^6} P_2^{(4,1)}(z) \\ & +\frac{12\left(-51975 + 8190\pi^2 - 315\pi^4 + 2\pi^6\right)}{\pi^8} P_3^{(4,1)}(z) \quad Z \sim \mathscr{B}(4, 1),\end{aligned} \tag{9.44}$$

and z is related to x via $x = \pi z + \pi$. For completeness, the values of the Jacobi polynomials in this equation are given in Table 9.11. If we expand the definitions of the Jacobi polynomials and make numerical approximations of the coefficients, we can get

$$\cos(x) \approx 2.50342z^3 + 4.14706z^2 - 0.536325z - 1.00484 \quad Z \sim \mathscr{B}(4, 1).$$

Table 9.11 The Jacobi polynomials $P_n^{(4,1)}$ through order 3

n	$P_n^{(4,1)}(z)$
0	1
1	$\frac{1}{2}(7z+3)$
2	$9(z-1)^2+24(z-1)+15$
3	$\frac{165}{8}(z-1)^3+\frac{315}{4}(z-1)^2+\frac{189(z-1)}{2}+35$

Table 9.12 The convergence of $\mathrm{Var}(G)$ for $g(x)=\cos(x)$, where $x=\pi z+\pi$ and $Z\sim\mathscr{B}(4,1)$

Order	Variance
1	0.3302376
2	0.4001581
4	0.4220198
6	0.4221829
8	0.4221832
∞	0.4221832

The variance of G is given by

$$\begin{aligned}\mathrm{Var}(G) &= \frac{2^{-(\alpha+\beta+1)}}{\mathrm{B}(\alpha+1,\beta+1)}\int_{-1}^{1}\cos^2(\pi z+\pi)(1-z)^{\alpha}(1+z)^{\beta}\,dz \\ &\quad -\left(\frac{15\left(\pi^2-9\right)}{2\pi^4}\right)^2 \\ &= \frac{1}{64}\left(\frac{135}{\pi^4}+32-\frac{60}{\pi^2}\right)-\frac{225\left(\pi^2-9\right)^2}{4\pi^8}\approx 0.4221832.\end{aligned} \tag{9.45}$$

The convergence of the variance estimate is given in Table 9.12. Notice that at fourth-order, the estimate is correct to three digits.

The convergence of the approximation to G as a function of the order of the Jacobi expansion is shown in Fig. 9.8. The "exact" distribution is determined by evaluating $g(x)$ at the 10^6 points shown in Fig. 9.7. By the fourth-order expansion, the overall character of the true distribution is captured. The eighth-order expansion is indistinguishable from the exact distribution.

9.2.4 Gauss-Jacobi Quadrature

To estimate the integrals required to compute a Jacobi expansion of a function of a beta-distributed random variables, we turn to Gauss-Jacobi quadrature. As in Gauss-Legendre quadrature (recall that Legendre polynomials are a special case of Jacobi polynomials), the quadrature rule looks like

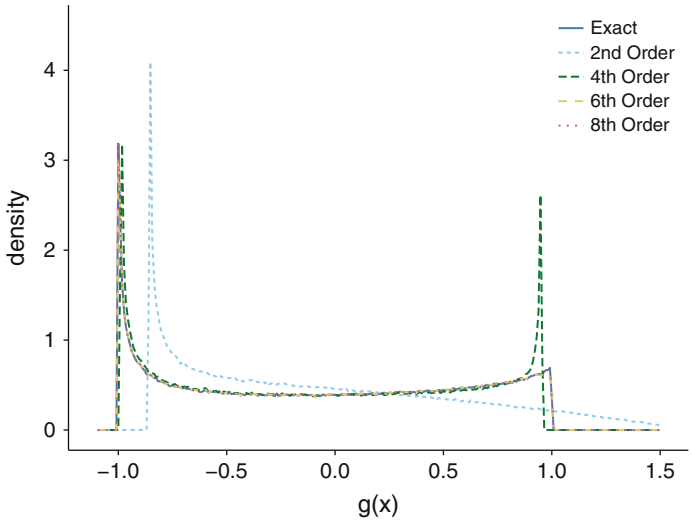

Fig. 9.8 PDF of the random variable $g(x) = \cos(x)$, where $x = \pi z + \pi$ and $Z \sim \mathscr{B}(4, 1)$, and various approximations. This figure was generated from 10^6 samples of x that were used to evaluate $g(x)$ and the various approximations

$$\int_{-1}^{1} f(z)(1-z)^{\alpha}(1+z)^{\beta}\,dz \approx \sum_{i=1}^{n} w_i f(z_i). \tag{9.46}$$

The abscissas, z_i, for the quadrature rule are the n roots of $P_n^{(\alpha,\beta)}(z)$, and the weights are given by

$$w_i = \frac{2n+\alpha+\beta+2}{n+\alpha+\beta+1}\frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{\Gamma(n+\alpha+\beta+1)(n+1)!}\frac{2^{\alpha+\beta}}{P_n'^{(\alpha,\beta)}(z_i)P_{n+1}^{(\alpha,\beta)}(z_i)}. \tag{9.47}$$

Here, unlike in Gauss-Legendre quadrature, the weights and abscissas depend on the choice of α and β. Therefore, we will not give an extensive table of coefficients because the generality makes the formulas lengthy. The first-order quadrature ($n = 1$) is

$$x_1 = \frac{b-a}{a+b+2}, \qquad w_1 = \frac{2^{a+b+1}\Gamma(a+2)\Gamma(b+2)}{(a+1)(b+1)\Gamma(a+b+2)}. \tag{9.48}$$

Beyond $n = 1$ the formulas for the weights and abscissas will not fit on a page, so they do not appear here.

For our example from above, where $Z \sim \mathscr{B}(4, 1)$, the quadrature rules are given in Table 9.13. Notice that unlike Gauss-Legendre quadrature rules, these rules are

Table 9.13 The abscissas and weights for Gauss-Jacobi quadrature up to order 5 with $\alpha = 4$ and $\beta = 1$

n	z_i	w_i
1	$-\frac{3}{7}$	$\frac{32}{15}$
2	0	$\frac{16}{21}$
	$-\frac{2}{3}$	$\frac{48}{35}$
3	0.273378	0.213558
	−0.313373	1.121472
	−0.778187	0.798303
4	0.451910	0.062182
	−0.037021	0.545298
	−0.497091	1.049649
	−0.840875	0.476204
5	0.573288	0.019805
	0.169240	0.233970
	−0.247188	0.732908
	−0.615377	0.850154
	−0.879964	0.296496

not symmetric about the origin. Moreover, the weights sum to the integral of the weight function over the domain:

$$\sum_{i=1}^{n} w_i = \int_{-1}^{1} (1-z)^4(1+z)\,dz = \frac{32}{15}. \tag{9.49}$$

We can use our previous example, of $g(x) = \cos(x)$, where $x = \pi z + \pi$ and $Z \sim \mathcal{B}(4, 1)$, as a test of estimating the coefficients using Gauss-Jacobi quadrature rules. In Fig. 9.9, the distribution, as approximated by a sixth-order Jacobi expansion, is computed using Gauss-Jacobi quadratures of different values of n. This means that we only need eight function evaluations to estimate the coefficients. We can see the convergence in the coefficients with the number of quadrature points in Table 9.14. This table bears out the observation that $n = 8$ is an adequate level of approximation.

9.2.5 Gamma Random Variables: Laguerre Polynomials

In the final class of random variable, we will consider gamma random variables. These random variables have support on $(0, \infty)$, and if x is a gamma-distributed random variable, we will write $x \sim \mathcal{G}(\alpha, \beta)$ where the PDF of the random variable is[7]

[7]There are several definitions of gamma random variables. One common definition has a different parameter $\alpha' = \alpha + 1$, but the same parameter β.

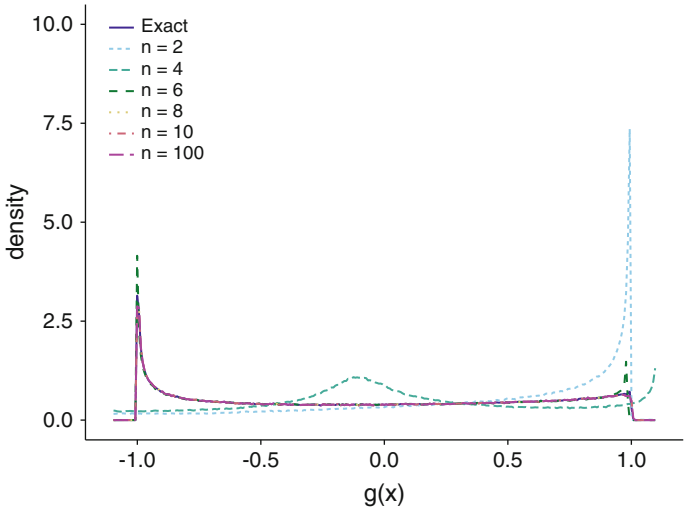

Fig. 9.9 PDF of the random variable $g(x) = \cos(x)$, where $x = \pi z + \pi$ and $Z \sim \mathscr{B}(4, 1)$ using a sixth-order Jacobi expansion with various Gauss-Jacobi quadrature rules to approximate the coefficients. This figure was generated from 10^6 samples of x that were used to evaluate $g(x)$ and the various approximations

Table 9.14 The convergence of the first seven coefficients in the Jacobi polynomial expansion $g(x) = \cos(x)$, where $x = \pi z + \pi$ and $Z \sim \mathscr{B}(4, 1)$ as estimated by Gauss-Jacobi quadrature rules using different values of n

n	c_0	c_1	c_2	c_3	c_4	c_5	c_6
2	−0.035714	−0.642857	0.000000	0.589286	−0.157292	−0.259369	−0.055473
3	−0.069292	−0.503277	0.282089	0.000000	−0.280037	0.478186	−0.131973
4	−0.066861	−0.514456	0.229440	0.132105	−0.000000	−0.135492	−0.210799
5	−0.066957	−0.513982	0.233355	0.120895	−0.058189	0.000000	0.060564
6	−0.066955	−0.513994	0.233197	0.121391	−0.053616	−0.011632	−0.000000
7	−0.066955	−0.513994	0.233201	0.121378	−0.053807	−0.011110	0.004949
8	−0.066955	−0.513994	0.233201	0.121378	−0.053802	−0.011124	0.004737
9	−0.066955	−0.513994	0.233201	0.121378	−0.053802	−0.011124	0.004742
10	−0.066955	−0.513994	0.233201	0.121378	−0.053802	−0.011124	0.004742
100	−0.066955	−0.513994	0.233201	0.121378	−0.053802	−0.011124	0.004742

$$f(x) = \frac{\beta^{(\alpha+1)} x^{\alpha} e^{-\beta x}}{\Gamma(\alpha + 1)}, \qquad x \in (0, \infty), \quad \alpha > -1, \ \beta > 0. \tag{9.50}$$

The distribution gets its name from the appearance of the gamma function in the PDF.

As in other variables, it will be useful to have a standardized gamma random variable. In this case we define a $Z \sim \mathscr{G}(\alpha, 1)$, so that Z has the PDF

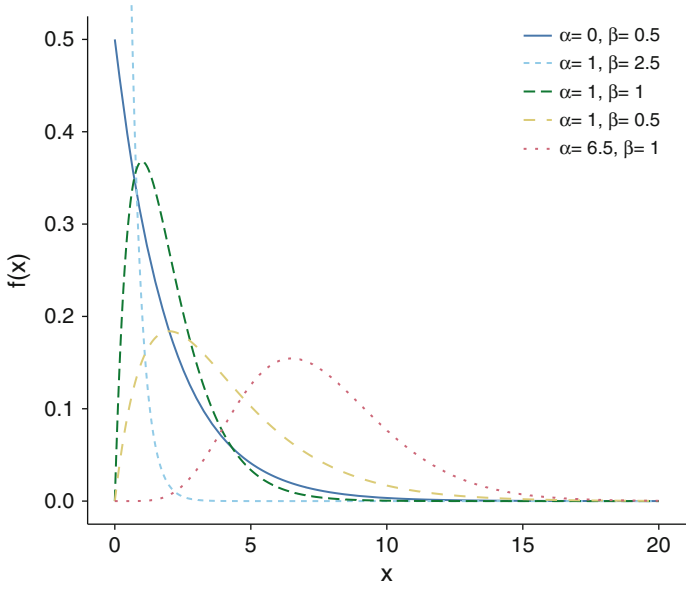

Fig. 9.10 PDF $x \sim \mathcal{G}(\alpha, \beta)$ for several values of α and β. Note that adjusting α moves the peak of the distribution and β scales the distribution along x

$$f(z) = \frac{z^{\alpha} e^{-z}}{\Gamma(\alpha + 1)}, \qquad z \in (0, \infty), \quad \alpha > -1. \tag{9.51}$$

We can change from Z to x using a simple scaling

$$z = \beta x. \tag{9.52}$$

The PDF for a gamma random variable with several different values for the α and β parameters is shown in Fig. 9.10. Here we see that α moves the peak of the distribution and that β, as we mentioned above, scales the distribution.

The expectation operator for a gamma random variable can be written as

$$E[g(x)] = \int_0^{\infty} g(x) \frac{\beta^{(\alpha+1)} x^{\alpha} e^{-\beta x}}{\Gamma(\alpha + 1)} \, dx = \int_0^{\infty} g\left(\frac{z}{\beta}\right) \frac{z^{\alpha} e^{-z}}{\Gamma(\alpha + 1)} \, dz. \tag{9.53}$$

Additionally, the mean and variance are given by

$$\bar{x} = \frac{\alpha + 1}{\beta}, \qquad \text{Var}(x) = \frac{\alpha + 1}{\beta^2}. \tag{9.54}$$

Table 9.15 The first three generalized Laguerre polynomials

n	$L_n^{(\alpha)}(z)$
0	1
1	$\alpha - x + 1$
2	$\frac{1}{2}\left(\alpha^2 + 3\alpha + x^2 - 2\alpha x - 4x + 2\right)$
3	$\frac{1}{6}\left(\alpha^3 + 6\alpha^2 + 11\alpha - x^3 + 3\alpha x^2 + 9x^2 - 3\alpha^2 x - 15\alpha x - 18x + 6\right)$

The orthogonal polynomials that we will use with functions of a gamma random variable are generalized Laguerre polynomials. Rodrigues' formula for these polynomials is

$$L_n^{(\alpha)}(x) = \frac{x^{-\alpha}e^x}{n!}\frac{d^n}{dx^n}\left(e^{-x}x^{n+\alpha}\right). \tag{9.55}$$

Some low-order generalized Laguerre polynomials are given in Table 9.15.

The generalized Laguerre polynomials have the following orthogonality condition

$$\int_0^\infty x^\alpha e^{-x} L_n^{(\alpha)}(x) L_m^{(\alpha)}(x)\, dx = \frac{\Gamma(n+\alpha+1)}{n!}\delta_{n,m}. \tag{9.56}$$

The generalized Laguerre polynomials form a basis for functions on $(0, \infty)$ that are square-integrable with the inner product

$$\langle g(z), h(z)\rangle = \int_0^\infty z^\alpha e^{-z} g(z) h(z)\, dz. \tag{9.57}$$

Therefore, we can write a function $g(x)$ where $x \sim \mathcal{G}(\alpha, \beta)$ using the following expansion

$$g(x) = \sum_{n=0}^{\infty} c_n L_n^{(\alpha)}(\beta x), \tag{9.58}$$

where the expansion coefficients are

$$c_n = \frac{n!}{\Gamma(n+\alpha+1)}\int_0^\infty g\left(\frac{z}{\beta}\right) z^\alpha e^{-z} L_n^{(\alpha)}(z)\, dz. \tag{9.59}$$

The value of c_0 is once again the mean of $G \sim g(x)$ where $x \sim \mathcal{G}(\alpha, \beta)$:

$$c_0 = \int_0^\infty g\left(\frac{z}{\beta}\right)\frac{z^\alpha e^{-z}}{\Gamma(\alpha+1)}\, dz = E[g(x)]. \tag{9.60}$$

The variance of G is related to the sum of the squares of the expansion coefficients:

$$\begin{aligned}\mathrm{Var}(G) &= \int_0^\infty \left(\sum_{n=0}^\infty c_n L_n^{(\alpha)}(z)\right)^2 \frac{z^\alpha e^{-z}}{\Gamma(\alpha+1)}\,dz - c_0^2 \\ &= \sum_{n=1}^\infty \frac{\Gamma(n+\alpha+1)}{\Gamma(\alpha+1)n!} c_n^2. \end{aligned} \tag{9.61}$$

As an example we will examine $G \sim g(x)$ where $g(x) = \cos x$ and $x \sim \mathscr{G}(1,2)$. For the generalized Laguerre expansion of this function, we have expansion coefficients given by

$$c_n = \frac{n!}{\Gamma(n+2)} \int_0^\infty \cos\left(\frac{z}{2}\right) z e^{-z} L_n^{(1)}(z)\,dz. \tag{9.62}$$

The expected value of G is

$$c_0 = \int_0^\infty \cos\left(\frac{z}{2}\right) z e^{-z}\,dz = \frac{12}{25}. \tag{9.63}$$

The expansion to third-order is

$$\begin{aligned}\cos(x) \approx\ & \frac{12}{25} + \frac{44}{125}(2-2x) + \frac{28}{625}\left(2x^2 - 6x + 3\right) \\ & + \frac{656}{9375}\left(x^3 - 6x^2 + 9x - 3\right) \quad x \sim \mathscr{G}(1,2). \end{aligned} \tag{9.64}$$

The variance of G is given by

$$\mathrm{Var}(G) = \sum_{n=1}^\infty \frac{\Gamma(n+2)}{\Gamma(2)n!} c_n^2 = \frac{337}{1250} = 0.2696. \tag{9.65}$$

The convergence of the variance estimate is given in Table 9.16. As we saw previously, the variance is well-estimated by the fourth-order expansion. We will also see that the fourth-order expansion is also a good estimate of the distribution of G.

The convergence of the approximation to G as a function of the order of the Laguerre expansion is shown in Fig. 9.11. The "exact" distribution is determined by evaluating $g(x)$ at the 10^6 samples from $x \sim \mathscr{G}(1,2)$. By the fourth-order expansion, the overall character of the true distribution is captured.

Table 9.16 The convergence of $\mathrm{Var}(G)$ for $g(x) = \cos(x)$, where $x \sim \mathcal{G}(1, 2)$

Order	Variance
1	0.2478080
2	0.2538291
4	0.2693313
6	0.2695484
8	0.2695967
∞	0.2696000

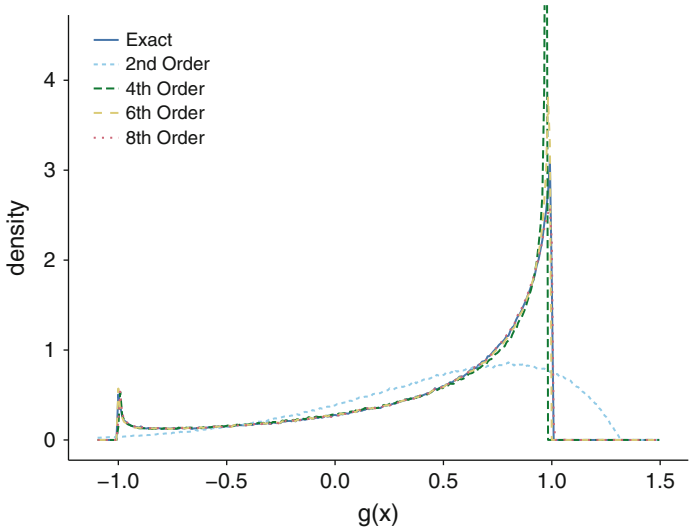

Fig. 9.11 PDF of the random variable $g(x) = \cos(x)$, where $x \sim \mathcal{G}(1, 2)$, and various approximations. This figure was generated from 10^6 samples of x that were used to evaluate $g(x)$ and the various approximations

9.2.6 Gauss-Laguerre Quadrature

To estimate the integrals required to compute a generalized Laguerre expansion of a function of a gamma-distributed random variables, we turn to generalized Gauss-Laguerre quadrature. The quadrature rule has the form

$$\int_0^\infty f(z) z^\alpha e^{-z}\, dz \approx \sum_{i=1}^{n} w_i f(z_i). \tag{9.66}$$

The abscissas, z_i, for the quadrature rule are the n roots of $L_n^{(\alpha)}(z)$, and the weights are given by

$$w_i = \frac{\Gamma(n+\alpha) z_i}{n!(n+\alpha)(L_{n-1}^{\alpha}(z_i))^2}. \tag{9.67}$$

Table 9.17 The abscissas and weights for generalized Gauss-Laguerre quadrature up to order 5 with $\alpha = 1$

n	z_i	w_i
1	2	1
2	$3 \pm \sqrt{3}$	$\frac{3\pm\sqrt{3}}{3\left(2-\left(3\pm\sqrt{3}\right)\right)^2}$
3	7.758770	0.020102
	3.305407	0.391216
	0.935822	0.588681
4	10.953894	0.001316
	5.731179	0.074178
	2.571635	0.477636
	0.743292	0.446871
5	14.260103	0.000069
	8.399067	0.008720
	4.610833	0.140916
	2.112966	0.502281
	0.617031	0.348015

The first-order quadrature ($n = 1$) is

$$x_1 = 1 + \alpha, \qquad w_1 = \frac{(\alpha + 1)\Gamma(a + 1)}{a + 1}. \tag{9.68}$$

For $n = 2$ we have

$$x_{1,2} = \alpha \pm \sqrt{\alpha + 2}, \qquad w_{1,2} = \frac{\left(3 \pm \sqrt{3}\right)\Gamma(a + 2)}{2(a + 2)\left(a + 1 - \left(3 \pm \sqrt{3}\right)\right)^2}. \tag{9.69}$$

Beyond second-order the quadratures are too lengthy to write for a general value of α. Note that if $\alpha = 0$, then the quadrature rule reduces to simple Gauss-Laguerre quadrature.

To use the generalized Gauss-Laguerre quadratures to compute the inner products for the Laguerre expansion of a gamma-distributed random variable, $x \sim \mathcal{G}(\alpha, \beta)$ as

$$\int_0^\infty f\left(\frac{z}{\beta}\right) z^\alpha e^{-z}\, dz \approx \sum_{i=1}^{n} w_i f\left(\frac{z_i}{\beta}\right). \tag{9.70}$$

For our example from above, where $x \sim \mathcal{G}(1, 2)$, the quadrature rules are given in Table 9.17. In this case the weights sum to the integral of the weight function over the domain:

$$\sum_{i=1}^{n} w_i = \int_0^\infty z e^{-z}\, dz = 2. \tag{9.71}$$

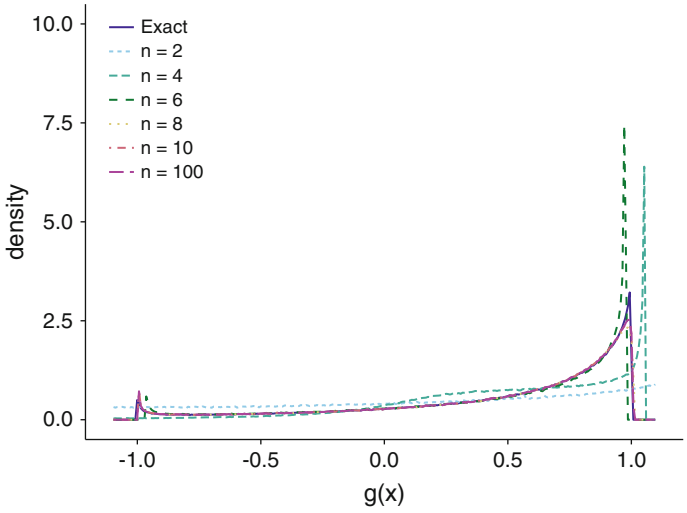

Fig. 9.12 PDF of the random variable $g(x) = \cos(x)$, where $x \sim \mathcal{G}(1, 2)$ using a sixth-order Laguerre expansion with various Gauss-Laguerre quadrature rules to approximate the coefficients. This figure was generated from 10^6 samples of x that were used to evaluate $g(x)$ and the various approximations

We can use our previous example, of $g(x) = \cos(x)$, where $x \sim \mathcal{G}(1, 2)$ as a test of estimating the coefficients using generalized Gauss-Laguerre quadrature rules. In Fig. 9.12, the distribution, as approximated by a fifth-order Laguerre expansion, is computed using generalized Gauss-Laguerre quadratures of different values of n. The distribution at about $n = 8$ the approximation is fairly accurate. We can see the convergence in the coefficients with the number of quadrature points in Table 9.14. This table bears out the observation that $n = 8$ is an adequate level of approximation.

9.2.7 *Example from a PDE: Poisson's Equation with an Uncertain Source*

The examples we have seen so far have been functions that have been simple to evaluate. In such examples, there is no benefit to minimizing the number of function evaluations. For a more realistic example where the function evaluations are expensive, but not too expensive, we will look at a quantity related to the solution of the 2-D Poisson's equation with Dirichlet boundary conditions:

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u(x, y) = -q(x, y). \tag{9.72}$$

$$u(1, y) = u(x, 1) = u(-1, y) = u(x, -1) = 0. \tag{9.73}$$

Table 9.18 The convergence of the first six coefficients in the generalized Laguerre polynomial expansion $g(x) = \cos(x)$, where $x \sim \mathcal{G}(1, 2)$ as estimated by generalized Gauss-Laguerre quadrature rules using different values of n

n	c_0	c_1	c_2	c_3	c_4	c_5
2	0.484528	0.438701	0.000000	−0.219350	−0.223933	−0.140776
3	0.478523	0.343285	0.077209	−0.000000	−0.046325	−0.099540
4	0.480185	0.352313	0.038293	−0.054229	−0.000000	0.036153
5	0.479984	0.352043	0.045559	−0.053931	−0.036908	−0.000000
6	0.480001	0.351990	0.044746	−0.052110	−0.029267	−0.004078
7	0.480000	0.352001	0.044801	−0.052532	−0.029939	−0.000867
8	0.480000	0.352000	0.044800	−0.052475	−0.029968	−0.001564
9	0.480000	0.352000	0.044800	−0.052480	−0.029949	−0.001480
10	0.480000	0.352000	0.044800	−0.052480	−0.029952	−0.001484
100	0.480000	0.352000	0.044800	−0.052480	−0.029952	−0.001485

The source q will be normal in space with an uncertain center in y:

$$q(x, y) = \exp\left[-x^2 - (y - \omega)^2\right]. \tag{9.74}$$

The center of the normal in the y-coördinate will be a uniform random variable in the range $[-0.25, 0.25]$ (i.e., $\omega \sim \mathcal{U}(-0.25, 0.25)$). We are interested in the integral over a quarter of the domain. Our quantity of interest is therefore

$$g(\omega) = \int_0^1 dx \int_0^1 dy\, u(x, y; \omega). \tag{9.75}$$

The notation $u(x, y; \omega)$ denotes that the solution depends on the center of Gaussian ω (Table 9.18).

Because ω is a uniform random variable, we will use a Legendre expansion to compute an approximation to $G \sim g(\omega)$. From Eq. (9.22), we are interested in computing the integral

$$c_n = \frac{2n + 1}{2} \int_{-1}^{1} g\left(\frac{z}{4}\right) P_n(z)\, dz. \tag{9.76}$$

We will estimate the Legendre expansion coefficients using Gauss-Legendre quadrature. For example, using an $n = 2$ quadrature rule, we would estimate the coefficients as

$$c_n \approx \frac{2n + 1}{2}\left(g\left(-\frac{1}{4\sqrt{3}}\right) P_n\left(-\frac{1}{4\sqrt{3}}\right) + g\left(\frac{1}{4\sqrt{3}}\right) P_n\left(\frac{1}{4\sqrt{3}}\right)\right). \tag{9.77}$$

Table 9.19 The convergence of the first six coefficients in the 2-D Poisson's equation example as a function of the number of Gauss-Legendre quadrature points used

n	c_0	c_1	c_2	c_3	c_4	c_5
1	0.386712	0.000000	−0.966780	0.000000	1.305153	0.000000
2	0.381378	0.000000	−0.000000	−0.000000	−1.334823	−0.000000
3	0.381406	−0.000000	−0.010613	−0.000000	0.014327	0.000000
4	0.381406	−0.000000	−0.010559	0.000000	−0.000000	0.000000
5	0.381406	0.000000	−0.010559	0.000000	0.000071	−0.000000
6	0.381406	−0.000000	−0.010559	0.000000	0.000071	−0.000000
7	0.381406	−0.000000	−0.010559	0.000000	0.000071	−0.000000
8	0.381409	0.000000	−0.010567	−0.000000	0.000079	0.000000
9	0.381406	0.000000	−0.010559	−0.000000	0.000071	−0.000000
10	0.381406	0.000000	−0.010559	−0.000000	0.000071	−0.000000

Note, to compute the c_n in this case will require solving Poisson's equation twice, each time with different sources, and computing the integral in Eq. (9.75). There are, at least, a countably infinite number of ways to estimate the solution to Poisson's equation. Here we will use Mathematica's `NDSolve` function. Solving Poisson's equation with these two values of ω gives

$$g\left(-\frac{1}{4\sqrt{3}}\right) = 0.381378, \qquad g\left(\frac{1}{4\sqrt{3}}\right) = 0.381378.$$

Therefore, for example, c_0 will be

$$c_0 \approx \frac{1}{2}\left[g\left(-\frac{1}{4\sqrt{3}}\right) + g\left(\frac{1}{4\sqrt{3}}\right)\right] = 0.381378. \tag{9.78}$$

In Table 9.19 estimates for the expansion coefficients up to c_n are shown. Note that in the best case, we could only hope for a quadrature rule with n points to integrate up to c_{2n-1} accurately and that this would only be the case if g were a constant function. From this table it seems that the integrals are accurate (though not exact) up to c_n for an n point quadrature rule once $n > 2$.

Using the results from Table 9.19, we can create an empirical PDF of G for different quadrature rules. For a given polynomial expansion, generating 10^6 samples requires only evaluating that many polynomials. In Fig. 9.13, these PDFs are shown for quadrature rules using 2, 4, 6, and 10 points as well as the PDF from 3000 Monte Carlo samples of G by randomly selecting ω's. Note that with only six function evaluations using the $n = 6$ quadrature rule, we get a better representation of G than thousands of Monte Carlo samples and a savings of about 0.75 h on my laptop.

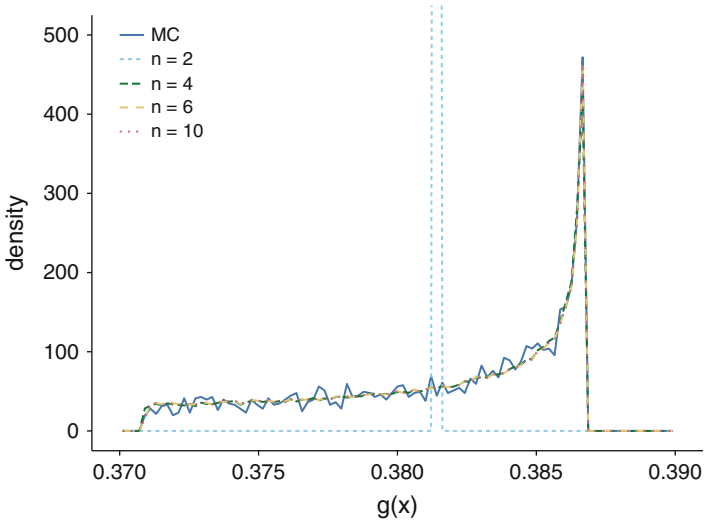

Fig. 9.13 PDF of the random variable $g(\omega) = \int_0^1 dx \int_0^1 dy\, u(x, y; \omega)$, where $\omega \sim \mathscr{U}(-0.25, 0.25)$ and u is the solution to Eq. (9.72), using several different Gauss-Legendre quadrature rules and a Monte Carlo simulation using 3×10^3 numerical solutions of Poisson's equation

9.3 Issues with Projection Techniques

Now that we have a discussed how to express a QoI as a projection onto a linear combination of orthogonal polynomials, it is a good point to point out some of the warts of this particular method. The projection method uses a single expansion to represent the QoI; such an expansion is called a global expansion. Whatever values the random variable takes, the expansion coefficients do not change. Global expansions work well and have rapid convergence if the underlying function is smooth. However, if the function being projected onto polynomials is not smooth, the expansions demonstrate large oscillations known as Gibbs' phenomena (Boyd 2001). These are especially present when the function is discontinuous. Gibbs' oscillations are present when a global polynomial (that is a single polynomial) is used to approximate a non-smooth function.

In practice many quantities of interest are discontinuous at some point in random variable space. An example of this would be a QoI that is zero until a threshold is met and then jumps up to a nonzero value. Such a function would not be well represented by projection onto orthogonal polynomials. To demonstrate this consider the function

$$g(x) = H(x + 1) - \frac{1}{2}H(x),$$

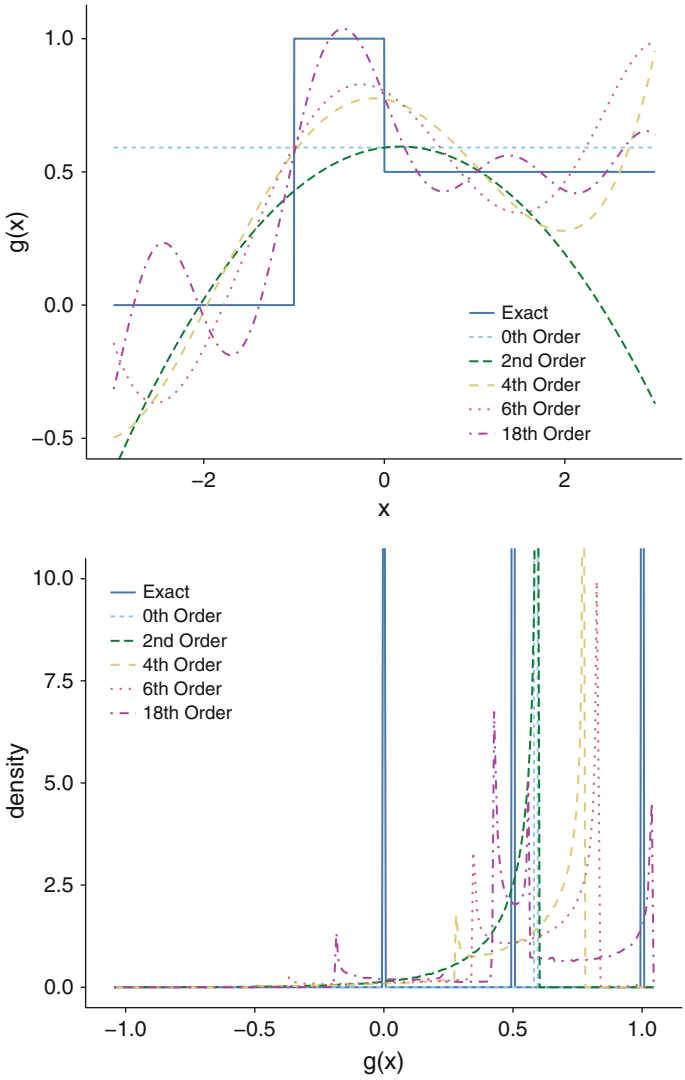

Fig. 9.14 Projection results for the approximation to the function $g(x) = H(x+1) - \frac{1}{2}H(x)$ where x is a standard normal random variable. The top panel is the Hermite approximation at various orders, and the bottom panel is the histogram from 10^6 samples of x

where $H(x)$ is the Heaviside step function and x is a standard normal random variable. Using Hermite expansions of various orders fails to give a reasonable approximation to the function, as shown in Fig. 9.14. Also, the empirical distribution has issues that even an 18th order expansion has spurious artifacts near the three possible values of the function. Moreover, the approximations using Hermite polynomials indicate that the probability of $g(x)$ being between 0.5 and 1 is fairly

large even though it is not possible for the true solution to have such a value. The takeaway from these results is that one needs to use caution when using expansion techniques if there is a possibility that the QoI is not a smooth function of the random variables.

The situation is the same with the collocation methods the we introduce later: non-smooth functions are not well-suited to global polynomial interpolation. The shortcomings of global expansions are worse when dealing with stochastic finite elements, a topic discussed toward the end of this chapter. There are approaches to correcting oscillations of global expansions including "local" expansions that use different projections in different domains and spline-based reconstructions that use piecewise polynomials. A hurdle for the local expansion techniques is that one often does not know where any discontinuities or other non-smooth features of the function lie. Therefore, effort (i.e., function evaluations) must be expended to find these points, potentially dulling the usefulness of the method.

9.4 Multidimensional Projections

It is likely that in a realistic problem, there will be several sources of uncertainty and several uncertain parameters. It may also be possible that the different parameters may have different types of distributions. Let us consider a generic function of d random variables, θ_i, with an expansion given by

$$g(\theta_1, \ldots, \theta_d) = \sum_{l_1=0}^{\infty} \cdots \sum_{l_d=0}^{\infty} c_{l_1,\ldots,l_d} \mathfrak{P}_{l_1,\ldots,l_d}(\theta_1, \ldots, \theta_d). \tag{9.79}$$

Here $\mathfrak{P}_{l_1,\ldots,l_d}(\theta_1, \ldots, \theta_d)$ is a product of the d orthogonal polynomials,

$$\mathfrak{P}_{l_1,\ldots,l_d}(\theta_1, \ldots, \theta_d) = \prod_{i=1}^{d} P_{l_i}(\theta_i), \tag{9.80}$$

and the expansion coefficients are

$$c_{l_1,\ldots,l_d} = \int_{D_1} d\theta_1 \cdots \int_{D_d} d\theta_d \, g(\theta_1, \ldots, \theta_d) \mathfrak{P}_{l_1,\ldots,l_d}(\theta_1, \ldots, \theta_d) \mathfrak{w}(\theta_1, \ldots, \theta_d), \tag{9.81}$$

and $\mathfrak{w}(\theta_1, \ldots, \theta_d)$ is the product of the weight functions for the d bases. If the sum is truncated at degree N polynomials, then there will be $(1 + N)^d$ terms in the expansion.

As a simple example, consider the function $g = \cos(\theta_1)\cos(\theta_2)$ with $\theta_i \sim \mathscr{U}(0, 2\pi)$. A second-order expansion would have the form

$$\begin{aligned} g(\theta_1, \theta_2) &= c_{0,0} + c_{1,0} P_1(\pi\theta_1 + \pi) + c_{0,1} P_1(\pi\theta_2 + \pi) + c_{2,0} P_2(\pi\theta_1 + \pi) \\ &+ c_{0,2} P_2(\pi\theta_2 + \pi) + c_{1,1} P_1(\pi\theta_1 + \pi) P_1(\pi\theta_2 + \pi) \\ &+ c_{2,1} P_2(\pi\theta_1 + \pi) P_1(\pi\theta_2 + \pi) + c_{1,2} P_1(\pi\theta_1 + \pi) P_2(\pi\theta_2 + \pi) \\ &+ c_{2,2} P_2(\pi\theta_1 + \pi) P_2(\pi\theta_2 + \pi). \end{aligned} \tag{9.82}$$

To compute the expansion coefficients, we can use what is known as a tensor-product quadrature rule. Here we take a 1-D quadrature rule with n points and weights given by $\{w_i, x_i\}$ for $i = 1 \ldots n$ that we denote as Q_n so that

$$Q_n f(x) = \sum_{l=1}^{n} w_\ell f(x_\ell), \tag{9.83}$$

and apply it over all dimensions as

$$Q_n^{(d)} g(\theta_1, \ldots, \theta_d) = \sum_{l_1=1}^{n} \cdots \sum_{l_d=1}^{n} w_{l_1} \cdots w_{l_d} g(\theta_{1l_1}, \ldots, \theta_{dl_d}), \tag{9.84}$$

where θ_{i,l_j} is the ith input evaluated at its jth point in the quadrature set. It is sometimes convenient to write $Q^{(d)}$ as a tensor product of 1-D quadrature rules. We define a tensor product of two quadrature rules as

$$Q_n \otimes Q_m = \{\{w_i w_j, (x_i, x_j)\} : i = 1 \ldots n, j = 1 \ldots m\}. \tag{9.85}$$

Therefore, we can write a tensor-product quadrature comprised of n point quadratures as

$$Q_n^{(d)} g(\theta_1, \ldots, \theta_d) = (Q_n^{(1)} \otimes \cdots \otimes Q_n^{(d)}) g. \tag{9.86}$$

We could in principle have each dimension have a different number of quadrature points, and in many cases, this will make the calculation more efficient.

The number of quadrature points scales geometrically with d. This is the so-called curse of dimensionality because the number of function evaluations needed explodes as d gets larger. For example, using a two-point quadrature rule, when $d = 26$, requires one simulation for every person in Germany. Even worse $d = 78$ requires a mole (6×10^{23}) of calculations for the $n = 8$ rule. In a full-scale engineering system, 78 uncertain parameters are not out of the question.

As an example, the tensor product quadrature rule for $d = 2$ comprised of 1-D, six-point Gauss-Legendre quadrature rules is shown in Fig. 9.15. Two things are evident in this figure: the weights are much larger in the middle of domain, and the points are more densely packed near the corners. These effects are even more pronounced as the number of points in the quadrature set goes up.

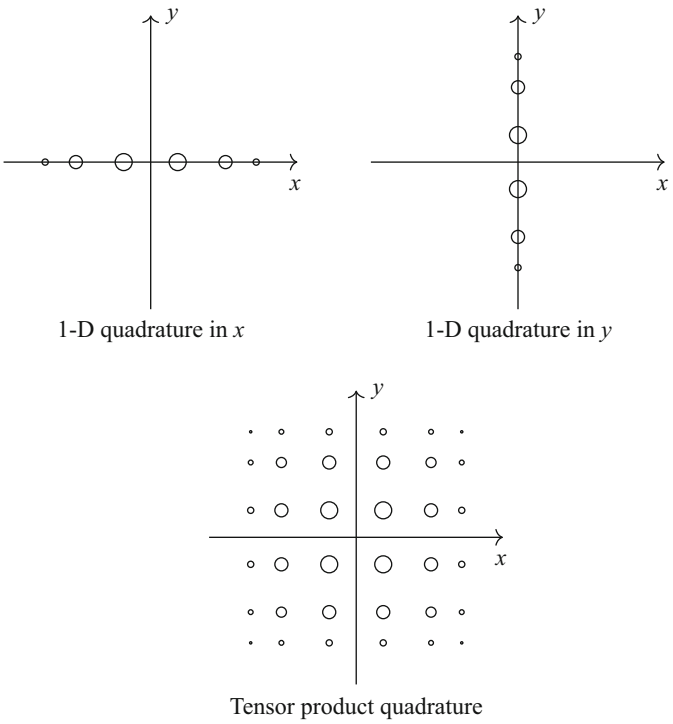

Fig. 9.15 Illustration of the 2-D tensor-product quadrature derived from the six-point Gauss-Legendre quadrature set. The size of a point is proportional to its weight

9.4.1 Example Three-Dimensional Expansion: Black-Scholes Pricing Model

For an example of a polynomial chaos expansion in multiple dimensions, we will look at the solution to the Black-Scholes partial differential equation for the value of a call option. A call option gives the holder the ability to purchase a stock at a given price, called the strike price, at a given future date. The value of the option is a function of the current price of the stock (S), the strike price (K), the time to expiration in years (T), the risk-free interest rate (r), the dividend rate the stock pays q, and the volatility of the stock (σ). Three of these, r, q, and σ, are uncertain parameters.

The Black-Scholes model is based on assuming that the stock price follows geometric Brownian motion. The solution for the price of an option from the Black-Scholes model can be given by

$$p = e^{-rT} \left(F\Phi(v_1) - K\Phi(v_2) \right), \tag{9.87}$$

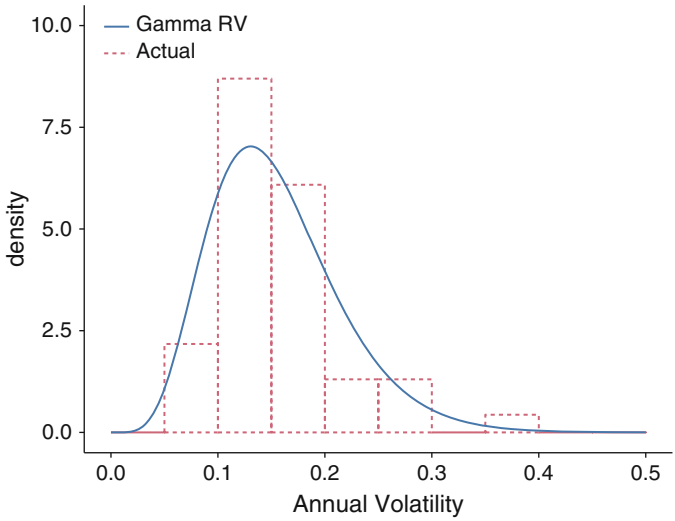

Fig. 9.16 The empirical distribution and a fitted Gamma distribution in the annual percentage change in Coca-Cola stock for each year between 1970 and 2015. The distribution has a mean of 0.154083 and variance of 0.0036984. This corresponds to a Gamma distribution of $\Sigma \sim \mathcal{G}(5.46636, 41.8142)$

where

$$F = Se^{(r-q)T}, \tag{9.88}$$

$$v_1 = \frac{\log \frac{S}{K} + (r - q + \frac{1}{2}\sigma^2)T}{T\sqrt{T}}, \qquad v_2 = v_1 - \sigma\sqrt{T}, \tag{9.89}$$

and $\Phi(z)$ is the standard normal CDF.

We are interested in calculating the current value of a call option for stock in the Coca-Cola company, ticker symbol KO. On 15 August 2016, KO was trading at \$44.15. We will consider a call with strike price of \$44. The option expiration is 158 days away ($T = 0.432877$). This option is trading at \$1.46. We need to estimate the distribution of random variables, r, q, and σ. The distribution that we will use for the volatility σ will be based on the actual annual standard deviations of daily returns for the years from 1970 to 2015. The histogram of these 45 volatilities is shown in Fig. 9.16, along with a Gamma distribution that matches the mean and variance of the observations, $\Sigma \sim \mathcal{G}(5.46636, 41.8142)$. This distribution is found by solving Eq. (9.54) to match the observed mean and variance in the data; this is an application of the method of moments discussed in Sect. 7.1.3. Note that the distribution indicates that 10% of the time, the volatility will be greater than 23.6%. For the interest rate, r, we will use the benchmark LIBOR 30-day interest rate with a Gamma distribution $r = 0.0048x$ with $x \sim \mathcal{G}(0, 1)$. This distribution had a mean

of the current rate (0.48%). For the dividend rate, we will use a uniform distribution so that $Q \sim \mathscr{U}(0.025, 0.045)$.

The expansion of $p(x, D, \Sigma)$ will have the form

$$p(x, Q, \Sigma) = \sum_{l_x=0}^{\infty} \sum_{l_d=0}^{\infty} \sum_{l_\sigma=0}^{\infty} c_{l_x l_d l_\sigma} L_{l_x}^{(0)}(x) P_{l_d}\left(\frac{2d - 0.7}{0.2}\right) L_{l_\sigma}^{(5.46636)}(41.8142\sigma). \tag{9.90}$$

From this equation, we can compute the mean of the distribution, c_{000} as

$$\begin{aligned} \bar{p} = c_{000} &= \int_0^{\infty} dx \int_{0.025}^{0.045} dq \int_0^{\infty} dz\, p\left(x, q, \frac{z}{41.8142}\right) \\ &\quad \times \frac{z^{5.46636}}{\Gamma(6.46636)} e^{-x-z} \left(\frac{1}{0.02}\right) \approx 1.56662. \end{aligned} \tag{9.91}$$

Note that this is slightly higher than the price the option is trading at, \$1.46.

Because the price of the option is a well-behaved function, we will expand p with polynomial degree up to order four:

$$p(x, Q, \Sigma) = \sum_{l_x=0}^{4} \sum_{l_d=0}^{4} \sum_{l_\sigma=0}^{4} c_{l_x l_d l_\sigma} L_{l_x}^{(0)}(x) P_{l_d}\left(\frac{2d - 0.7}{0.2}\right) L_{l_\sigma}^{(5.46636)}(41.8142\sigma). \tag{9.92}$$

Such an expansion will have $5^3 = 125$ terms. Using tensor-product Gauss quadrature—Gauss-Laguerre in x and σ, Gauss-Legendre in q—we can estimate these coefficients. The results from these calculations with various numbers of points in the 1-D quadrature rules that comprise the tensor-product quadrature are shown Fig. 9.17. This figure indicates the maximum single polynomial degree in each point using a color/shape. Here we only show coefficients with a magnitude larger than 10^{-6}; for the $n = 2$ rules, we do not show any coefficients corresponding to degree three or four polynomials.

From Fig. 9.17 we can see that the $n = 2$ quadrature rule does a good job of estimating the low-order, large-magnitude coefficients. This indicates that most of the variation in the distribution can be captured using only $2^3 = 8$ evaluations of the function. The higher-order coefficients have a smaller magnitude and can be captured using $n = 4$ rules, and the largest, significant coefficient c_{004} or the coefficient for a quartic in volatility can be captured using $n = 6$ or a total of $6^3 = 216$ function evaluations.

One way to compare the quadrature rules is to look at the convergence of the variance. The variance in the price is

$$\mathrm{Var}(P) = \sum_{l_x=1}^{\infty} \sum_{l_d=1}^{\infty} \sum_{l_\sigma=1}^{\infty} \frac{\Gamma(l_x + 1)\Gamma(l_\sigma + 6.46636)}{l_x! l_\sigma! \Gamma(6.46636)(2l_d + 1)} c_{l_x l_d l_\sigma}^2. \tag{9.93}$$

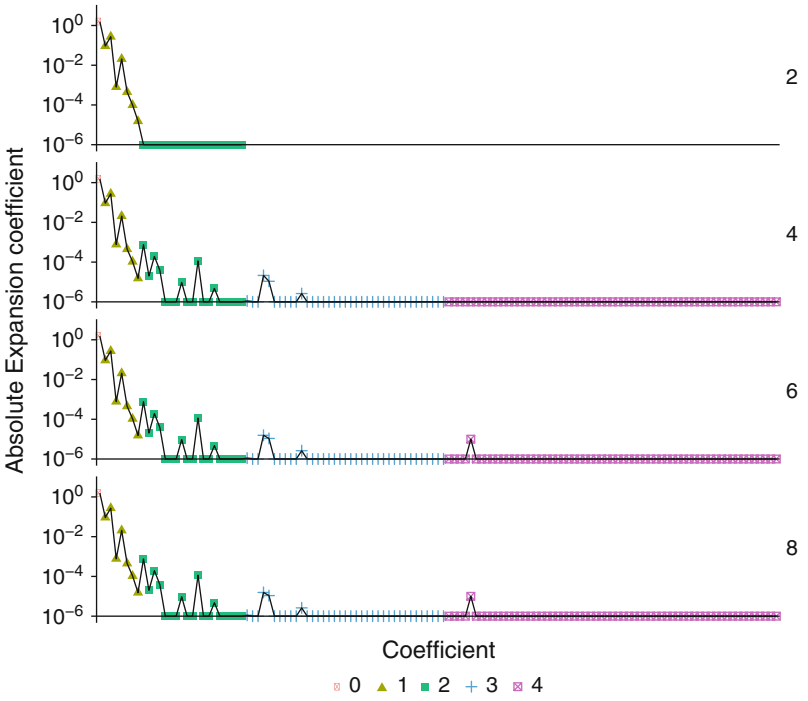

Fig. 9.17 The magnitude of the coefficients in the expansion of the value of a call option as a function of three uncertain parameters, $r = 0.0048x$ with $x \sim \mathcal{G}(0, 1)$, $Q \sim \mathcal{U}(0.025, 0.045)$, and $\Sigma \sim \mathcal{G}(5.46636, 41.8142)$. The color and shape of the points indicate the maximum polynomial degree that the coefficient responds to, e.g., c_{011} would be a "1" in the figure. The different panels on the figure indicate the number, n, of Gauss quadrature points in each dimension. Those points with a maximum polynomial degree greater than n are not shown, and the coefficients are "floored" to a minimum of 10^{-6}

Table 9.20 The convergence of the variance in the option price as a function of the quadrature rule used

n	Var(P)
2	0.486085
4	0.486321
6	0.486321
8	0.486321

The results for this calculation using the expansion coefficients in Fig. 9.17 are shown in Table 9.20. This table indicates that the $n = 2$ coefficients estimate the variance to three digits of accuracy.

We compare random 10^6 samples from the Black-Scholes solution to the same number of samples to the expansions as estimated with the various quadrature rules, in Fig. 9.18. This figure indicates that, because of the smoothness of the underlying function, an expansion with only a few terms is accurate.

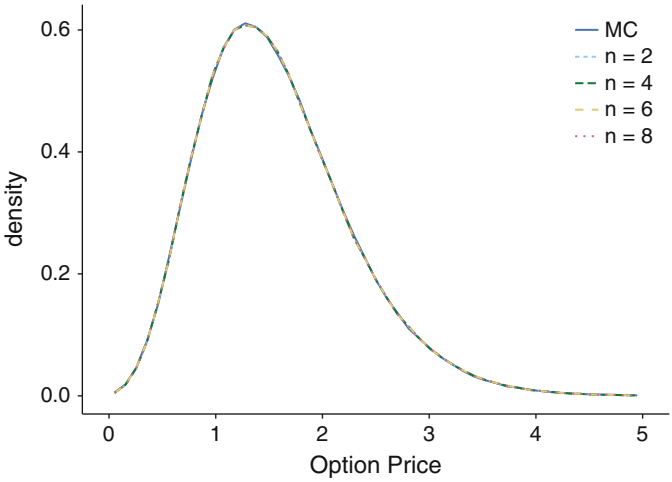

Fig. 9.18 The distribution of the price of an option with a strike price of \$44, a stock price of \$44.15, and days to expiration of 158. The risk-free interest rate is $r = 0.0048x$ with $x \sim \mathcal{G}(0, 1)$, the dividend rate is $Q \sim \mathcal{U}(0.025, 0.045)$, and the volatility of the stock is $\Sigma \sim \mathcal{G}(5.46636, 41.8142)$. We compare the polynomial chaos expansion as computed using tensor-products of quadrature rules with $n = 2, 4, 6, 8$ and compare these distributions to a Monte Carlo distribution with 10^6 samples

From this example, several things are evident. With a smoothly varying function, the expansion order required to estimate the distribution of the quantity of interest and the number of function evaluations needed are small. The results also indicate that of the many coefficients possible in a high-order expansion, most will be negligible. In the next sections, we will investigate how to take advantage of this structure.

9.5 Sparse Quadrature

The explosion of terms in multidimensional expansions comes, in part, from the cross-terms that appear in the expansion. For example, in a fourth-order expansion, we end up with $4d$ degree polynomials because the highest-order terms in the series are a product of four d-degree polynomials. The tensor-product Gauss quadratures that we use to estimate the expansion can accurately integrate these polynomials. Nevertheless, it is often the case that these high-degree interactions (that is the product of several high-degree polynomials) are unnecessary in the expansion (as we saw in the previous case).

In such a scenario, it can be useful to change the way that we expand the output in orthogonal polynomials. Instead of including combinations of polynomials up to a given degree, we look to include only polynomials up to a maximum degree. In other words

$$g(\theta_1 \ldots, \theta_d) \approx \sum_{l_1+\cdots+l_d<N} c_{l_1,\ldots,l_d} \mathfrak{P}_{l_1,\ldots,l_d}(\theta_1, \ldots, \theta_d), \tag{9.94}$$

With this expansion, we no longer need to integrate any polynomials of degree higher than N. Therefore, our tensor-product quadrature rule integrates higher-degree polynomials than we need.

For this situation we can use Smolyak sparse quadrature sets. These rules construct quadrature points that do not grow as fast as product quadrature grids. To accomplish this, we combine quadrature rules to ensure that a polynomial of a given degree in any single dimension, but not products of polynomials of that degree, is exactly integrated.

We have all the pieces we need to define a Smolyak sparse quadrature set. For a given value of ℓ, in d dimensions the quadrature rule is defined as

$$S_\ell^{(d)} f = \sum_{q=\ell-d}^{\ell-1} (-1)^{\ell-1-q} \binom{d-1}{\ell-1-q} \sum_{\|\mathbf{k}\|_1=q+d} Q_{2^{k_1}-1} \otimes \cdots \otimes Q_{2^{k_d}-1} f, \tag{9.95}$$

where $\|\mathbf{k}\|_1 = \sum_{i=1}^d |k_i|$. The term in parenthesis is $(d-1)$ choose $(\ell-1-q)$.

Looking at this formula, we see that the tensor products where the sum of the number of points in each dimension equals a constant are included. Note that the quadrature rule can have negative weights.

To demonstrate how these rules work, we will look at the quadrature rule with $\ell = 3$ and Gauss-Legendre quadrature. In this case we should have the a quadrature rule with up to $2^3 - 1$ points:

$$\begin{aligned} S_3^{(2)} f &= \sum_{q=1}^{2} (-1)^{2-q} \binom{1}{2-q} \sum_{\|\mathbf{k}\|_1=q+2} Q_{2^{k_1}-1} \otimes Q_{2^{k_2}-1} f \\ &= -\sum_{\|\mathbf{k}\|_1=3} Q_{2^{k_1}-1} \otimes Q_{2^{k_2}-1} f + \sum_{\|\mathbf{k}\|_1=4} Q_{2^{k_1}-1} \otimes Q_{2^{k_2}-1} f \\ &= -(Q_1 \otimes Q_3) f - (Q_3 \otimes Q_1) f + (Q_3 \otimes Q_3) f + (Q_1 \otimes Q_7) f \\ &\quad + (Q_7 \otimes Q_1) f \end{aligned}$$

Counting up the total number of points in this rule, there are[8] 21 compared to 49 for the tensor product quadrature rule for $Q_7 \otimes Q_7$.

We show the points for $S_3^{(2)}$ based on Gauss-Legendre quadrature in Fig. 9.19 as well as the comparable tensor-product quadrature rule, $Q_7 \otimes Q_7$.

[8] The $(Q_1 \otimes Q_3)$ and $(Q_3 \otimes Q_1)$ rules are completely redundant with $(Q_3 \otimes Q_3)$ and the $(Q_1 \otimes Q_7)$ and $(Q_3 \otimes Q_1)$ rules share the origin with $(Q_3 \otimes Q_3)$.

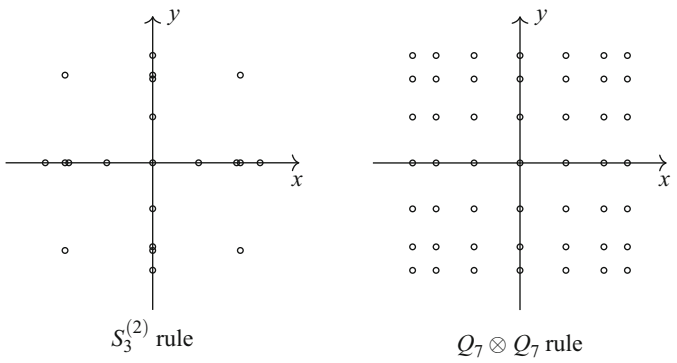

Fig. 9.19 Comparison of the Smolyak sparse quadrature rule of level $\ell = 3$ and the tensor-product rule comprised of 7-point Gauss-Legendre quadrature rules

Another way to show the construction of a 2-D Smolyak quadrature rule is to write all the quadrature rules up to order $2^\ell - 1$ in a tableau of tensor-product quadratures where the number of points in the x-direction increases from left to right and the number of points in the y-direction increases from bottom to top. The Smolyak quadrature rule will be a linear combination of the tensor-product rules from the diagonal and below. This construction is shown in Fig. 9.20.

Now that we have seen how the sparse grids work, we will discuss why they are constructed in the form that they are. As we have said, a product quadrature rule comprised of n points in 1-D will integrate d-dimensional polynomials where *any single* component polynomial has degree less than or equal to $(2n - 1)$. The Smolyak construction is designed to integrate polynomials with a total degree of equal to $(2n - 1)$. This is shown in Fig. 9.21 for $n = 2$. Indeed, it can be shown that the Smolyak sparse grid that is exact on polynomials of N in the one-dimensional quadrature rules will be exact on polynomials of total degree N for the multidimensional integral (Holtz 2011).

The construction of the quadrature set will illuminate the origin of Eq. (9.95), in particular why there needs to be negatively weighted points. Looking at Fig. 9.21, to integrate the polynomials in the triangle, we can think about it in terms of "adding" quadrature rules:

$$\begin{pmatrix} & 1 & \\ x & & y \\ x^2 & xy & y^2 \end{pmatrix} = \begin{pmatrix} & 1 & \\ x & & \\ x^2 & & \end{pmatrix} + \begin{pmatrix} 1 & & \\ & y & \\ & & y^2 \end{pmatrix} + \begin{pmatrix} & 1 & \\ x & & y \\ & xy & \end{pmatrix} - \begin{pmatrix} & 1 & \\ x & & \\ & & \end{pmatrix} - \begin{pmatrix} 1 & & \\ & y & \\ & & \end{pmatrix}. \tag{9.96}$$

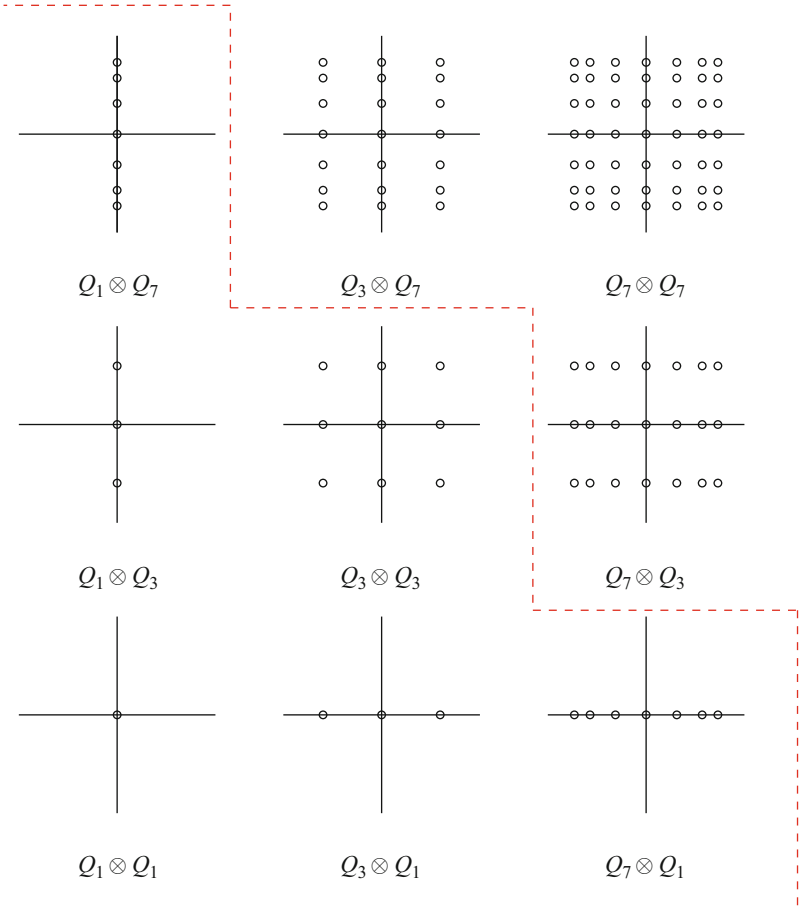

Fig. 9.20 Demonstration of the construction of the Smolyak quadrature rule with $\ell = 3$ in two dimensions comprised of Gauss-Legendre quadrature rules. The Smolyak quadrature is a linear combination of the points below the dashed line

Here we see the reason for the appearance of the term $(-1)^{\ell-1-q}$ term in Eq. (9.95). It is worth mentioning that there is an alternate form of the Smolyak rule. For this will also need to define a difference in quadratures as

$$\Delta_{2^\ell-1} f = Q_{2^\ell-1} f - Q_{2^{\ell-1}-1} f, \tag{9.97}$$

and $Q_0 = \emptyset$. Using this the Smolyak rule can be written as

$$S_\ell^{(d)} f = \sum_{q=0}^{\ell-1} \sum_{\|\mathbf{k}\|_1 = q+d} \Delta_{2^{k_1}-1} \otimes \cdots \otimes \Delta_{2^{k_d}-1} f. \tag{9.98}$$

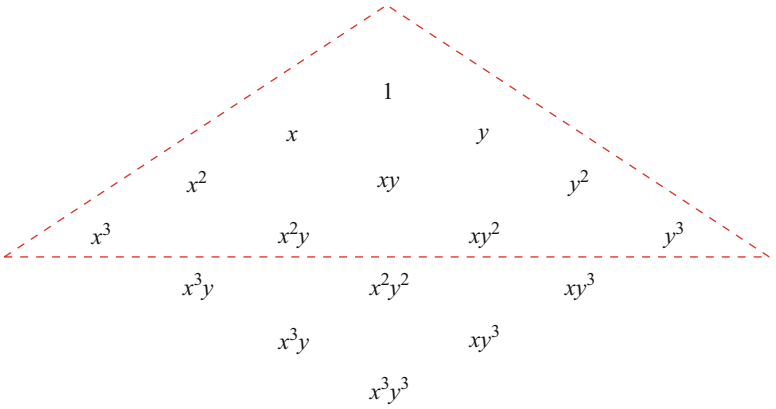

Fig. 9.21 The polynomials that can be integrated exactly by a two-dimensional tensor-product Gauss quadrature rule comprised of two-point rules. The dashed line encloses the polynomials that the sparse grid will integrate

9.5.1 Black-Scholes Example Redux

Turning back to our Black-Scholes example from before, we will construct a Smolyak sparse grid for this 3-D expansion. As we saw before, a tensor-product quadrature comprised of six-point quadrature rules was able to capture the most important coefficients in the expansion. This rule has $6^3 = 216$ function evaluations. In this case, we will use the $\ell = 3$ Smolyak sparse grid to compute the coefficients in the expansion (Holtz 2011). This quadrature rule can be calculated from

$$
\begin{aligned}
S_3^{(3)} f &= \sum_{q=0}^{2} (-1)^{2-q} \binom{2}{2-q} \sum_{\|\mathbf{k}\|_1 = q+3} Q^{(\sigma)}_{2^{k_1}-1} \otimes Q^{(x)}_{2^{k_2}-1} \otimes Q^{(z)}_{2^{k_3}-1} f \\
&= Q_1^{(\sigma)} \otimes Q_1^{(x)} \otimes Q_1^{(z)} f \\
&\quad -2\left[Q_3^{(\sigma)} \otimes Q_1^{(x)} \otimes Q_1^{(z)} f + Q_1^{(\sigma)} \otimes Q_3^{(x)} \otimes Q_1^{(z)} f \right. \\
&\quad \left. + Q_1^{(\sigma)} \otimes Q_1^{(x)} \otimes Q_3^{(z)} f \right] \\
&\quad + Q_7^{(\sigma)} \otimes Q_1^{(x)} \otimes Q_1^{(z)} f + Q_1^{(\sigma)} \otimes Q_7^{(x)} \otimes Q_1^{(z)} f \\
&\quad + Q_1^{(\sigma)} \otimes Q_1^{(x)} \otimes Q_7^{(z)} f + Q_3^{(\sigma)} \otimes Q_3^{(x)} \otimes Q_1^{(z)} f \\
&\quad + Q_3^{(\sigma)} \otimes Q_1^{(x)} \otimes Q_3^{(z)} f + Q_1^{(\sigma)} \otimes Q_3^{(x)} \otimes Q_3^{(z)} f.
\end{aligned}
$$

The component 1-D rules in $S_3^{(3)}$ are shown in Table 9.21. Note that only the z points are nested at all (notice the repeated 0).

Table 9.21 The 1-D quadrature rules that comprise the sparse rule $S_3^{(3)}$

	$\beta\sigma$	w_σ	x	w_x	z	w_z
Q_1	6.466360	271.060701	1.000000	1.000000	0.000000	2.000000
Q_3	13.811184	13.236834	6.289945	0.010389	0.774597	0.555556
	7.787369	148.010162	2.294280	0.278518	0.000000	0.888889
	3.800528	109.813705	0.415775	0.711093	−0.774597	0.555556
Q_7	28.226889	0.000454	19.395728	0.000000	0.949108	0.129485
	20.399826	0.129138	12.734180	0.000016	0.741531	0.279705
	14.769642	4.663395	8.182153	0.001074	0.405845	0.381830
	10.417345	42.165053	4.900353	0.020634	0.000000	0.417959
	6.984121	116.015439	2.567877	0.147126	−0.405845	0.381830
	4.281556	93.279531	1.026665	0.421831	−0.741531	0.279705
	2.185142	14.807693	0.193044	0.409319	−0.949108	0.129485

The nesting of points in the z direction leads to seven redundant points and a total of 50 unique points in the $S_3^{(3)}$ set. The points in the set are shown in Fig. 9.22 (compare these to the full tensor product rule in Fig. 9.23).

Using the sparse quadrature, we look at calculating the expansion coefficients for the Black-Scholes example in Fig. 9.24. In these results we see that the $l = 2$ rule exactly integrates the 1-D polynomials up to order 2. The $l = 3$ rule is accurate for the univariate polynomials up to degree 4. The mixed degree polynomials are less accurate at $l = 3$, as observed in the polynomials with maximum order 1–3. At $l = 4$ the coefficients are as accurate as the tensor-product quadrature set.

9.5.2 *Extensions to Sparse Quadratures*

The Smolyak sparse quadrature addresses the problem of the number of quadrature points growing exponentially with the number of dimensions and leads to polynomial growth in the number of quadrature points. It does not, however, address the issue regarding how many points will be needed in any single dimension. In fact, our Black-Scholes example indicates that the volatility variable should require more points than the other two. One way to accomplish this is to use an anisotropic quadrature.

Anisotropic quadratures are a way to handle integrals that require more accuracy in a given dimension. A simple way of doing this is to introduce a weight into the selection of quadrature rules. This makes Eq. (9.95)

$$S_{\ell,\mathbf{a}}^{(d)} f = \sum_{q=\ell-d}^{\ell-1} (-1)^{\ell-1-q} \binom{d-1}{\ell-1-q} \sum_{q+d-1<\|\mathbf{k}\|_{\mathbf{a}}\le q+d} Q_{2^{k_1}-1} \otimes \cdots \otimes Q_{2^{k_d}-1} f, \tag{9.99}$$

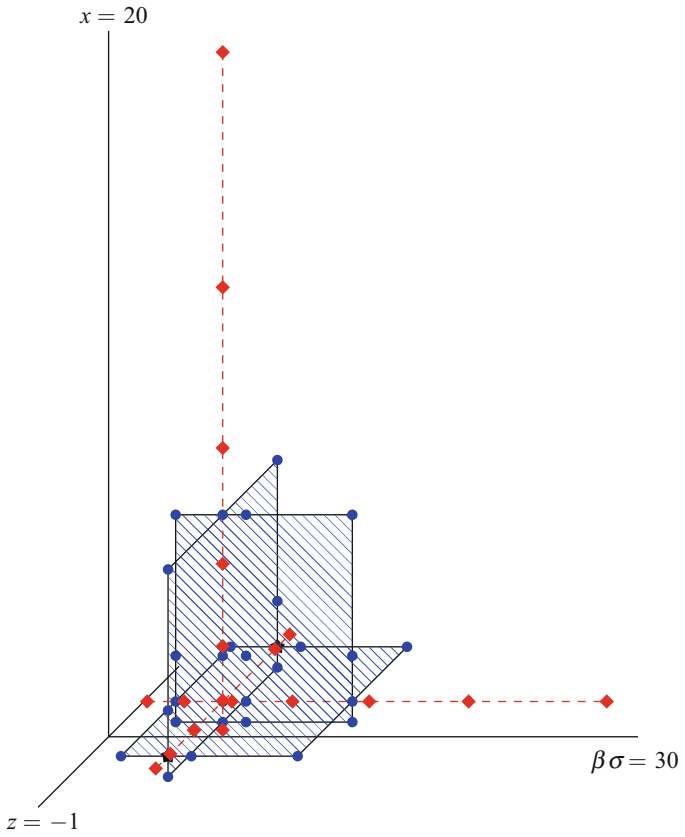

Fig. 9.22 Depiction of the points for the $S_3^{(3)}$ quadrature set comprised of the rules in Table 9.21. The diamond-shaped points are the Q_7 rules in each dimension, and the points and planes are those from the three permutations of the rule $Q_3 \otimes Q_3 \otimes Q_1$. The star-shaped points are the two nonredundant points from permutations of the $Q_3 \otimes Q_1 \otimes Q_1$ rules

where $\mathbf{a}$ is a d-length vector of weights, and $\|\mathbf{k}\|_{\mathbf{a}} = \sum_{i=1}^{d} |a_i k_i|$.

As an example, if $\mathbf{a} = (1, 0.5)$, then the $\ell = 3$ quadrature rule with $d = 2$ would be

$$\begin{aligned} S^{(d)}_{\ell,(1,0.5)} f \; = & -Q_1 \otimes Q_7 f - Q_1 \otimes Q_{15} f - Q_3 \otimes Q_3 f - Q_3 \otimes Q_1 f \\ & + Q_3 \otimes Q_{15} f + Q_3 \otimes Q_7 f + Q_1 \otimes Q_{31} f \\ & + Q_7 \otimes Q_3 f + Q_7 \otimes Q_1 f \end{aligned} \tag{9.100}$$

This rule has a maximum of 31 points in one direction and 7 in the other dimension.

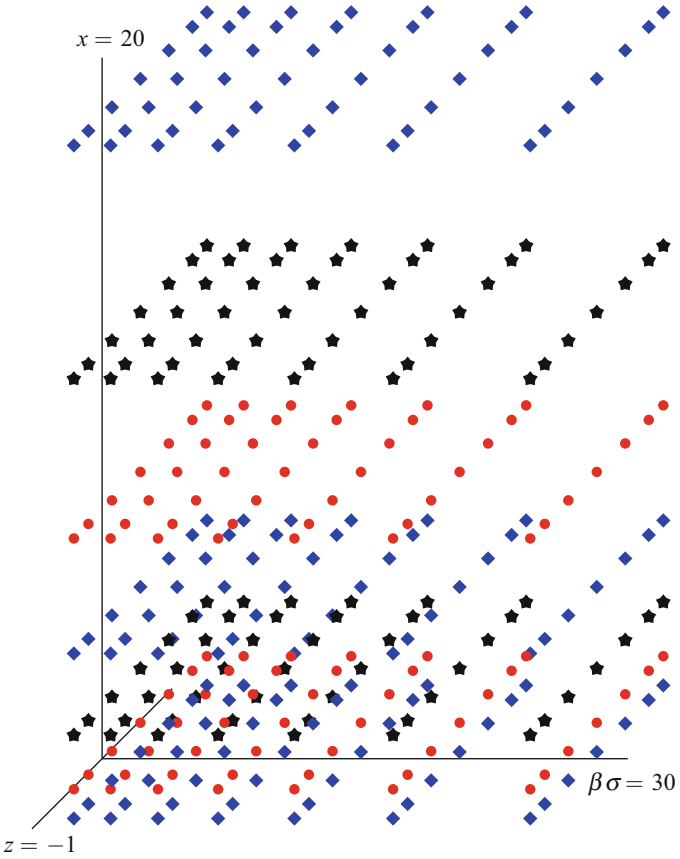

Fig. 9.23 The points from the $Q_7 \otimes Q_7 \otimes Q_7$ tensor-product quadrature using the points from Table 9.21. The different x levels are colored to distinguish them in the 2-D projection

Another possible extension is to make the quadrature adaptive in each dimension to try and automatically determine which direction to add more points in. In such a procedure, we would compute a quadrature rule as

$$A^d f = \sum_{\mathbf{k} \in I} \Delta_{2^{k_1}-1} \otimes \cdots \otimes \Delta_{2^{k_d}-1} f, \tag{9.101}$$

where I is the set of all indices included in the rule. The adaptive algorithm starts with $I = \{(1, \cdots, 1)\}$. Then, we add a point to I with an additional level in the dimension with the largest value of the tensor product of Δ quadratures because the magnitude of a Δ quadrature indicates how much the integral changes when adding new points. Then an additional level in the direction of the level just added. The rule grows by considering those tensor products that are adjacent to terms already in the set.

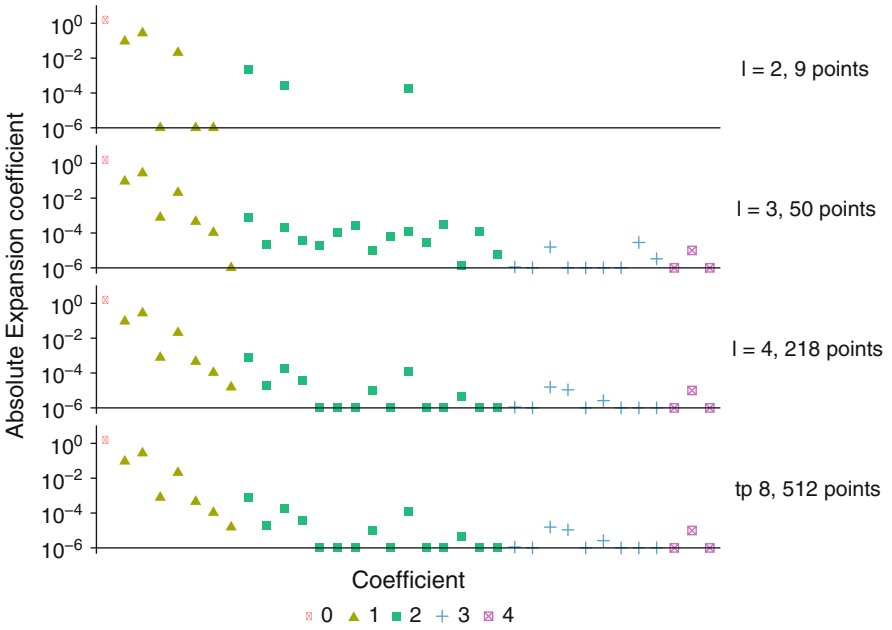

Fig. 9.24 The magnitude of the coefficients in the expansion of the value of a call option as a function of three uncertain parameters, $r = 0.0048x$ with $x \sim \mathcal{G}(0, 1)$, $Q \sim \mathcal{U}(0.025, 0.045)$, and $\Sigma \sim \mathcal{G}(5.46636, 41.8142)$. The color and shape of the points indicate the maximum polynomial degree that the coefficient responds to, e.g., c_{011} would be a "1" in the figure. The different panels on the figure indicate the level l of the sparse Gauss quadrature followed by the total number of points in the quadrature. The bottom panel shows the coefficients calculated with a tensor-product quadrature rule with 8 points in the 1-D quadrature rules. Those coefficients with a total polynomial degree greater than $(2^{\ell} - 1)/2$ are not shown, and the coefficients are "floored" to a minimum of 10^{-6}

Figure 9.25 demonstrates an example of an adaptive quadrature rule in two dimensions. We start with $\Delta_1 \otimes \Delta_1$ being the only member of I. Then we compute $\Delta_1 \otimes \Delta_2 f$ and $\Delta_2 \otimes \Delta_1 f$; these are the hatched blocks in part (a) of Fig. 9.25. In the figure the magnitude of $\Delta_1 \otimes \Delta_2 f$ is larger, so it is added to I, and then $\Delta_1 \otimes \Delta_3 f$ is computed, c.f. part (b). Then $\Delta_2 \otimes \Delta_1 f > \Delta_1 \otimes \Delta_3 f$, so $\Delta_2 \otimes \Delta_1 f$ is added to the set in part (c). Now to proceed $\Delta_2 \otimes \Delta_2 f$ and $\Delta_3 \otimes \Delta_1 f$ are calculated. Finally, in part (d) $\Delta_3 \otimes \Delta_1 f$ is added to I, and $\Delta_4 \otimes \Delta_1 f$ is computed. This process will continue until some stopping criterion is reached, such as the maximum magnitude of the $\Delta_i \otimes \Delta_j f$ under consideration is smaller than some threshold. Once this stopping criteria is reached, all the hatched blocks computed can be included in the set because the work of including them has already been done.

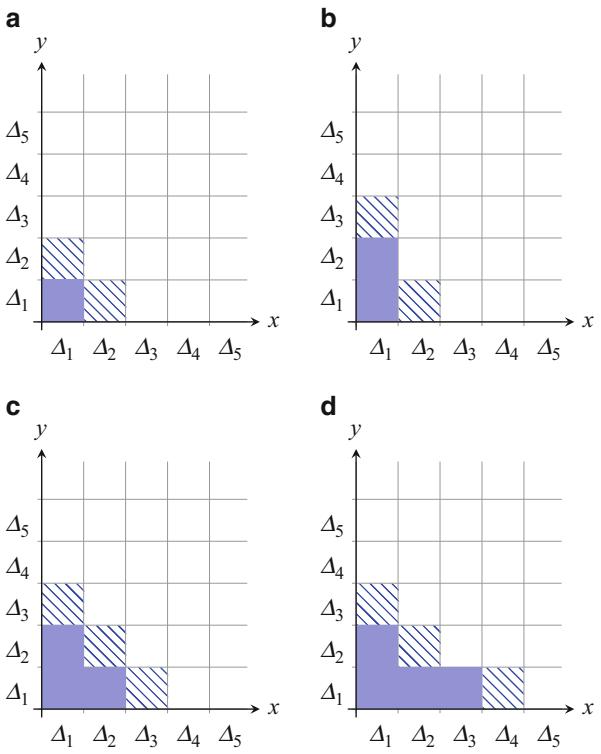

Fig. 9.25 Demonstration of an adaptive sparse quadrature rule. The set is built from left to right, and the solid blocks are the elements of the quadrature rule, and the hatched blocks are the new tensor products to be considered. The first step in the cycle is (**a**), and based on the results, the quadrature rule grows to those seen in (**b**)–(**d**)

9.6 Estimating Expansions Using Regularized Regression

The approximation of a quantity using polynomial chaos can be constructed using other means than quadrature. One possibility is to think of the expansion as a function to be estimated via regression and use regularized regression techniques to minimize the number of function evaluations needed.

To set the stage for this approach let us consider an Nth order Hermite expansion of a function $g(x)$,

$$g(x) \approx \sum_{n=0}^{N} c_n He_n(x).$$

Now consider that we have evaluated $g(x)$ at M values of x. The resulting data gives us the following system of equations

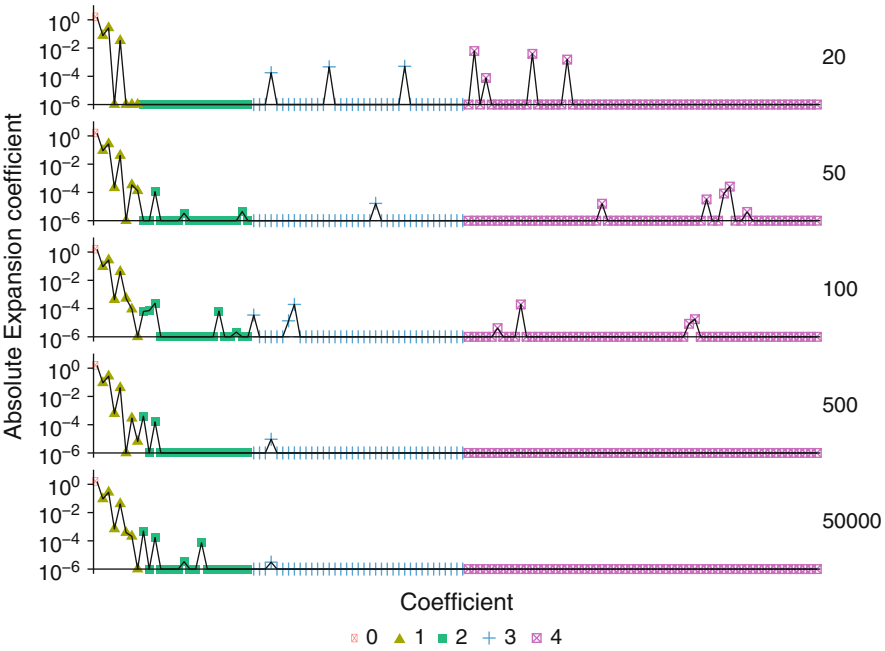

Fig. 9.26 The magnitude of the coefficients in the expansion of the value of a call option as a function of three uncertain parameters, $r = 0.0048x$ with $x \sim \mathcal{G}(0, 1)$, $Q \sim \mathcal{U}(0.025, 0.045)$, and $\Sigma \sim \mathcal{G}(5.46636, 41.8142)$ as computed by elastic net regression with $\alpha = 0.75$ and λ picked via cross-validation. The different panels on the figure indicate the number, n, of samples of the output used to construct the fits. The coefficients are "floored" to a minimum of 10^{-6}

$$
\begin{aligned}
g(x_1) &= c_0 He_0(x_1) + c_1 He_1(x_1) + \cdots + c_n He_n(x_1) + \epsilon_1, \\
g(x_2) &= c_0 He_0(x_2) + c_1 He_1(x_2) + \cdots + c_n He_n(x_2) + \epsilon_2, \\
&\vdots \\
g(x_M) &= c_0 He_0(x_M) + c_1 He_1(x_M) + \cdots + c_n He_n(x_M) + \epsilon_M.
\end{aligned}
$$

Here we have written the expansion error for each case as ϵ_i. This system is M equations for $N + 1$ unknowns, the c_n coefficients, and therefore has no unique solution unless $M = N + 1$. We can write this system using a rectangular matrix as

$$\mathbf{y} = \mathbf{A}\mathbf{c},$$

where $\mathbf{y}$ is the vector of length M that contains the $g(x_i)$, $\mathbf{A}$ is the $M \times (N + 1)$ matrix of the Hermite functions evaluated at x_i, and $\mathbf{c}$ is a vector of length $(N + 1)$ for the unknown c_i coefficients. To estimate the coefficients, we can use regularized regression, as discussed in Sect. 5.2.

As a simple test of regularized regression, we will consider the Black-Scholes example from above. In Fig. 9.26, the results using $\alpha = 0.75$ and λ picked

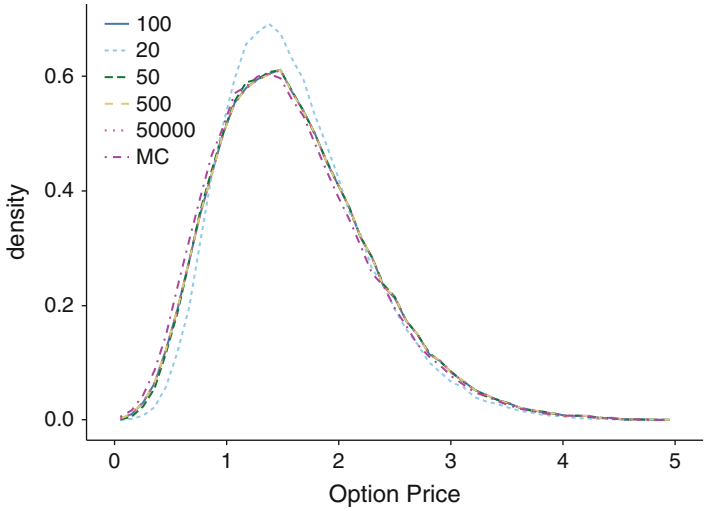

Fig. 9.27 The distribution of the price of an option with a strike price of \$44, a stock price of \$44.15, and days to expiration of 158. The risk-free interest rate is $r = 0.0048x$ with $x \sim \mathcal{G}(0, 1)$, the dividend rate is $Q \sim \mathcal{U}(0.025, 0.045)$, and the volatility of the stock is $\Sigma \sim \mathcal{G}(5.46636, 41.8142)$. We compare the distribution using a polynomial chaos expansion calculated via elastic net regression as computed using different number of function samples and compare these distributions to a Monte Carlo distribution with 10^5 samples

via a cross-validation procedure are shown. In this figure, we notice that with only 50 samples from the output function, we approximate the coefficients with magnitude larger than 10^{-4} well, and further samples do improve some of these small coefficients. Another observation is that even with 5000 function evaluations, the nonzero coefficient for the fourth degree polynomial in volatility is not captured, as it was with quadrature using fewer points. This is likely due to the nature of generating the samples of the output randomly leading to difficulty estimating parameters that are otherwise swamped by the randomness of the sampling.

Despite not capturing the low-magnitude coefficients estimated by quadrature, Fig. 9.27 shows that the distributions produced by regularized regression capture the features of the true distribution of the output, as calculated via Monte Carlo sampling of the output.

9.7 Stochastic Collocation Methods

Thus far in this chapter, we have considered projection methods where the distribution for a QoI is projected onto a polynomial subspace using quadrature. In this section we seek a polynomial representation of the QoI that exactly matches the QoI at a set of points of input space. As before, this method involves evaluating the

QoI at particular values of the uncertain inputs. It differs in that it uses the value of the QoI at those points to create an interpolating polynomial. It is this interpolating polynomial that can then be evaluated at samples of the inputs to get approximate samples of the QoI. This procedure is called stochastic collocation because it assures that the approximation to the QoI is exact at the sampled points and is analogous to collocation methods for deterministic problems.

The points that are used to evaluate the inputs can be generated by any means (e.g., random sampling, which would lead to Monte Carlo when combined with quadrature), but it is most common to use the quadrature points associated with the distribution of the uncertain inputs or their sparse constructions. Using the quadrature points allows an equivalence between the polynomial chaos projection methods and collocation if the QoI is a polynomial. To demonstrate this, consider a QoI that is a polynomial of degree d of a single input,

$$Q(x) = \sum_{i=0}^{d} c_i P_i(x),$$

where $P_i(x)$ is a degree i orthogonal polynomial. Clearly, the projection of $Q(x)$ onto the polynomials P_i will be exact, and the integrations required to carry out the projection will be integrals of polynomials of degree d^2 or less. Therefore, to compute the projection of $Q(x)$ onto the P_i will require $d + 1$ quadrature points, if Gauss quadrature is used because Gauss quadrature of n points is exact for polynomials up to degree $2n - 1$.

Furthermore, if stochastic collocation is used with $d + 1$ points defined by the quadrature points, it will yield a degree d polynomial. Projecting this polynomial onto the P_i will have the same coefficients as spectral projection because the collocation polynomial is exact at the quadrature points. Then, by uniqueness of polynomials, the collocation and projection methods must be equivalent.

The difference between the two methods occurs when the QoI is not a polynomial in the inputs. For the projection technique, the number of quadrature points can be independent of the degree of the polynomial expansion. As an example, in the previous sections, we saw examples where the coefficients in a projection onto a quartic basis required between 8 and 10 quadrature points to be exact. With stochastic interpolation, ten collocation points would construct a tenth degree polynomial. This high-degree polynomial may have issues with oscillations or other artifacts, as we will see.

A benefit of collocation is that it can deal with random variables that do not have an associated orthogonal polynomial. For example, if the input uncertainties are represented by a spline fit to an empirical histogram, it may be impossible to adequately represent it as a "standard" random variable. We could use collocation for such a variable. We may not, however, have the ability to rely on standard quadratures to choose the collocation points in such a situation. The collocation procedure would in principle work, however.

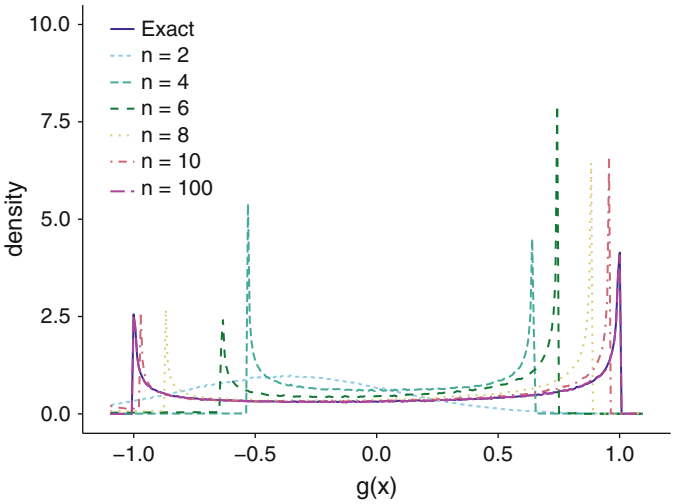

Fig. 9.28 PDF of the random variable $g(x) = \cos(x)$, where $x \sim \mathcal{N}(\mu = 0.5, \sigma^2 = 4)$ using stochastic collocation with various Gauss-Hermite quadrature points to evaluate the function for interpolation. This figure was generated from 10^6 samples of x that were used to evaluate $g(x)$ and the various approximations

To demonstrate the procedure of stochastic collocation, we consider the random variable $g(x) = \cos x$ where $x \sim \mathcal{N}(\mu = 0.5, \sigma^2 = 4)$. We will evaluate this function at different values of $x = \mu + \sigma Z$ where Z is given by the Gauss-Hermite quadrature rules as discussed in Sect. 9.1.3. For an n point quadrature rule, we will construct the interpolating polynomial using the Lagrange formula:

$$g(x) \approx \sum_{i=1}^{n} \prod_{j=1,\, i \neq j}^{n} \frac{x - x_j}{x_i - x_j} f(x_i). \tag{9.102}$$

Other polynomial constructions are possible, but will give equivalent polynomials due to uniqueness of polynomials—only one polynomial of degree $n - 1$ will pass through the n points $(x_i, f(x_i))$.

The results for stochastic collocation on this example are shown in Fig. 9.28. These results appear very similar to those in Fig. 9.3 where several different quadrature orders are used to approximate a projection onto a fifth-order Hermite polynomial. The biggest difference between the two examples is that the collocation example is converging to the exact distribution because as more points are added the degree of the interpolating polynomial must go up. With the projection technique, we fixed the number of moments required, and, as a result, the approximation stopped improving at some point, even as more quadrature points were added. The other noticeable difference is the $n = 2$ approximation has very different character in the two approaches. Using collocation we see a unimodal shape in the

Table 9.22 The convergence of the mean and variance in $g(x) = \cos(x)$, where $x \sim \mathcal{N}(\mu = 0.5, \sigma^2 = 4)$ as estimated using collocation at points defined by different Gauss-Hermite quadrature rules

n	Mean	Variance
2	−0.365375	0.189648
4	0.065227	0.228159
6	0.116899	0.324055
8	0.119714	0.431657
10	0.119851	0.475111
100	0.114505	1.411939
∞	0.118768	0.48599

These moments were estimated by evaluating the collocation approximation at 10^6 samples of x

distribution, whereas the projection results with the same number of points have a shape similar to the exact result, except shifted.

Stochastic collocation will have an error in the approximation of the moments of the QoI, just as the projection method had errors when too few quadrature points were used. To estimate the moments, we have two options: we could estimate empirical distributions by sampling from x and evaluating the collocation approximation (this is done in Table 9.22), or we could project the collocation estimate onto the Hermite polynomials up to some degree. This second approach gives the same approximation as before when the number of collocation points is sufficient to estimate the moment integrals. As the results in Table 9.22 indicate, the empirical approach can be problematic because it is possible that a sampled point will be an extrapolation for the polynomial, i.e., a sampled value x is outside the range of collocation points, and large errors are possible. This is what happened in the variance estimate for $n = 100$ collocation points: extrapolating with a high-degree polynomial is a risky idea.

The connections between collocation and projection suggest a way to combine them together. Given a fixed number of samples of the QoI that one can afford, a set of runs can be used based on the appropriate quadrature points or their sparse version. Then using those quadrature points project the QoI onto a large set of orthogonal polynomials. Using the same points to construct a collocation approximation to the QoI, one can then project the approximation onto different degree expansions in orthogonal polynomials using different quadrature rules (i.e., evaluating the interpolating polynomial at different quadrature points). These quadrature calculations do not require any additional function evaluations. Then comparing how these projections converge to the projection using all the quadrature points, one can decide what degree polynomial to use to represent the QoI.

To illustrate this procedure, we consider the QoI from above, $g(x) = \cos(x)$, where $x \sim \mathcal{N}(\mu = 0.5, \sigma^2 = 4)$. We assume we can only afford ten function evaluations and use these to project onto a ninth degree Hermite expansion (the penultimate row in Table 9.23). Then using those ten points as collocation points, we construct the interpolating polynomial to approximate $g(x)$. With this approxima-

Table 9.23 The expansion coefficients of a 9 degree Hermite expansion of the ninth degree collocation approximation to $g(x) = \cos(x)$, where $x \sim \mathcal{N}(\mu = 0.5, \sigma^2 = 4)$ as estimated using different Gauss-Hermite quadrature rules

n	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
2	−0.291235	−0.445239	−0.000000	0.148413	0.024270	−0.022262	−0.006472	0.001767	0.000953	−0.000034
3	0.239458	0.098535	−0.578011	−0.000000	0.144503	−0.004927	−0.016446	0.000938	0.000965	−0.000088
4	0.103214	−0.216895	−0.050571	0.176795	0.000000	−0.035359	0.001686	0.003558	−0.000302	−0.000215
5	0.118767	−0.114544	−0.276399	0.031698	0.140768	0.000000	−0.023461	−0.000755	0.002079	0.000122
6	0.118767	−0.129769	−0.237515	0.101766	0.059634	−0.030029	−0.000000	0.004290	−0.001065	−0.000381
7	0.118767	−0.129769	−0.237515	0.086541	0.079075	−0.012053	−0.014931	−0.000000	0.001866	0.000167
8	0.118767	−0.129769	−0.237515	0.086541	0.079075	−0.017382	−0.010394	0.002742	0.000000	−0.000305
9	0.118767	−0.129769	−0.237515	0.086541	0.079075	−0.017382	−0.010394	0.001727	0.000648	−0.000000
10	0.118767	−0.129769	−0.237515	0.086541	0.079075	−0.017382	−0.010394	0.001727	0.000648	−0.000127
∞	0.118768	−0.129766	−0.237536	0.086511	0.079179	−0.017302	−0.010557	0.001648	0.000754	−0.000092

The values of c_n for $n < 7$ agree well with the exact values *and* appear to be converged based on the different quadrature rules

tion, we then apply Gauss-Hermite quadrature of different orders to the interpolating polynomial to compute different projections. This procedure allows one to have confidence in the degree selected for the projection. As seen in Table 9.23), the coefficient c_9 is clearly not converged, and we should not trust its value as estimated using 10 quadrature points. Similarly, the values for c_7 and c_8 agree only at $n = 9$ and $n = 10$. Compared with the exact value of the coefficients, these are off by 5 and 15%, respectively. When we look at c_6 and below, the estimates for the coefficients seem to have converged. These results suggest that for this case a projection onto a sixth degree Hermite polynomial expansion is the best one can do with ten function evaluations.

This example suggests that combining polynomial chaos projection with stochastic collocation can lead to more confidence in the results. However, we cannot discount the fact that there are artifacts in the distribution that are missed by our approximations. Alas, certainty is an illusion when only a finite number of function evaluations are possible.

The examples above did not go into detail with multidimensional interpolation. Interpolating polynomials in multiple dimensions is possible, though the formulas get tedious. There are libraries to handle such interpolation in 2-D in most coding platforms. For general dimensions, the best tool for automated construction may be Mathematica's `InterpolatingPolynomial` function.

9.8 Stochastic Finite Elements

Another approach to applying projections onto orthogonal polynomials is the stochastic finite element method (SFEM). In this approach we begin with the original PDEs and write a polynomial approximation to the solution. Then using projection techniques, we can get information about the variability of the solution to the PDE as a random process.

To demonstrate the procedure, we begin with a general time-dependent partial differential equation, for a quantity $u(\mathbf{z}, t; \mathbf{x})$, where $\mathbf{z}$ is spatial dependence, t is a time variable, and $\mathbf{x}$ is a vector of p random variables; note that u is now a random process because it is a random function. We write the PDE as

$$F(u, \dot{u}, \mathbf{x}) = 0; \qquad \dot{u} = \frac{\partial u}{\partial t}. \tag{9.103}$$

The function F also depends on $\mathbf{x}$ independent of u.

We then write u as a truncated polynomial chaos expansion (c.f. Eq. (9.79)) as

$$\hat{u}(\mathbf{z}, t; \mathbf{x}) \approx \sum_{\ell_1=0}^{N_1} \cdots \sum_{\ell_p=0}^{N_p} u_{\ell_1,\ldots,\ell_p}(\mathbf{z}, t)\mathfrak{P}_{\ell_1,\ldots,\ell_p}(\mathbf{x}), \tag{9.104}$$

where, as before, $\mathfrak{P}_{\ell_1,\ldots,\ell_p}(\mathbf{x})$ is the product of p orthogonal polynomials

$$\mathfrak{P}_{\ell_1,\ldots,\ell_p}(\mathbf{x}) = \prod_{i=1}^{p} P_{\ell_i}(x_i).$$

The coefficient functions are defined by a projection onto the polynomial

$$u_{\ell_1,\ldots,\ell_p}(\mathbf{z},t) = \int_{D_1} dx_1 \cdots \int_{D_p} dx_p \, \mathfrak{P}_{\ell_1,\ldots,\ell_p}(\mathbf{x}) c^{-1}_{\ell_1,\ldots,\ell_p} u(\mathbf{z},t;\mathbf{x}). \tag{9.105}$$

where $c_{\ell_1,\ldots,\ell_p}$ is a normalization constant

$$c^{-1}_{\ell_1,\ldots,\ell_p} = \int_{D_1} dx_1 \cdots \int_{D_p} dx_p \, \mathfrak{P}_{\ell_1,\ldots,\ell_p}(\mathbf{x})^2.$$

Using the expansion in Eq. (9.104), we will attempt to determine the coefficients $u_{\ell_1,\ldots,\ell_p}(\mathbf{z},t)$ using the method of weighted residuals. That is, we use the expansion in Eq. (9.103), multiply the result by a weight function $w_{\ell_1,\ldots,\ell_d}(\mathbf{x})$, and integrate over the domain of each of the p random variables. The resulting system of equations is

$$\int_{D_1} dx_1 \cdots \int_{D_p} dx_p \, w_{\ell_1,\ldots,\ell_d}(\mathbf{x}) F(\hat{u},\dot{\hat{u}},\mathbf{x}) = 0. \tag{9.106}$$

If we use the $\mathfrak{P}_{\ell_1,\ldots,\ell_d}(\mathbf{x})$ as the weighting function, this is known as Galerkin weighting. The result will be a system of coupled partial differential equations for the $u_{\ell_1,\ldots,\ell_p}(\mathbf{z},t)$ functions. The benefit of this approach is that if the polynomials are chosen such that $\mathfrak{P}_{0,\ldots,0}$ is the PDF of the joint distribution of $\mathbf{x}$ assuming each input is independent, then the function $u_{0,\ldots,0}(\mathbf{z},t)$ will be the mean of the random process $u(z,t;\mathbf{x})$. The variance can be written in terms of the sum of squares of the expansion functions, as in the projection methods already discussed.

To demonstrate this technique, we use the 2-D Poisson's equation with an uncertain source that we have seen in Sect. 9.2.7. The problem we are interested in solving is

$$-\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u(x,y;\tau) = q(x,y;\tau). \tag{9.107}$$

$$u(1,y;\tau) = u(x,1;\tau) = u(-1,y;\tau) = u(x,-1;\tau) = 0. \tag{9.108}$$

The source q will be a Gaussian in space with an uncertain center in y:

$$q(x,y;\tau) = \exp\left[-x^2 - (y-\tau)^2\right], \qquad \tau \sim \mathscr{U}(-0.25, 0.25). \tag{9.109}$$

Table 9.24 Expansion of source in Legendre polynomials

n	$q_n(x, y)$
0	$e^{-x^2}\sqrt{\pi}\left(\text{erf}\left(\frac{1}{4}-y\right)+\text{erf}\left(y+\frac{1}{4}\right)\right)$
1	$12e^{-x^2}\left(\sqrt{\pi}y\left(\text{erf}\left(\frac{1}{4}-y\right)+\text{erf}\left(y+\frac{1}{4}\right)\right)-e^{-\frac{1}{16}(4y+1)^2}\left(-1+e^y\right)\right)$
2	$\frac{5}{2}e^{-x^2}\left(\sqrt{\pi}\left(48y^2+23\right)\left(\text{erf}\left(\frac{1}{4}-y\right)+\text{erf}\left(y+\frac{1}{4}\right)\right)$ $+-12e^{-\frac{1}{16}(4y+1)^2}\left(-4y+e^y(4y+1)+1\right)\Big)$
3	$14e^{-x^2}\left(e^{-\frac{1}{16}(4y+1)^2}\left(20y(4y-1)-2e^y(10y(4y+1)+41)+82\right)\right.$ $\left.+\sqrt{\pi}y\left(80y^2+117\right)\left(\text{erf}\left(\frac{1}{4}-y\right)+\text{erf}\left(y+\frac{1}{4}\right)\right)\right)$

In the notation of Eq. (9.103) for this problem, we have $\mathbf{z} = (x, y)$, $\mathbf{x} = \tau$, and

$$F(u, \dot{u}; \mathbf{x}) = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u(x, y; \tau) - q(x, y; \tau) = 0.$$

Given that we have a single, uniform random variable, we expand $u(x, y; \tau)$ in terms of Legendre polynomials. For this example we choose a cubic expansion:

$$\hat{u}(x, y; \tau) \approx u_0(x, y)P_0(\theta) + u_1(x, y)P_1(\theta) + u_2(x, y)P_2(\theta) + u_3(x, y)P_3(\theta), \tag{9.110}$$

where, for simplicity, we have written $\theta = 4\tau$. From this definition we note that

$$\int_{-1}^{1} \hat{u}(x, y; \theta)P_n(\theta)\, d\theta = u_n(x, y), \qquad 0 \le n \le 3.$$

Additionally, we will write the source q as a Legendre expansion

$$q(x, y; \theta) \approx q_0(x, y)P_0(\theta) + q_1(x, y)P_1(\theta) + q_2(x, y)P_2(\theta) + q_3(x, y)P_3(\theta).$$

The coefficient functions $q_n(x, y)$ are given in Table 9.24. Using these results, we insert $\hat{u}$ into $F(u, \hat{u}; \mathbf{x}) = 0$, multiply by a Legendre polynomial, and integrate to get the four equations:

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u_n(x, y) = q_n(x, y), \qquad 0 \le n \le 3. \tag{9.111}$$

These are four uncoupled partial differential equations for the $u_n(x, y)$. The fact that they are uncoupled is due to the fact that the uncertainty was in the source term to the equation.

We can also see that our expansion for $\hat{u}$ in Eq. (9.110) gives us the mean of the random process $u(x, y; \tau)$ as

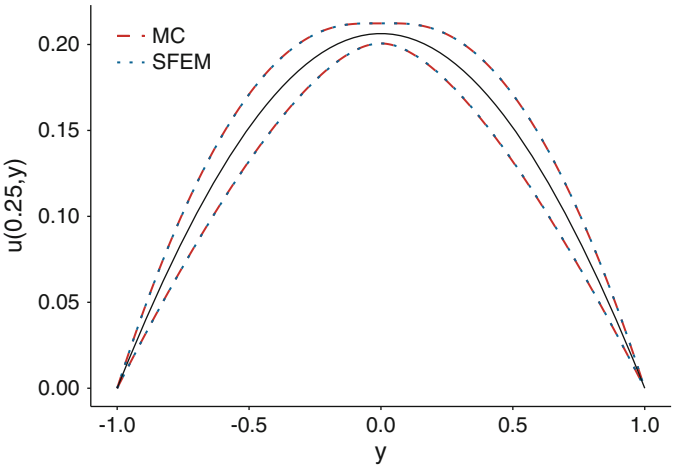

Fig. 9.29 The mean (solid line) and ±2 standard deviation for the function $u(0.25, y; \tau)$ from the Poisson's equation with uncertain source as approximated by the stochastic finite element method with a cubic Legendre expansion and Monte Carlo with 10^4 samples of τ. The mean is coincident on the scale of the plot for both methods

$$2\int_{-1/4}^{1/4} \hat{u}(x, y; \tau)\, d\tau = \frac{1}{2}\int_{-1}^{1} \hat{u}(x, y; \theta)\, d\theta$$
$$= u_0(x, y).$$

Similarly, the variance in the random process is

$$\frac{1}{2}\int_{-1}^{1} P_0(\theta)(\hat{u}(x, y; \theta))^2\, d\theta - (u_0(x, y))^2 \approx \sum_{n=1}^{3} \frac{u_n(x, y)^2}{2n+1}. \tag{9.112}$$

By performing the polynomial chaos expansion and using the Galerkin-weighted residual to find the coefficient functions in the expansion, we have gone from a single PDE to four. The fact that the equations are uncoupled is helpful, this means we can solve these equations separately to get an approximation to $u(x, y; \tau)$. To demonstrate this solution technique, we solve the four PDEs using Mathematica's `NDSolve` function. We then compute the variance as a function of space using Eq. (9.112) and compare it to the result from Monte Carlo sampling of τ with 10^4 points. The results are shown for $x = 0.25$ in Fig. 9.29. In this plot we see that the Monte Carlo and SFEM results are nearly identical. However, the SFEM results required the solution to four PDEs where the Monte Carlo result required solving the PDE thousands of times. Also, we note that the variance goes to zero at the boundary of the domain because there was no randomness in the boundary conditions (the solution always went to zero).

The SFEM method with Galerkin weighted residuals gives equations that are more difficult to solve when products of random variables occur in the equations. This was not the case in the above example. We consider the steady ADR equation for a quantity $u(z;\mathbf{x})$ where $\mathbf{x} = (x_1, x_2)$ as

$$v(x_1)\frac{du}{dz} - \omega\frac{d^2u}{dz^2} + \kappa(x_2)u = qz(10-z), \qquad u(0;\mathbf{x}) = u(10;\mathbf{x}) = 0, \tag{9.113}$$

where

$$v(x_1) = 10 + x_1 \qquad x_1 \sim \mathcal{N}(0,1),$$

$$\kappa(x_2) = \begin{cases} 0.1 + 0.01x_2 & 5 \le z \le 7.5 \\ 1 + 0.1x_2 & \text{otherwise} \end{cases} \equiv \kappa_0(z) + \kappa_1(z)x_2, \qquad x_2 \sim \mathcal{N}(0,1).$$

We will use linear Hermite expansions to approximate $u(z,\mathbf{x})$ as

$$\hat{u}(z,\mathbf{x}) = u_0(z) + u_{10}(z)x_1 + u_{01}(z)x_2;$$

v and κ are already expressed as expansions in Hermite polynomials. To determine the coefficients, we will need to integrate Eq. (9.113) times a weight function and Hermite polynomials where u is replaced by $\hat{u}$. We will compute these integrals term by term. We start with the diffusion term:

$$\begin{aligned}
&\int_{-\infty}^{\infty} dx_1 \int_{-\infty}^{\infty} dx_2 \frac{e^{-x_1^2/2 - x_2^2/2}}{2\pi n!m!} He_n(x_1) He_m(x_2)\omega \\
&\quad\times \frac{d^2}{dz^2}(u_0(z) + u_{10}(z)x_1 + u_{01}(z)x_2) \\
&\qquad= \omega\frac{d^2}{dz^2}\int_{-\infty}^{\infty} dx_1 \int_{-\infty}^{\infty} dx_2 e^{-x_1^2/2} He_n(x_1) e^{-x_2^2/2} He_m(x_2) \\
&\qquad\quad\times (u_0(z) + u_{10}(z)x_1 + u_{01}(z)x_2) \\
&\qquad= \omega\frac{d^2}{dz^2}\left(\delta_{m0}\delta_{n0}u_0(z) + \delta_{n1}\delta_{m0}u_{10}(z) + \delta_{n0}\delta_{m1}u_{01}(z)\right).
\end{aligned} \tag{9.114}$$

Similarly, the source term integrates to

$$\int_{-\infty}^{\infty} dx_1 \int_{-\infty}^{\infty} dx_2 \frac{e^{-x_1^2/2 - x_2^2/2}}{2\pi n!m!} He_n(x_1) He_m(x_2) qz(10-z) = \delta_{m0}\delta_{n0} qz(10-z). \tag{9.115}$$

The terms with v and κ are a bit trickier. To whit, the integrals of the κu terms are

$$\int_{-\infty}^{\infty} dx_1 \int_{-\infty}^{\infty} dx_2 \frac{e^{-x_1^2/2-x_2^2/2}}{2\pi n!m!} He_n(x_1)He_m(x_2)(\kappa_0 z + \kappa_1(z)x_2)$$
$$\times(u_0(z) + u_{10}(z)x_1 + u_{01}(z)x_2)$$
$$= \begin{cases} \kappa_0(z)u_0(z) + \kappa_1(z)u_{01}(z) & n = 0 \quad \& \quad m = 0 \\ \kappa_1(z)u_0(z) + \kappa_0(z)u_{01}(z) & n = 0 \quad \& \quad m = 1 \\ \kappa_0(z)u_{10}(z) & n = 1 \quad \& \quad m = 0 \\ \kappa_1(z)u_{10}(z) & n = 1 \quad \& \quad m = 1 \\ 0 & \text{otherwise} \end{cases}. \tag{9.116}$$

Note that this term couples the uncertainties in the two different variables: κ depends on x_2, but u_{10} is the x_1 dependence of u. This comes about from the product of the two variables in $\kappa\hat{u}$. The advection term has similar coupling

$$\frac{d}{dz}\int_{-\infty}^{\infty} dx_1 \int_{-\infty}^{\infty} dx_2 \frac{e^{-x_1^2/2-x_2^2/2}}{2\pi n!m!} He_n(x_1)He_m(x_2)(10 + x_1)$$
$$\times(u_0(z) + u_{10}(z)x_1 + u_{01}(z)x_2)$$
$$= \frac{d}{dz}\begin{cases} 10u_0(z) + u_{10}(z) & n = 0 \quad \& \quad m = 0 \\ u_0(z) + 10u_{10}(z) & n = 1 \quad \& \quad m = 0 \\ 10u_{01}(z) & n = 0 \quad \& \quad m = 1 \\ u_{01}(z) & n = 1 \quad \& \quad m = 1 \\ 0 & \text{otherwise} \end{cases}. \tag{9.117}$$

Putting this all together, we get that the projection of Eq. (9.113) onto $He_0(x_1)He_0(x_2)$ is

$$\left(10\frac{d}{dz} - \omega\frac{d^2u}{dz^2} + \kappa_0(z)\right)u_0(z) + \frac{du_{10}}{dz} + \kappa_1(z)u_{01}(z) = qz(10-z). \tag{9.118a}$$

Notice that this equation is the equation assuming the mean values of v and $\kappa(z)$ with the additional coupling terms to the linear terms in both random variables. Continuing on, the projection onto $He_1(x_1)He_0(x_2)$ is

$$\left(10\frac{d}{dz} - \omega\frac{d^2u}{dz^2} + \kappa_0(z)\right)u_{10}(z) + \frac{du_0(z)}{dz} = 0. \tag{9.118b}$$

Finally, the projection onto $He_0(x_1)He_1(x_2)$ is

$$\left(10\frac{d}{dz} - \omega\frac{d^2u}{dz^2} + \kappa_0(z)\right)u_{01}(z) + \kappa_1(z)u_0(z) = 0. \tag{9.118c}$$

A couple of observations are in order. Firstly, the resulting equations form a coupled system of PDEs. This means that we cannot simply use a code that solves the ADR equation to compute the expansion of $\hat{u}$. This means that we need to develop a new code to solve the coupled system. This is an example of an *intrusive* UQ method because the solution procedure for the underlying models is different than that for solving the original equations without uncertainty. For a simple example, like ADR, this is not a large inconvenience, but for existing production software, changing the code base is likely to be a nonstarter.

Additionally, we only had two variables and a linear expansion and is likely to be inadequate for many problems. Had there been many more random variables (as is common in practice), we would have had a large system of coupled equations (remember the number of terms in the projection grows geometrically). Therefore, we are limited in applying Galerkin SFEM to problems with a small number of random variables and moderate expansion orders. As discussed previously, screening out unimportant uncertainties is requisite to the application of this method.

9.8.1 SFEM Collocation

There is a modification to the SFEM procedure that can simplify the application of the method and allow it to be nonintrusive in many cases. Rather than computing the expansion coefficients using Eq. (9.106) where the weight function is an orthogonal polynomial and the expansion of u is an orthogonal polynomial expansion of the random process $\hat{u}$, we evaluate the random process at particular values of $\mathbf{x}$ and then use interpolation to get the value of the random process at other values of the random variable. As we will see, this method does not require the solution of coupled partial differential equations.

Consider the generic system from Eq. (9.103). Given a definition for the random variables $\mathbf{x}$, we then choose points to evaluate the function at based on the quadrature rule appropriate for each random variable or the sparse version of the quadratures. This will give us a set of decoupled partial differential equations to solve. These can then be solved independently and polynomial interpolation can be used to produce a representation of the full random process in a similar manner as that done for a single QoI in Sect. 9.7.

On the advection-diffusion-reaction problem solved in the previous section, defined by Eq. (9.113), the application of collocation would be as follows. Given that there are two normal random variables, we would evaluate x_1 and x_2 at the four points $x_i = \pm 2^{-1/2}$ as given in Sect. 9.1.3. The resulting equations are

$$\frac{1}{\sqrt{2}}\frac{du_1}{dz} - \omega\frac{d^2u_1}{dz^2} + \kappa\left(\frac{1}{\sqrt{2}}\right)u_1 = qz(10-z), \tag{9.119a}$$

$$-\frac{1}{\sqrt{2}}\frac{du_2}{dz} - \omega\frac{d^2u_2}{dz^2} + \kappa\left(\frac{1}{\sqrt{2}}\right)u_2 = qz(10-z), \tag{9.119b}$$

$$-\frac{1}{\sqrt{2}}\frac{du_3}{dz}-\omega\frac{d^2u_3}{dz^2}+\kappa\left(-\frac{1}{\sqrt{2}}\right)u_3=qz(10-z), \tag{9.119c}$$

and

$$\frac{1}{\sqrt{2}}\frac{du_4}{dz}-\omega\frac{d^2u_4}{dz^2}+\kappa\left(-\frac{1}{\sqrt{2}}\right)u_4=qz(10-z). \tag{9.119d}$$

Using the four resulting $u_i(z)$ functions, we then can use Lagrange interpolation to construct a representation of $u(z,\mathbf{x})$.

Comparing this result to that from Galerkin SFEM, we see that we have one more equation, but we do not need to solve any coupled differential equations. Moreover, the method is nonintrusive. A code that can solve the ADR equation can be wrapped in this collocation procedure—this is a definite benefit of this approach over standard SFEM. The other benefits of collocation are the same as discussed previously. If the random variables do not have a polynomial representation, we can still use collocation, though convergence may be slower.

9.9 Summary of Methods

In this chapter we discussed methods based on polynomial representations of either a quantity of interest or a function of random variables. A summary of this discussion appears below.

9.9.1 Quantities of Interest

9.9.1.1 Projection Methods

- For a given random variable, choosing the appropriate orthogonal polynomial expansion of the quantity of interest leads to the following properties:
  - The leading-order expansion coefficient will be the mean of the QoI,
  - The sum of the squares of the remaining coefficients is related to the variance of the quantity of interest.
  - The integrals that must be necessarily computed to estimate the expansion coefficients can be efficiently computed with Gauss quadrature when the number of random variables is small.
  - The expansions can be computed nonintrusively.
  - If the QoI is a smooth function of the random variables, the polynomial expansion will demonstrate rapid convergence.

- There are significant drawbacks to this approach that the practitioner must be aware of.
  - When the number of random variables is large, the number of evaluations needed to compute even a modest polynomial representation is prohibitively large due to the geometric increase in the number of function evaluations. This is the curse of dimensionality.
  - Sparse quadratures and regularized regression techniques can help with the curse of dimensionality but are not a panacea.
  - If the QoI is not a smooth function of the random variables, the resulting expansion can be inaccurate and give spurious results due to the Gibbs' phenomenon.

9.9.1.2 Collocation

- Collocation takes the value of the QoI at different values of the random variables and constructs an interpolating polynomial. This nonintrusive method has the following properties:
  - If the QoI can be expressed as a polynomial, collocation constructed by evaluating the function at the Gauss quadrature points for the appropriate orthogonal polynomials will give an equivalent representation to projection onto the same orthogonal polynomials.
  - Sparse collocation grids based on sparse quadrature points can be used for collocation.
  - Collocation and projection can be combined when one has a fixed budget of samples from the QoI and wants to get the best expansion possible from the available samples.

  The drawbacks from collocation are the same as those for projection: the curse of dimensionality and Gibbs' phenomena for non-smooth QoIs.

9.9.2 Representations of Solutions to Model Equations (SFEM)

9.9.2.1 Galerkin SFEM

The Galerkin projection technique writes the solution to a PDE as a polynomial expansion where the coefficients of the expansion are functions of the nonrandom variables (e.g., space and time).

- The resulting equations are often coupled PDEs that require different solution techniques than the original equation making the method intrusive (i.e., one needs to write a new code to solve the resulting equations).

- The curse of dimensionality and Gibbs' oscillations are still present in Galerkin SFEM solutions.
- The cost of solving, possibly large, systems of coupled PDEs is typically much larger than solving a single PDE many times.

9.9.2.2 SFEM Collocation

We can avoid the coupled PDEs and intrusive character of Galerkin SFEM by apply collocation to the solutions in an analogous way to how it is applied to a single QoI. The curse of dimensionality and Gibbs' oscillations remain, however.

9.10 Notes and References

The theory of spectral methods is covered in detail in the monograph by Boyd (2001). That work contains techniques to combat the Gibbs' oscillations and other drawbacks to polynomial representations of functions. A thorough, and thoroughly readable, discussion of function approximation in practice can be found in Trefethen (2013).

9.11 Exercises

1. A beam of radiation that strikes a slab of material will have the intensity decreased by a factor $t = \exp(-kx)$ where x is the thickness of the slab and k is the extinction coefficient, sometimes called a macroscopic cross section. If $K \sim \mathcal{N}(\mu = 5, \sigma^2 = 1)$ and $x = 1$, compute the mean and variance of $t(K)$. Plot the distribution of $t(K)$ as well.
2. Repeat the exercise using $K \sim \mathcal{N}(\mu = 2, \sigma^2 = 1)$.
3. Consider a stochastic medium where the distribution of thicknesses of two different materials is unknown. In this case the beam transmission will be given by

$$t = \exp(-k_1 x_1 - k_2(x - x_1)).$$

If $k_1 = 5$, $k_2 = 0.2$, $x = 1$ and $x_1 \sim \mathcal{N}(\mu = 0.5, \sigma^2 = 0.1)$, compute the mean and variance of $t(2)$, and plot the distribution. Is there a value of $\bar{k}$ that you can define so that

$$\exp(-\bar{k}x) = E[\exp(-k_1 x_1 - k_2(x - x_1)]?$$

4. The function $f(x) = 1/(1+x^2)$ is called the Witch of Agnesi. If $x \sim \mathscr{U}(-2, 2)$, find the best approximation to the distribution possible using 10 and 100 function evaluations to build polynomial chaos projections and stochastic collocation. Compare your results to the analytic distribution and its moments. Has the witch cast a spell on the methods?
5. Using a discretization of your choice, solve the equation.

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = D\frac{\partial^2 u}{\partial x^2} - \omega u,$$

for $u(x, t)$ on the spatial domain $x \in [0, 10]$ with periodic boundary conditions $u(0^-) = u(10^+)$ and initial conditions

$$u(x, 0) = \begin{cases} 1 & x \in [0, 2.5] \\ 0 & \text{otherwise} \end{cases}.$$

Using a polynomial chaos expansion, estimate the mean and variance in the total number of reactions

$$\int_5^6 dx \int_0^5 dt\, \omega u(x, t).$$

Use $v = 0.5$, $D = 0.125$, and assume that ω is an uncertain parameter distributed via $\omega \sim \mathscr{G}(1, .1)$. Also, report the distribution of the total number of reactions.
6. Write a code to solve Eq. (9.118), and compare the results to SFEM collocation and Monte Carlo sampling of x_1 and x_2.

Part IV
Combining Simulation, Experiments, and Surrogate Models

In this part we will make predictions that combine simulation and experimental data. In many cases we are limited by how many times we can evaluate the QoI in simulation. To help with this, we discuss the construction of surrogate models for the simulation. These surrogates allow us to bridge the gap between a limited set of simulations and experiments to make predictions using both calibrated parameters and the observed discrepancy between our simulation and previous experiments. The workhorse for this task is a surrogate model based on Gaussian process regression, as introduced in the next chapter. Chapter 11 then constructs predictive models in the framework of Kennedy and O'Hagan to use Gaussian processes to make data-informed predictions. The final chapter discusses the phenomenon of epistemic uncertainty and gives tools to address the question of how to deal with unknown uncertainties.

Chapter 10
Gaussian Process Emulators and Surrogate Models

Ille malum virus serpentibus addidit atris
praedarique lupos iussit pontumque moveri,
mellaque decussit foliis ignemque removit
et passim rivis currentia vina repressit

He to black serpents gave their venom-bane,
And bade the wolf go prowl, and ocean toss;
Shook from the leaves their honey, put fire away,
And curbed the random rivers running wine

—Virgil, *The Georgics*

The idea of a surrogate model or emulator is to have an inexpensive proxy for the QoI calculation. That is, rather than running a simulation to compute a QoI, we have some function that adequately approximates the QoI evaluation from the simulation; sometimes this function is called a response surface. With an accurate enough emulator, we can explore the uncertainty space without requiring additional QoI evaluations. This can aid design studies, worst-case scenario identification, and other applications where one needs to evaluate the QoI at many different input points.

The idea behind building a surrogate model is that the QoI is a function that takes a set of inputs and gives a scalar quantity. Therefore, we can use any type of function approximation to build an emulator/surrogate model, and we have already build surrogate models in previous chapters without calling them that. Polynomial chaos expansions and linear regression approximations to QoIs are both surrogate models: they are an approximation of the map from inputs to outputs.

In this chapter we consider an approach that has been found to be useful in uncertainty analyses in the past: Gaussian process regression. This method is designed to not only produce estimates of the QoI but also to estimate the magnitude of the error in that estimate. This estimate of the uncertainty in the surrogate

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-319-99525-0_10) contains supplementary material, which is available to authorized users.

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0_10

prediction can be added to other uncertainties in the system. The methods we discuss are based on Bayesian statistics, and it is this character of the methods that allows for the estimate of the uncertainty. We will begin by introducing a Bayesian version of linear regression before generalizing to Gaussian process regression.

10.1 Bayesian Linear Regression

We consider the case where we have a dependent variable y, the output, and a set of p independent variables, $\mathbf{x}$, the inputs. For these input/output pairs, we have n realizations, that is, for n different values of $\mathbf{x}_i$, $i = 1, \ldots, n$, we know the corresponding y_i. We are interested in computing a linear approximation to y as

$$y = \mathbf{x}^{\mathrm{T}}\mathbf{w} + \epsilon, \tag{10.1}$$

where $\mathbf{w}$ is a vector of length p of weights and $\epsilon \sim \mathcal{N}(0, \sigma_d^2)$ is the independent, identically distributed error in the model. Previously in Chap. 5, we discussed a procedure to find the weights using the least-squares procedure. As we will see, this is a maximum likelihood approach to minimizing the error in the model. We have assumed the error is random because we assume that all of our knowledge of the functional relationship is captured in the independent variables. The supposition that this error is normal is reasonable if the error might be due to measurement uncertainty in an experiment. Otherwise, the assumption of normality is for convenience and may have to be revisited.

To find appropriate values for $\mathbf{w}$, we will build a distribution of weights based on Bayes' rule (c.f. Sect. 2.7). To evaluate Bayes' rule, we need to compute the probability density of observing y_i given that the weights are $\mathbf{w}$, the inputs were $\mathbf{x}_i$, and the variance in the error is σ_d^2. Because the error is normal, we can write this probability density, sometimes called the likelihood of the data, as

$$f(y_i|\mathbf{x}_i, \mathbf{w}, \sigma_d) = \frac{1}{\sigma_d\sqrt{2\pi}} \exp\left[-\frac{(y_i - \mathbf{x}_i^{\mathrm{T}}\mathbf{w})^2}{2\sigma_d^2}\right]. \tag{10.2}$$

This equation implies that $y_i|\mathbf{x}_i, \mathbf{w}, \sigma_d \sim \mathcal{N}(\mathbf{x}_i^{\mathrm{T}}\mathbf{w}, \sigma_d^2)$ or that the data likelihood is a normal distribution for y_i that has mean $\mathbf{x}_i^{\mathrm{T}}\mathbf{w}$ and variance σ_d^2. Also, the errors were independent so that we can write the likelihood given all n data points as

$$\begin{aligned} f(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma_d) &= \prod_{i=1}^{n} \frac{1}{\sigma_d\sqrt{2\pi}} \exp\left[-\frac{(y_i - \mathbf{x}_i^{\mathrm{T}}\mathbf{w})^2}{2\sigma_d^2}\right] \\ &= \frac{1}{(2\pi\sigma_d^2)^{n/2}} \exp\left[\frac{-1}{2\sigma_d^2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})\right], \end{aligned} \tag{10.3}$$

where $\mathbf{X}$ is the $n \times p$ data matrix with each column a value of $\mathbf{x}_i$ and $\mathbf{y} = (y_1, \ldots, y_n)^{\mathrm{T}}$. This equation implies that the collection of n-dependent variables is a multivariate normal with mean vector $\mathbf{Xw}$ and a diagonal covariance matrix $\sigma_d^2\mathbf{I}$, i.e., $\mathbf{y} \sim \mathcal{N}(\mathbf{Xw}, \sigma_d^2\mathbf{I})$.

Therefore, from Bayes' rule we can write the probability of the weights given the n observations $\mathbf{y}$ and data matrix $\mathbf{X}$ as

$$\pi(\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma_d) = \frac{f(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma_d)\pi(\mathbf{w})}{\int f(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma_d)\pi(\mathbf{w})\, d\mathbf{w}}, \tag{10.4}$$

where $\pi(\mathbf{w})$ is the prior distribution of the weights. To compute the posterior distribution of the weights given the data, we need to specify a prior distribution on the weights. A reasonable prior is $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$, where Σ_p is a $p \times p$ covariance matrix. This choice of prior attempts to make the weights close to zero (the mean of the distribution is zero), and, as we will see, it is a form of regularization.

Using the prior in Eq. (10.4), we find the posterior distribution of the weights can be written as

$$\begin{aligned}\pi(\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma_d) &\propto \exp\left[\frac{-1}{2\sigma_d^2}(\mathbf{y} - \mathbf{Xw})^T(\mathbf{y} - \mathbf{Xw})\right]\exp\left[-\frac{1}{2}\mathbf{w}^{\mathrm{T}}\Sigma_p^{-1}\mathbf{w}\right] \\ &\propto \exp\left[\frac{-1}{2}(\mathbf{w} - \mathbf{w}^*)^{\mathrm{T}}\mathbf{A}(\mathbf{w} - \mathbf{w}^*)\right],\end{aligned} \tag{10.5}$$

where we have defined

$$\mathbf{w}^* = \frac{1}{\sigma_d^2}\mathbf{A}^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y},$$

and

$$\mathbf{A} = \frac{1}{\sigma_d^2}\mathbf{X}^{\mathrm{T}}\mathbf{X} + \Sigma_p^{-1}.$$

Equation (10.5) tells us what the posterior is proportional to, but we also know that the posterior is a probability distribution so the proportionality constant must properly normalize the distribution. From this argument, and the form we derived, we can state that the posterior is a normal distribution with mean $\mathbf{w}^*$ and covariance matrix $\mathbf{A}^{-1}$ or $\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma_d \sim \mathcal{N}(\mathbf{w}^*, \mathbf{A}^{-1})$.

With this posterior we could sample weights from the multivariate normal and evaluate the model. However, it is more convenient to specify a set of points that we want to evaluate the model at in a matrix denoted $\mathbf{X}^*$ and average the result over the posterior distribution of the weights. This average can be thought of as the probability density of $\mathbf{X}^*\mathbf{w}$ given the data, and $\mathbf{X}^*$:

$$f(\mathbf{X}^*\mathbf{w}|\mathbf{X}^*, \mathbf{X}, \mathbf{y}, \sigma_d) = \int f(\mathbf{X}^*\mathbf{w}|\mathbf{X}^*, \mathbf{w})\pi(\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma_d)\, d\mathbf{w}. \tag{10.6}$$

It can be shown that the distribution of $\mathbf{X}^*\mathbf{w}$ is a multivariate normal so that

$$\mathbf{X}^*\mathbf{w}|\mathbf{X}^*, \mathbf{X}, \mathbf{y}, \sigma_d \sim \mathcal{N}\left(\mathbf{X}^*\mathbf{w}^*, \mathbf{X}^*\mathbf{A}^{-1}\mathbf{X}^{*\mathrm{T}}\right). \tag{10.7}$$

The result in Eq. (10.7) gives us a way to get the distribution of the prediction from the linear model at point $\mathbf{X}^*$ by sampling from a multivariate normal. Given that we have a distribution for the prediction, we can compute the variance in the prediction, confidence intervals, etc. The form of the covariance matrix indicates that the uncertainties are quadratic in $\mathbf{X}^*$ so that the uncertainty in the prediction grows with the magnitude of $\mathbf{X}^*$. Additionally, the larger the eigenvalues of Σ_p are, the larger the uncertainty in the prediction will be. This is sensible because Σ_p represents the amount of uncertainty we believe the weights will/should have in the prior for the weights.

To this point we have not addressed the variance of the error, σ_d^2. It is likely that this error will not be known in practice. In that case we could modify the procedure to allow for a prior on σ_d^2 and allow the data to suggest this value by computing a posterior for the weights and the variance of the error. This does, unfortunately, introduce a great deal of algebraic complexity that will not provide much gain for our studies.

As an example we consider a simple linear model of the form

$$y = w_1 + w_2 x_1 + \epsilon,$$

with $n = 3$, inputs $x_1 = \{-5, 1, 5\}$, and corresponding outputs $y = \{-5.1, 0.25, 4.9\}$. If we specify a prior on the weights of $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we get the following values:

$$\mathbf{X} = \begin{pmatrix} 1 & -5 \\ 1 & 1 \\ 1 & 5 \end{pmatrix}, \qquad \mathbf{w} = (w_1, w_2)^{\mathrm{T}},$$

$$\mathbf{A} = \begin{pmatrix} \frac{3}{\sigma_d^2} + 1 & \frac{1}{\sigma_d^2} \\ \frac{1}{\sigma_d^2} & \frac{51}{\sigma_d^2} + 1 \end{pmatrix}, \qquad \mathbf{w}^* = \begin{pmatrix} \frac{0.05\sigma_d^2 - 47.7}{\sigma_d^4 + 54\sigma_d^2 + 152} \\ \frac{50.25\sigma_d^2 + 150.7}{\sigma_d^4 + 54\sigma_d^2 + 152} \end{pmatrix}.$$

We specify $\mathbf{X}^*$ to have x_1 at 100 equally spaced points between -6 and 6:

$$\mathbf{X}^* = \begin{pmatrix} 1 & -6 \\ 1 & -5.8787 \\ \vdots & \vdots \\ 1 & 5.8787 \\ 1 & 6 \end{pmatrix}.$$

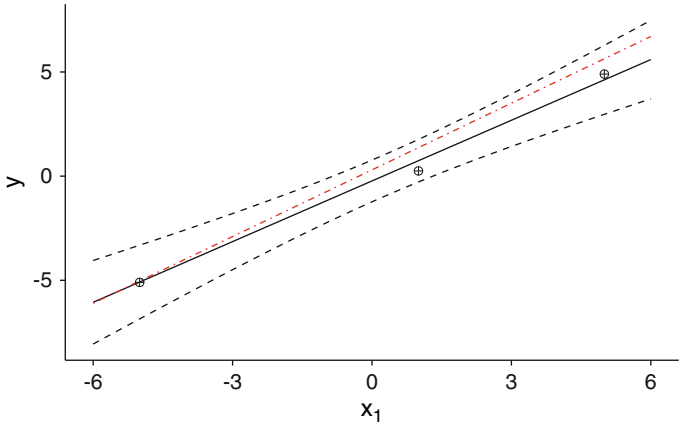

Fig. 10.1 Plot of results for Bayesian linear regression mean model (solid line) and the ± 2 standard deviation (dashed lines) for the model $y = w_1 + w_2 x_1$ using data $x_1 = \{-5, 1, 5\}$ and $y = \{-5.1, 0.25, 4.9\}$ (the three symbols) and $\sigma_d^2 = 1$. A sample from the posterior of the weight distribution is shown in the dash-dot line

The results for the fit with $\sigma_d^2 = 1$ are shown in Fig. 10.1. In this figure we see the predicted mean model (i.e., the mean value of the weights in the posterior) and the ± 2 standard deviations from the mean model that represents the estimated uncertainty in the model. The predicted behavior that the uncertainty grows as $|x_1|$ increases can be seen in the width of the uncertainty bands.

There are ways that we could improve our model, for example, as we noted that it would be ideal to estimate σ_d as a function of the data. Depending on the type of prior distribution we chose for σ_d, we would likely lose the ability to write the posterior distribution as a multivariate normal distribution. As a result we would need a means to evaluate the posterior distribution from Bayes' rule. One approach would be numerical integration of the denominator, but there is a simple means to produce samples from the posterior without knowing the full distribution. That idea, however, will be left until the next chapter. Rather we turn to how we can increase the class of functions in our regression model.

10.2 Gaussian Process Regression

We could attempt to enrich the class the models that we try to fit to include polynomials in the inputs or other functions. This added complexity is known as enhancing the feature space because the data used to build the model and any manipulations of that data are often called model features. Perhaps surprisingly, we can derive a model that has nearly the same complexity as the linear regression

case but can model a wide class of nonlinear functions. This can be thought of as finding a transformation of the independent variables to find a new set of variables that do provide a linear representation for the dependent variable.

Consider the set of monomials of a single variable x up to degree $d-1$. We write the set as

$$\phi(x) = \{1, x, x^2, \ldots, x^{d-1}\}.$$

Given the data matrix of the p variables $\mathbf{X}$ at n points, we can define a $n \times pd$ data matrix $\Phi(\mathbf{X})$ where the ith row of the matrix is $(1, x_{1i}, x_{1i}^2, \ldots, x_{1i}^{d-1}, \ldots, 1, x_{pi}^1, \ldots, x_{pi}^{d-1})$. Then we can write a model for the dependent variable y as

$$y_i = \phi(\mathbf{x})^{\mathrm{T}}\mathbf{w} + \epsilon,$$

where now there are pd weights. Clearly this model is more general than the pure linear model, and we would expect it to be able to match a wider class of functions. Nevertheless, we can replace the data matrix $\mathbf{X}$ in our previous derivation of the Bayesian linear regression model with $\Phi(\mathbf{X})$ and arrive at the predictive distribution at a set of points $\mathbf{X}^*$ as

$$\Phi(\mathbf{X}^*)\mathbf{w}|\mathbf{X}^*, \mathbf{X}, \mathbf{y} \sim \mathcal{N}\left(\frac{1}{\sigma_d^2}\Phi(\mathbf{X}^*)\mathbf{A}^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}, \Phi(\mathbf{X}^*)\mathbf{A}^{-1}\Phi(\mathbf{X}^*)^{\mathrm{T}}\right), \tag{10.8}$$

with the matrix $\mathbf{A}$ now defined as

$$\mathbf{A} = \frac{1}{\sigma_d^2}\Phi(\mathbf{X})^{\mathrm{T}}\Phi(\mathbf{X}) + \Sigma_p^{-1}.$$

Given this form of the predicted distribution, the evaluation of the action of the inverse of $\mathbf{A}$ could be expensive if the product of p and d is large. However, it is possible to specify an arbitrarily large d using the kernel trick. The kernel trick takes advantage of the fact that the feature space only appears as a quadratic form such as $\Phi(\mathbf{X})^{\mathrm{T}}\Phi(\mathbf{X})$. If we can specify a function, called a kernel function, that is equivalent to this quadratic form, we do not need to work with the entire feature space. Indeed, it is possible to specify a kernel function that is equivalent to a quadratic form involving an infinite number of features, as we will see.

To demonstrate the kernel trick we rearrange Eq. (10.8) to be

$$\begin{aligned}\Phi(\mathbf{X}^*)\mathbf{w}|\mathbf{X}^*, \mathbf{X}, \mathbf{y} \sim \mathcal{N}\Big(&\Phi(\mathbf{X}^*)\Sigma_p\Phi(\mathbf{X})(\mathbf{K} + \sigma_d^2\mathbf{I})^{-1}\mathbf{y}, \\ &\Phi(\mathbf{X}^*)^{\mathrm{T}}\Sigma_p\Phi(\mathbf{X}^*) - \Phi(\mathbf{X}^*)^{\mathrm{T}}\Sigma_p\Phi(\mathbf{X})(\mathbf{K} + \sigma_d^2\mathbf{I})^{-1}\Phi(\mathbf{X})^{\mathrm{T}}\Sigma_p\Phi(\mathbf{X}^*)\Big),\end{aligned} \tag{10.9}$$

with $\mathbf{K} = \Phi(\mathbf{X})^{\mathrm{T}} \Sigma_p \Phi(\mathbf{X})$. In Eq. (10.9) the feature space only appears in the form $\Phi(\hat{\mathbf{X}})^{\mathrm{T}} \Sigma_p \Phi(\hat{\mathbf{X}})$ with $\hat{\mathbf{X}}$ equal to either $\mathbf{X}$ or $\mathbf{X}^*$. Therefore if we specify a *kernel function* of the form

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}} \Sigma_p \phi(\mathbf{x}'), \tag{10.10}$$

we only need to specify weighted inner products of the feature space and never need to specify the feature space.

At this point we recall some insights from Sect. 2.4 regarding Gaussian processes. In that section we defined a Gaussian process as a random process where a finite collection of points are distributed as a multivariate normal. The predictive distribution for $\Phi(\mathbf{X}^*)\mathbf{w}$ is a Gaussian process with a known covariance matrix related to the kernel function.

10.2.1 Specifying a Kernel Function

As we argued above, the kernel function, also called a covariance function, can replace the feature space. In other words, specifying a kernel function is the same as specifying a feature space. The power-exponential kernel is given by

$$k(\mathbf{x}, \mathbf{x}') = \frac{1}{\lambda} \exp\left[-\sum_{k=1}^{p} \beta_k |x_k - x_k'|^{\alpha} \right]. \tag{10.11}$$

It can be shown (Rasmussen and Williams 2006) that this kernel function leads to a feature space that includes an infinite number of basis functions. The parameter β_k^{-1} can be thought of as a length scale for variable x_k. The power α is related to the smoothness of the model: a value of $\alpha = 2$ creates an infinitely differentiable covariance function. Finally, the larger λ is the smaller the covariance function is and the more the model ignores nearby points when computing the model. The statistical interpretation of this is that if λ is large, the data is significant and the model needs to respect that reality. Other kernel functions are possible, though the power-exponential covariance is the most commonly used in practice.

When using a kernel function on a data matrix, we define the matrix $K(\mathbf{X}, \mathbf{X}')$ as the matrix

$$K(\mathbf{X}, \mathbf{X}') = \begin{pmatrix} k(x_1, x_1') & k(x_1, x_2') & \ldots k(x_1, x_{n'}') \\ & \vdots & \\ k(x_n, x_1') & k(x_n, x_2') & \ldots k(x_n, x_{n'}'). \end{pmatrix}$$

This matrix is of size $n \times n'$ where n is the number of rows in the data matrix $\mathbf{X}$ and n' is the number of rows in $\mathbf{X}'$.

10.2.2 Predictions Where $\sigma_d = 0$

If there is no uncertainty assumed in the model, that is, the error in the model is assumed to be zero and $\sigma_d = 0$, we can simplify Eq. (10.9) to be

$$\Phi(\mathbf{X}^*)\mathbf{w}|\mathbf{X}^*, \mathbf{X}, \mathbf{y} \sim \mathcal{N}\Big(K(\mathbf{X}^*, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}\mathbf{y}, K(\mathbf{X}^*, \mathbf{X}^*) - K(\mathbf{X}^*, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}K(\mathbf{X}, \mathbf{X}^*)\Big). \quad (10.12)$$

This equation defines the mean and covariance function for a Gaussian process. Computing the mean function involves solving a linear system of equations that is $n \times n$, as does computing the action of the covariance matrix.

The regression model defined by Eq. (10.12) is called a Gaussian process model or Gaussian process regression (GPR). This model is clearly flexible because it is completely defined by the input data and the kernel function. A drawback of these models is that the covariance matrix and mean function are defined in terms of the inverse of an $n \times n$ matrix. Therefore, if there is a large amount of training data, it can be expensive to compute.

The parameters β_k, α, and λ are sometimes referred to as hyperparameters. This designation indicates that these values are needed to fit the model, but they also have an influence as to how the model fits the data. Later we will see how we can use the data to choose these values, but for now we assume that they are fixed. The term hyperparameter can refer to more than just these three parameters and can mean any parameter that influences the model fit, but are not given in the data.

We have implicitly assumed in the derivation that the prior mean for the mean function is the zero function (i.e., a function that is zero everywhere). This comes from the assumptions we made in the original Bayesian linear regression model. This prior can be relaxed, or the training data can be mean-centered. Rasmussen and Williams (2006) demonstrate how such a nonzero mean function can be defined. For our purposes we will assume the training data is mean-centered.

To demonstrate Gaussian process regression, we use a data set generated from the function $y = e^{-x}\sin 4x + (x-1)H(x-1) - 0.732$. The data points we have are

$$x = \{1.475, 1.859, 0.757, 0.665, 0.161, 0.175, 0.185, 1.243, 0.939, 1.606\},$$

and

$$y = \{-0.343, 0.269, -0.68, -0.493, -0.221, -0.191, -0.172, -0.767, -0.957, -0.097\},$$

and choose x^* to be 100 equally spaced points between 0 and 2. If we fit a GPR model using this data and choose $\beta = 1$, $\alpha = 1.9$, and $\lambda = 1$, we obtain the results in Fig. 10.2. In this figure we see that the GPR model interpolates the data and gives estimates of the uncertainty (as shown by the ± 2 standard deviation confidence

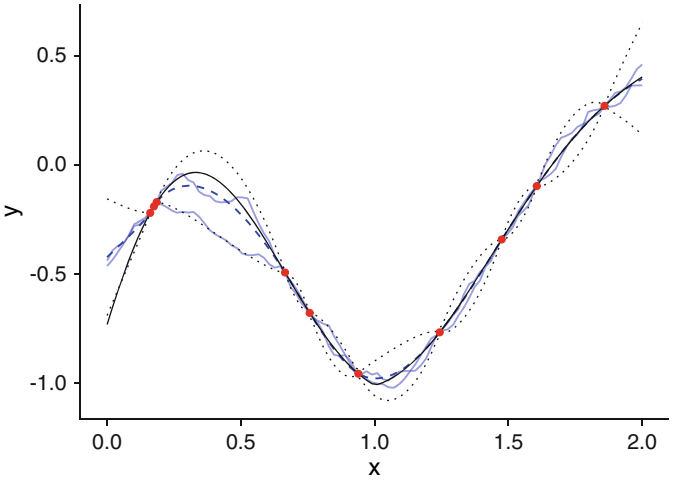

Fig. 10.2 Example GPR fit for the function $y = e^{-x} \sin 4x + (x-1)H(x-1) - 0.732$, where $H(x)$ is the Heaviside step function. The 10 points shown were used to fit the model, and the hyperparameters are $\beta = 1$, $\alpha = 1.9$, and $\lambda = 1$. The dark solid line is the true function, the dashed line is the estimated mean function in Eq. (10.12), and the dotted lines are the ± 2 standard deviation bounds around the mean. Two sample functions are also shown

intervals). This uncertainty grows where the model is extrapolating. Additionally, the mean function appears to be smooth, while the sampled functions from the distribution given in Eq. (10.12) are not smooth.

It is reasonable to wonder how the results will change when the hyperparameters are changed. Figure 10.3 shows how the results for the same data change when the hyperparameters are changed from the nominal case in Fig. 10.2. In these results we see that shrinking or growing β adjusts how quickly the function can change as a function of x. When $\beta = 0.5$ the covariance between neighboring x points is larger, and as a result, the estimated mean function does not match the true function near the peaks because it is anchored to the data. Similarly, the sampled functions are more strongly varying when β is larger.

The smoothness parameter α has a strong effect on the results. When $\alpha = 1.1$, indicating that the function is less smooth than the nominal example, we see that the predicted confidence intervals are wider, the sampled functions are highly variable. This is clearly a worse fit than the nominal example. This is because the function we are trying to fit is smooth, except at a single point ($x = 1$). If we increase the smoothness of the fit function by setting $\alpha = 2$, we see that the resulting fit is a smooth function (and the sampled functions are nearly indistinguishable from the mean). This fit appears to be better than the nominal example, except that near the non-smooth point, the prediction has a small amount of error. Additionally, this fit also has larger errors when extrapolating or near the edges of the training data.

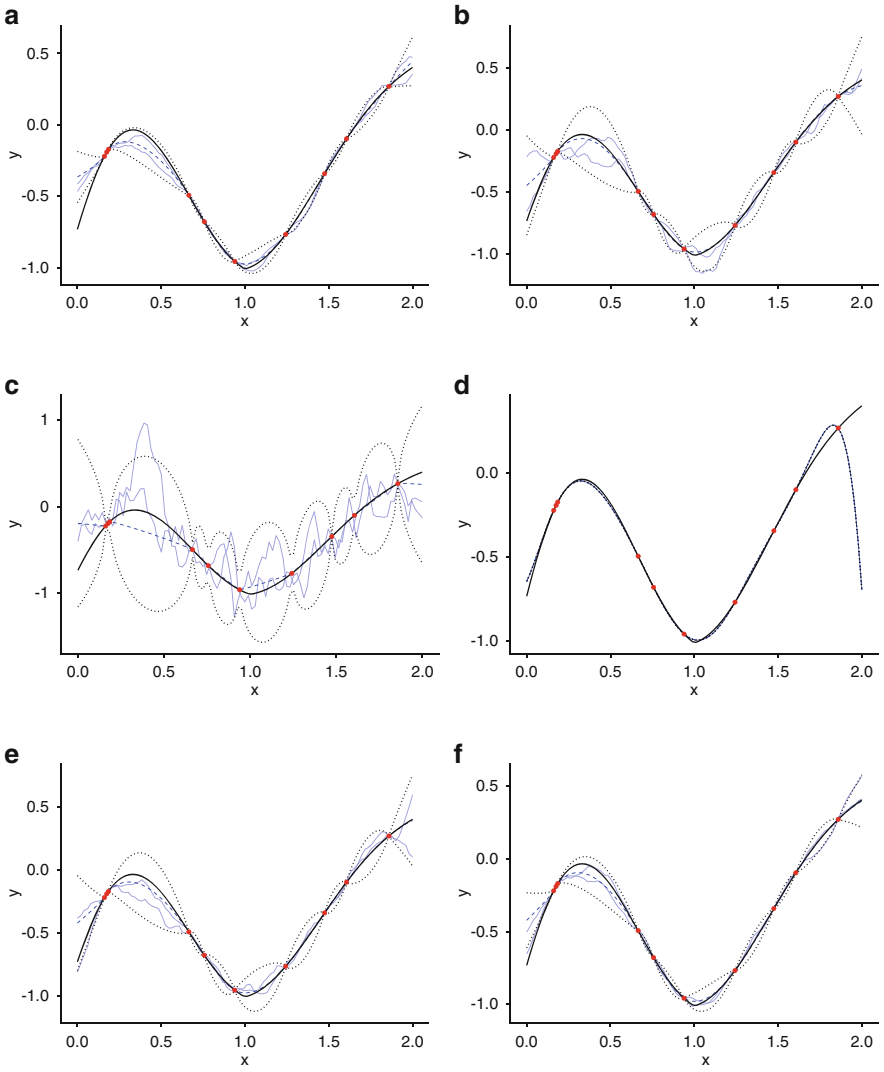

Fig. 10.3 The effect of varying hyperparameters on the GPR fit for the function $y = e^{-x}\sin 4x + (x-1)H(x-1) - 0.732$. The dark solid line is the true function, the dashed line is the estimated mean function in Eq. (10.12), and the dotted lines are the ± 2 standard deviation bounds around the mean. Two sample functions for each model are also shown. (**a**) $\beta = 0.5, \alpha = 1.9, \lambda = 1$. (**b**) $\beta = 2, \alpha = 1.9, \lambda = 1$. (**c**) $\beta = 1, \alpha = 1.1, \lambda = 1$. (**d**) $\beta = 1, \alpha = 2, \lambda = 1$. (**e**) $\beta = 1, \alpha = 1.9, \lambda = 0.5$. (**f**) $\beta = 1, \alpha = 1.9, \lambda = 2$)

Finally, the influence of λ is on the confidence in the model. When $\lambda = 0.5$, the confidence intervals are wider than the nominal example with $\lambda = 1$. Increasing λ decreases these intervals.

10.2.3 Prediction From Noisy Data

If we relax the assumption that σ_d is zero, we can write the posterior distribution of $\Phi(\mathbf{X}^*)\mathbf{w}$ as a finite sample from a Gaussian process by defining

$$\text{cov}(\mathbf{y}) = K(\mathbf{X}, \mathbf{X}) + \sigma_d^2 \mathbf{I}. \tag{10.13}$$

With this definition we can write Eq. (10.9) as

$$\Phi(\mathbf{X}^*)\mathbf{w}|\mathbf{X}^*, \mathbf{X}, \mathbf{y} \sim \mathcal{N}\Big(K(\mathbf{X}^*, \mathbf{X})\text{cov}(\mathbf{y})^{-1}\mathbf{y}, K(\mathbf{X}^*, \mathbf{X}^*) - K(\mathbf{X}^*, \mathbf{X})\text{cov}(\mathbf{y})^{-1}K(\mathbf{X}, \mathbf{X}^*)\Big), \tag{10.14}$$

The addition of a nonzero σ_d gives a floor to the covariance function and has the affect of making the model less confident near the training data.

We will modify the previous example to include noise to see the effect on the resulting model. We include a measurement uncertainty of $\sigma_d = 0.05$ and perturb the values of y in the same way so that $y = e^{-x}\sin 4x + (x-1)H(x-1) - 0.732 + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma_d^2)$. In this case we know the exact value of the measurement uncertainty, so we can force the GP model to have the correct uncertainty near the data. In Fig. 10.4 two realizations of this model are shown using 10 and 25 training points. The values for the hyperparameters are $\beta = 1$, $\alpha = 1.9$, and $\lambda = 1$. In the figure we see that the addition of noise makes the uncertainty in the model be nonzero at the data points, i.e., the confidence interval has a nonzero width at the data points. Additionally, because the data is to be less trusted due to the uncertainty, the less influence the data can have on the inferred shape of the underlying function. This is evident in the 10-point results where the peak between 0 and 0.5 is underestimated because of the noise in the data to the left of the peak. As more points are added to the training set, in this case making the total 25, we see that the estimated uncertainty in the model does decrease and the true underlying function is better approximated by the model.

In this example we knew what σ_d was for the data, and we assumed values for the other parameters. In the next chapter, we turn to answering the more common question of how to fit a GP emulator without knowledge of these parameters.

10.3 Fitting GPR Models

As we saw above the hyperparameters in the Gaussian process, regression model can have a large impact on the fit of the model. We will discuss how to fit these models and optimize these parameters. To begin we develop a simple, implementable version of the GP emulator. To do this we begin with Eq. (10.14). This equation requires the solution of two systems involving the matrix cov(y). Additionally,

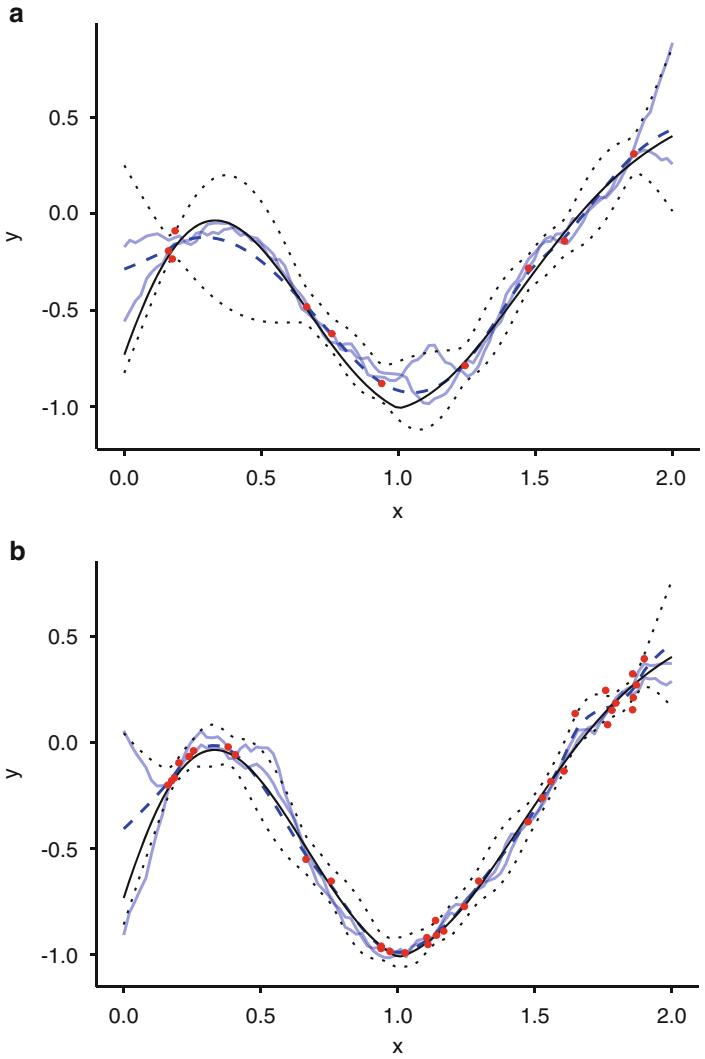

Fig. 10.4 The effect of noise in the dependent variable in the GPR fit for the function $y = e^{-x}\sin 4x + (x-1)H(x-1) - 0.732 + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma_d^2)$ using different numbers of training points. The dark solid line is the true function without noise, the dashed line is the estimated mean function in Eq. (10.14), and the dotted lines are the ± 2 standard deviation bounds around the mean. Two sample functions for each model are also shown. (**a**) 10 training points. (**b**) 25 training points

because this is a covariance matrix, it is symmetric positive definite, and we can take the Cholesky factorization of the matrix into the "square" of a lower-triangular matrix:

$$\text{cov}(y) = \mathbf{L}\mathbf{L}^{\mathrm{T}}.$$

We also define a vector that is the same length as the number of training data points, $\mathbf{k}^*$. This vector holds the covariance between a single prediction point $\mathbf{x}^*$ and the training data $\mathbf{X}$:

$$\mathbf{k}^* = K(\mathbf{X}, \mathbf{x}^*). \tag{10.15}$$

Then using Eq. (10.14) we can write the mean prediction at point $\mathbf{x}^*$ as

$$K(\mathbf{x}^*, \mathbf{X})\text{cov}(\mathbf{y})^{-1}\mathbf{y} = \mathbf{k}^* \cdot \mathbf{u}, \tag{10.16}$$

with

$$\mathbf{u} = \left(\mathbf{L}^{\mathrm{T}}\right)^{-1} \mathbf{L}^{-1}\mathbf{y}. \tag{10.17}$$

The vector $\mathbf{u}$ can be calculated by doing two triangular solves; note that this vector does not depend on the prediction point, only on the data. Therefore, we can obtain the mean prediction at point $\mathbf{x}^*$ by dotting the covariance function evaluated at the prediction point with the vector $\mathbf{u}$. In Eq. (10.16) we used the fact that the covariance kernel is a symmetric function of its arguments.

To evaluate the variance in the prediction at point $\mathbf{x}^*$, we also need to solve a linear system, but in this case, it does depend on $\mathbf{k}^*$. From Eq. (10.14), the variance at a single prediction point can be written as

$$K(\mathbf{x}^*, \mathbf{x}^*) - K(\mathbf{x}^*, \mathbf{X})\text{cov}(\mathbf{y})^{-1}K(\mathbf{X}, \mathbf{x}^*) = K(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}^* \cdot \left(\mathbf{L}^{\mathrm{T}}\right)^{-1} \mathbf{L}^{-1}\mathbf{k}^*. \tag{10.18}$$

Note that this equation will involve the solution of a linear system for each point $\mathbf{x}^*$. Also, we have to evaluate the covariance function at the point $\mathbf{x}^*$.

Therefore, the mean and variance of the prediction $f^* = \Phi(\mathbf{x}^*)\mathbf{w}$ are

$$E[f^*] = \mathbf{k}^* \cdot \mathbf{u}, \qquad \text{Var}(f^*) = K(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}^* \cdot \left(\mathbf{L}^{\mathrm{T}}\right)^{-1} \mathbf{L}^{-1}\mathbf{k}^*. \tag{10.19}$$

A python implementation of the GP regression emulator is given in Algorithm 10.1. This algorithm is based on the generic algorithm in Rasmussen and Williams (2006). The function defined takes the name of a covariance function as the argument `k`. This function must only take two arguments, i.e., `k(x,y)`. Therefore, if there are other parameters in the covariance function, a lambda function can be defined to make the covariance function compatible with the `GPR` function. In Algorithm 10.2 an example covariance function based on Eq. (10.11) is defined, along with an example lambda function to make a compatible function for GPR.

The above gives a means to fit a GP emulator given data and evaluate it a point $\mathbf{x}^*$. It does not give a means to select the hyperparameters, however. We can use cross-validation to estimate the hyperparameters. In this process we choose values for a set of hyperparameters, build the model using $N - 1$ points, and compute

Algorithm 10.1 Python code to fit a GP regression model. The covariance function k is assumed to take only two arguments

```
import numpy as np
def GPR(X,y,Xstar,k,sigma_n):
    N = y.size
    #build covariance matrix
    K = np.zeros((N,N))
    kstar = np.zeros(N)
    for i in range(N):
        for j in range(0,i+1):
            K[i,j] = k(X[i,:],X[j,:])
            if not(i==j):
                K[j,i] = K[i,j]
            else:
                K[i,j] += sigma_n**2
    #compute Cholesky factorization
    L = np.linalg.cholesky(K)
    u = np.linalg.solve(L,y)
    u = np.linalg.solve(np.transpose(L),u)
    #now loop over prediction points
    Nstar = Xstar.shape[0]
    ystar = np.zeros(Nstar)
    varstar = np.zeros(Nstar)
    kstar = np.zeros(N)
    for i in range(Nstar):
        #fill in kstar
        for j in range(N):
            kstar[j] = k(Xstar[i,:],X[j,:])
        ystar[i] = np.dot(u,kstar)
        tmp_var = np.linalg.solve(L,kstar)
        varstar[i] = k(Xstar[i,:],Xstar[i,:])
         - np.dot(tmp_var,tmp_var)
    return ystar, varstar
```

Algorithm 10.2 Example covariance function to be used with the GPR model in Algorithm 10.1

```
def cov(x,y,beta,l,alpha):
    exponent = np.sum(beta*np.abs(x-y)**alpha)
    return 1/l * np.exp(-exponent)
beta = [1.0, 2.0]
lambda = 1.0
alpha = 1.9
k = lambda x,y: cov(x,y,beta, lambda, alpha)
```

the mean prediction and variance for f^* at the point not used to build the model. This type of cross-validation is called "leave-one-out" cross-validation because it repeatedly leaves a single instance out of the training data. Using the prediction for a single point, we can compute the likelihood for the actual value y from a normal

Algorithm 10.3 Cross-validation function that performs leave-one-out cross-validation for a GPR model. The function returns the sum of the likelihoods for each predicted point

```
from scipy.stats import norm
def cross_validate(X,y,k,sigma_n):
    assert X.shape[0] == y.size
    N = y.size
    total_like = 0
    for i in range(N):
        Xstar = np.reshape(X[i,:],(1,X.shape[1]))
        Xtmp = X[np.arange(N) != i,:]
        ytmp = y[np.arange(N) != i]
        ystar, varstar = GPR(Xtmp,ytmp,Xstar,k,sigma_n)
        total_like += norm.pdf(ystar-y[i],
                               scale=math.sqrt(varstar))
    return total_like
```

distribution with a mean and variance predicted by the model. This is then repeated N times, and we compute the sum of the likelihoods from each test at the same values of the hyperparameters, $\epsilon(\mathbf{t})$, where $\mathbf{t}$ is the set of hyperparameters. We then have an optimization problem to solve: maximize the sum of the likelihoods for the predictions over the hyperparameters. Solving this optimization problem subject to reasonable constraints on the hyperparameters will give a GP model which can make predictions at a new data point that will have a high likelihood of being "correct."

Algorithm 10.3 gives a python function that will perform cross-validation to compute the sum of the likelihoods for the predicted points. This function could be an input in one of the optimization functions found in the `SciPy` package for python to find the hyperparameters that maximize the predicted likelihood.

To demonstrate the GP fitting with the algorithms defined in this section, we use a set of simulation runs from the simulation of a laser-driven shock in a disc of beryllium (Be) as reported in McClarren et al. (2011) and Stripling et al. (2013). In this data the QoI is the shock breakout time, and there are five parameters that are varied: the disc thickness, the laser energy, the Be gamma (a parameter in an ideal gas equation of state), the wall opacity, and the flux limiter constant. For more details on these parameters, see McClarren et al. (2011). The QoI as a function of these inputs are shown in Fig. 10.5. From the data in the figure, we can observe that the disc thickness and the Be gamma are clearly important parameters as the graphs show a trend in the breakout time as a function of these parameters.

We will now use this data in the GPR functions defined above. To begin we normalize and center each variable by subtracting the mean of the maximum and minimum value of the variable and dividing by the range: this makes each parameter vary between -0.5 and 0.5. Then we use cross-validation and use an optimization function to find the best values for the hyperparameters. We also want to set bounds on the hyperparameters so that they are physical. In this case since we are using the power-exponential covariance function, we set the β_i to be in the range [0.001, 10] and allow λ to vary between [0.001, 10]; α is fixed to be 2, and we do not vary it in

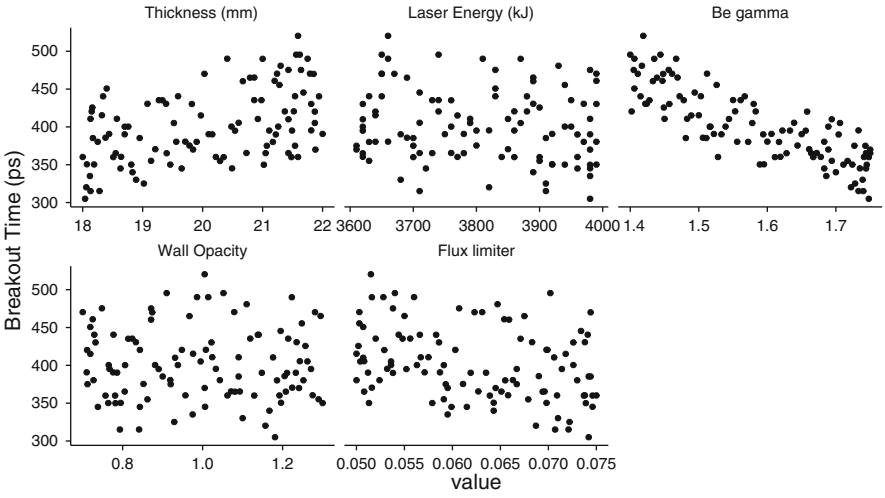

Fig. 10.5 The QoI, shock breakout time, as a function of the five inputs for the laser-driven shock simulation

this problem. Furthermore, since this is simulation data that is not subject to noise in the observation (i.e., if we ran the same simulation again, we would get the same result), we set $\sigma_d = 0$. Another consideration when fitting the model is that we split the data into a test and training set, with 80% of the data being randomly placed in the training set. This allows us to ensure that using cross-validation and maximizing the likelihood of the model are not overfitting the available data.

We use the cross-validation procedure given in Algorithm 10.3 and a minimization function from `scipy.optimize` to find the maximum likelihood values starting at $\beta = (1, 1, 1.5, 0.01, 0.05)$ and $\lambda = 1$. Results from this fit are shown in Fig. 10.6. The optimized values were

$$\beta = (0.98944159, 0.95621941, 1.50252907, 0.02134776, 0.04615761).$$

These values indicate that we started near a local maximum in the likelihood because the β's did not change much. It also indicates that the value of Be gamma was the most important input variable in predicting the breakout time. Because we normalized the inputs, the β's give an indication of what variables have the most impact on the covariance function: a larger value of β_i indicates that variable is more important. The values of β_i are sometimes called the relative relevances. In this case the relative relevances suggest that Be gamma, laser energy, and disc thickness are the key variables to consider when seeking to affect the shock breakout time.

In the figure we can see that the GPR model exactly predicts the training data (as expected since we set $\sigma_d = 0$), and for the test data, we see a small disagreement for some of the points. For many of the test points, the true value was within ± 2 standard deviations of the prediction, though some are outside that bound. On the whole the predictions have an average error of 9.74 ps.

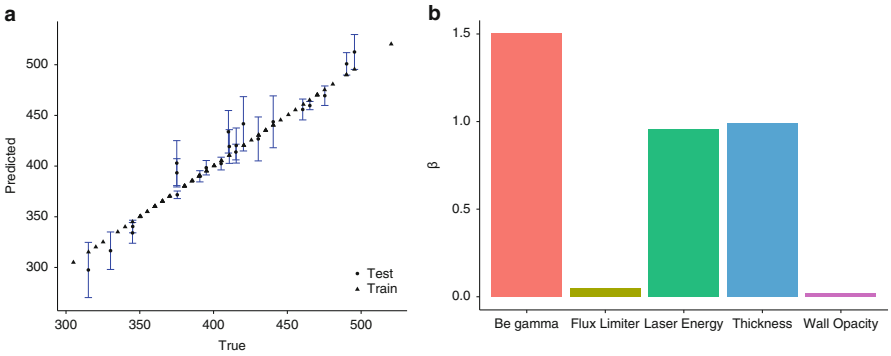

Fig. 10.6 Results for a GPR model fit with 80% of the simulation data. For the predicted versus actual plot, the error bars are the two times estimated standard deviation in the prediction. (**a**) Predicted versus actual breakout times. (**b**) Relative relevance of inputs

10.4 Drawbacks of GPR Models and Alternatives

The most common complaint about GPR is that it is expensive to build a model when there is a lot of training data. This is due to the fact that the construction of the model requires the Cholesky factorization of a dense matrix with a size equal to the number of training points; Cholesky factorization of such a matrix requires $O(N^3)$ operations for a size N matrix. As we saw above and will see in the next chapter, it is typical to build many GPR models to find an optimal one, so this cost is further multiplied by the number of models we want to construct. To this end there has been work on local GP models that only include a subset of the data (Gramacy and Apley 2015) and can be implemented efficiently (Gramacy et al. 2014).

Another issue with the GP models we used here is that the covariance kernel was applied to all of input space; that is, we assumed a stationary covariance function. In many problems the character of the covariance function needs to change in different regimes. This is particularly acute in problems where the dependent variable is constant for a large region of space and then begins to vary once some threshold is crossed. To address this issue, Gramacy and Lee (2008) developed a hybrid tree-Gaussian process model that allows the covariance function to change over the range of input data.

Indeed GPR is not the only possible approach to model computer simulations. There has been success using the Bayesian Multiple Adaptive Regression Splines (MARS) method of Denison et al. (2002, 1998). This method allows for a distribution of piecewise polynomial functions to be fit to the data. These methods can automatically handle some of the issues with nonstationary covariances, and the underlying computation in fitting a Bayesian MARS model is a least-squares solve, which can be done efficiently.

The techniques of Gaussian process models and Bayesian MARS are but two examples of machine learning approaches to finding the functions underlying a

given data set. No discussion of machine learning would be complete without mentioning neural networks. This approach to determining a functional fit to a set of input/output data has been shown to be able to solve a variety of problems (LeCun et al. 2015), including constructing surrogate models for computer simulation of complicated multi-physics problems (Spears 2017; Humbird et al. 2017) and discovering physical laws (Raissi and Karniadakis 2018). Additionally, the theory of neural networks suggests that a common technique to regularize neural networks called dropout is equivalent to a Gaussian process (Gal and Ghahramani 2016), giving a direct connection between our discussion here the world of neural networks.

In addition to neural networks, decision tree-based methods are a common, black box technique to develop surrogate models. In particular the random forest method, where an ensemble of decision trees is used to create predictions (Breiman 2001), has few hyperparameters to tune and can give good results for a variety of problems. For example, random forests have been used as a surrogate for simulation data to discover a new means of increasing the amount of nuclear fusion in experiments (Peterson et al. 2017) and to understand when mathematical model assumptions are violated (Ling 2015). The combination of random forests and neural networks has also shown promise on creating emulators for simulation data (Humbird et al. 2017).

10.5 Notes and References

The monograph by Rasmussen and Williams (2006) gives a book-length discussion of Gaussian process models for regression and classification. There are implementations of GP regression for python via the `sklearn` library and for R in the `tgp` package (Gramacy et al. 2007).

10.6 Exercises

1. Show that maximizing the likelihood in Eq. (10.2) over the weights leads to the standard least-squares regression model.
2. Consider the function

$$f(x,k) = \frac{1}{1+e^{kx}} + \epsilon,$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2 = 0.01)$. Generate 100 samples from this function for $x \in [-2, 2]$ and $k \in [1, 10]$. Fit a Gaussian process regression model to this data as a function of x and k using the correct measurement uncertainty, $\alpha = 2$ and $\beta_i = 1$. Compare the result to the true function. Repeat the exercise by finding the most likely value of the hyperparameters starting the search near these parameters.

Chapter 11
Predictive Models Informed by Simulation, Measurement, and Surrogates

The most difficult challenge to the ideal is its transformation into reality, and few ideals survive.

—William Gaddis, *The Recognitions*

In this chapter we develop the idea of using statistical models to fuse experiments/measurements with simulation data. Our approach will use Gaussian process models to model both simulation results and discrepancies between simulations and experiments. The idea behind all of these approaches is to construct a model that can be trained to assign differences between a measurement and a simulation to a calibration parameter and, if necessary, find a function for the difference between the results and the simulation. The discussion of calibration and Kennedy-O'Hagan models follows the notation and form of Higdon et al. (2004); the interested reader is encouraged to see that work for more applications of these models. We also introduce the idea of having a hierarchy of simulations and how to combine them, including how to use a low-fidelity model that is "free" to evaluate.

11.1 Calibration

We begin with the problem of calibration. In this situation we have a computer model (i.e., a simulation tool) that we believe to be an adequate model of the true QoI that we measure in an experiment when certain parameters are appropriately calibrated to the correct value. These calibration parameters could be the coefficients in an approximate model such as a turbulence model, a constitutive model, or even in a phenomenological model. We may have bounds or even an estimate of these parameters.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-319-99525-0_11) contains supplementary material, which is available to authorized users.

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0_11

We separate the calibration parameters from the controlled experimental parameters by denoting the calibration parameters by $\mathbf{t} = (t_1, \ldots, t_q)^{\mathrm{T}}$ and the controllable parameters by $\mathbf{x} = (x_1, \ldots, x_p)^{\mathrm{T}}$. We denote the simulation's prediction of the QoI, which is a function of both $\mathbf{x}$ and $\mathbf{t}$ as $\eta(\mathbf{x}, \mathbf{t})$. If we have N measurements of the QoI, y, then our calibration problem can be formulated as

$$y(\mathbf{x}_i) = \eta(\mathbf{x}_i, \mathbf{t}_i) + \epsilon_i, \qquad i = 1, \ldots, N. \qquad \text{Calibration Problem}$$

In this problem we have assigned all of the disagreement between the simulation and the measurement of the QoI as measurement error ϵ. Additionally, we have purposefully written y as a function of $\mathbf{x}$ only and not as a function of $\mathbf{t}$. This is because typically the calibration parameters may not have a physical interpretation: they are parameters that we need to make our code give good answers. In other words, nature does not care about our calibration parameters.

The calibration problem gives a straightforward methodology to attempt to combine experimental and simulation data to improve the simulation. Nevertheless, we have not specified how to solve the problem, and at this point, we have not specified enough information to solve it. A reasonable approach to solving this problem, due to its statistical nature and the combination of deterministic (the simulator) and stochastic information (the measurement), would be to use Bayes' rule.

To use Bayes' rule, we will need to specify a prior for the calibration parameters and the measurement error. For the calibration parameters, we will typically have an interval of values that each can take or have other information that we can use to construct a prior. The measurement error will typically be reported using some notion of a distribution by the experimenter that could be used to inform the prior. As a word of caution, do not always assume that the measurement error reported by the experiment to be a normal distribution, even if the experiment uses the parlance of normal random variables, such as standard deviation. In the author's experience, further investigation will uncover that some sources of error are non-normal.

Given priors for the calibration parameters and the measurement error, we can use Bayes' rule to update our estimate for $\mathbf{t}$ and the measurement error distribution given a set of measurements, $\mathbf{y}$ as

$$\pi(\mathbf{t}, \epsilon|\mathbf{y}, \mathbf{x}) = \frac{f(\mathbf{y}|\mathbf{x}, \mathbf{t}, \epsilon)\pi(\mathbf{t})\pi(\epsilon)}{\displaystyle\int d\mathbf{t} \int d\epsilon \; f(\mathbf{y}|\mathbf{x}, \mathbf{y}, \epsilon)\pi(\mathbf{t})\pi(\epsilon)}. \tag{11.1}$$

11.1.1 Simple Calibration Example

It may be possible that the experimental error is well characterized and we know the properties of the distribution. We will explore the case of a known measurement error distribution to illustrate how calibration can be performed. If the measurement

errors are said to be independent and each is normal with mean zero and a known standard deviation, σ, then we can write the likelihood in Eq. (11.1) as

$$f(\mathbf{y}|\mathbf{x}, \mathbf{t}, \epsilon) = \frac{1}{(2\pi)^{N/2}\sigma^N} \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y(\mathbf{x}_i) - \eta(\mathbf{x}_i, \mathbf{t}_i))^2\right]. \tag{11.2}$$

This results in the posterior distribution being written as

$$\pi(\mathbf{t}, \epsilon|\mathbf{y}, \mathbf{x}) = \frac{f(\mathbf{y}|\mathbf{x}, \mathbf{y}, \epsilon)\pi(\mathbf{t})}{\int d\mathbf{t} f(\mathbf{y}|\mathbf{x}, \mathbf{y}, \epsilon)\pi(\mathbf{t})}. \tag{11.3}$$

To further explore this case, we consider a simple experiment designed to measure the acceleration due to gravity. An object at rest is dropped in a vacuum from a known height and the time to drop to the bottom of the container is measured. Using a simple kinematic model, the time in seconds, y for the object to fall the distance x in meters is

$$\eta(x, g) = \sqrt{\frac{2x}{g}},$$

where g in meters per second squared is the calibration parameter. We obtain the following measurements and model evaluations:

$$x[m] = \{11.3315, 10.1265, 10.5592, 11.7906, 10.8204\},$$

$$y[s] = \{1.51877, 1.43567, 1.46605, 1.54926, 1.48409\}.$$

We also know that the measurement error is normally distributed with mean zero and standard deviation of 0.001 s. The prior distribution[1] for g is said to be normal with mean 9.81 and standard deviation 0.01 m/s^2. Evaluating the posterior using numerical integration, we get the logarithm of the posterior distribution for g to be

$$\log \pi(g|\mathbf{y}, \mathbf{x}) \approx -200g^2 + 3924g + \frac{3.486 \times 10^7}{\sqrt{g}} - \frac{5.463 \times 10^7}{g} - 5.58 \times 10^6.$$

The results for this calibration are shown in Figs. 11.1 and 11.2. The prior and posterior estimates and confidence intervals for the time are shown in Fig. 11.2. It is clear that the measurement data selected values of g that agree with the data within the measurement errors. Additionally, we can see that the five measurements cause our knowledge of g to improve when we compare the width of the posterior distribution to the prior for g in Fig. 11.1.

[1] In actuality this a very wide range for g as it has been known to five significant digits since at least the 1960s (Cook 1965; Tate 1968).

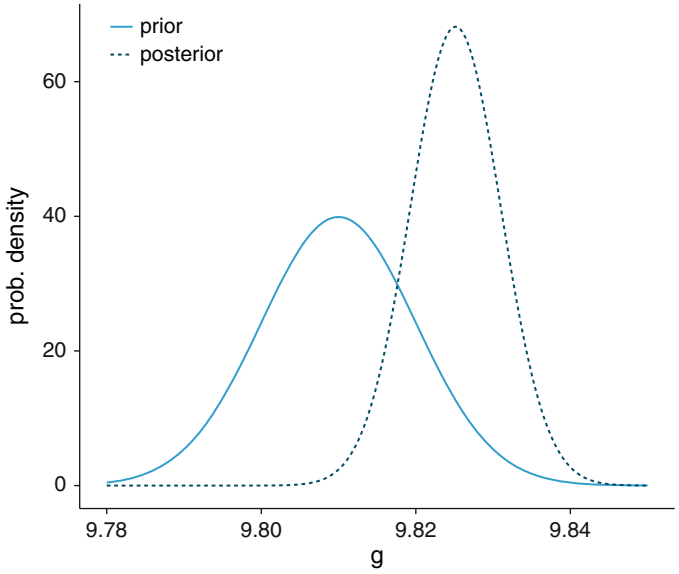

Fig. 11.1 Comparison of the posterior and prior distributions for the calibration example after five measurements

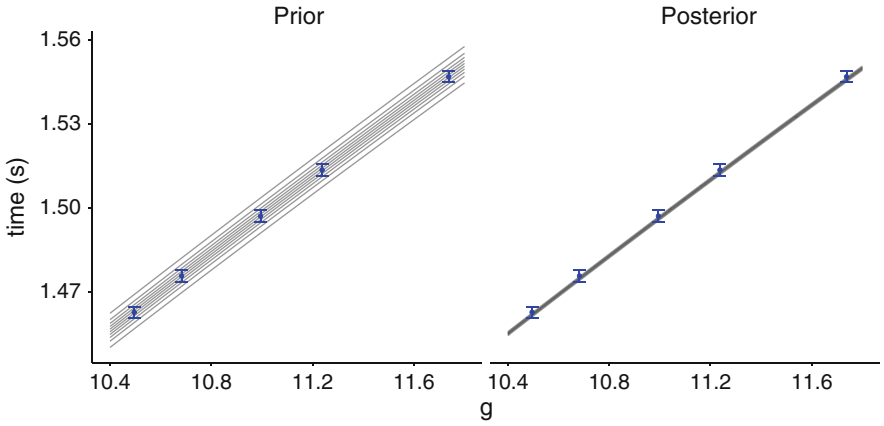

Fig. 11.2 Model results before and after calibration. The lines are results from the model with g selected at the $5, 15, \ldots, 85, 95$ percentile of the prior and posterior, respectively. The points are the experimental measurements with a two standard deviation uncertainty

11.1.2 Calibration with Unknown Measurement Error

The example above was simple for several reasons. Two of those reasons regard the problem formulation: it had only a single calibration variable, and the experimental measurement uncertainty was known. The fact that the cost to evaluate the QoI was

"free" is what made the calibration simple to perform. It allowed us to perform the integration in Bayes' rule using numerical integration and get a closed form for the prior. In practice most QoI calculations will require running a computer code, and we cannot afford to run it as many times as it would require to perform the numerical integration for the denominator in Bayes' rule, and then each evaluation of the posterior would require another evaluation of the QoI.

To handle this situation, we can generate samples from the posterior distribution without needing to evaluate the integral in the denominator. This is accomplished using Markov Chain Monte Carlo, which we will discuss next.

11.2 Markov Chain Monte Carlo

When dealing with Bayes' rule, we can often write down the numerator in the expression for the posterior, but the denominator, which normalizes the distribution, may not be known or may be difficult to compute. The knowledge of the numerator gives us an expression for the posterior distribution but only up to a multiplicative constant. There is a method for generating samples from a distribution if one only knows a constant multiple of the distribution known as the Metropolis-Hastings algorithm for Markov Chain Monte Carlo. We can use this algorithm to sample the posterior from Bayes' rule if we only know the data likelihood and the prior.

11.2.1 Markov Chains

Consider a collection of random variables $\{x_0, x_1, \ldots, x_t, x_{t+1}, \ldots\}$ such that at each index, $t \geq 0$, the next state x_{t+1} is a sample from the conditional probability $P(x_{t+1}|x_t)$. That is, each random variable only depends on the one that immediately precedes it in the sequence. Such a sequence of random variables is referred to as a Markov chain, and $P(x_{t+1}|x_t)$ is known as the transition probability. The index t is sometimes referred to as "time."

One important property of Markov chains is that as t gets large, the distribution of x_t is independent of x_0. In other words, the chain can forget its initial state. The resulting distribution of the x_t is called the stationary distribution. The fact that if the transition probably is properly defined, we can control the distribution of x_t for $t \gg 0$; we use this property to generate samples from the posterior.

The property that the Markov chain forgets its initial state if t gets large enough leads to the Markov Chain Monte Carlo estimator. If we want to estimate the expected value of $g(x)$ where x is distributed according to the stationary distribution of the Markov chain, we define a time m, where we say once $t > m$, we have reached the stationary distribution. This cutoff time is called the "burn-in" period. The resulting estimator is

$$E[g(x)] \approx \frac{1}{n-m} \sum_{t=m+1}^{n} g(x_t). \tag{11.4}$$

The choice of burn-in length is nontrivial and we will discuss it below.

11.2.2 Metropolis-Hastings Algorithm

We want to construct a Markov chain with stationary distribution that is the posterior from Bayes' rule. The Metropolis-Hastings algorithm (MH) provides a means to accomplish this task. We only need to be able to evaluate the product of the prior and the likelihood function; we call this unnormalized target distribution $\hat{p}(x)$. MH is a rejection sampling technique that uses a distribution that is not the target distribution to generate proposed samples. The algorithm begins with this proposal distribution that we write as $q(y|x_t)$: the proposal distribution can depend on the current chain state. In practice the proposal distribution is often chosen to be a multivariate normal with mean x_t. A sample is proposed by sampling y from $q(y|x_t)$. Then the acceptance probability of y is computed as

$$\alpha(x_t, y) = \min\left(1, \frac{\frac{\hat{p}(y)}{q(y|x_t)}}{\frac{\hat{p}(x_t)}{q(x_t|y)}}\right) = \min\left(1, \frac{\hat{p}(y)q(x_t|y)}{\hat{p}(x_t)q(y|x_t)}\right). \tag{11.5}$$

The acceptance probability is defined so that if the ratio of the proposed point to its probability of being proposed is greater than the ratio of the likelihood of the current chain state to the probability of it being proposed from y, the proposal is always accepted. In other words, if the gain in likelihood is high relative to the probability of it being proposed, as compared with the current chain likelihood relative to the chain going back to x_t, we accept. Otherwise, we accept with some probability based on the ratio in Eq. (11.5). This allows the chain to not get stuck at a local maximum because it can step to a lower likelihood with some probability.

If the proposal y is accepted, then $x_{t+1} = y$, otherwise the chain does not change and $x_{t+1} = x_t$. The MH algorithm is written in Algorithm 11.1.

MH generates a Markov chain where the stationary distribution is the properly normalized form of $\hat{p}$. Also, once MH generates a sample from the target distribution, all the subsequent samples will also be from the target distribution. This explains the necessity of the burn-in period. In the following subsection, we demonstrate the properties of the stationary distribution created by MH. This discussion is optional and not essential to comprehend the remainder of this chapter.

Algorithm 11.1 Metropolis-Hastings Algorithm for generating samples from a Markov Chain with unnormalized stationary distribution $p(x)$

```
Pick x_0.
for t = 0 to T do
    Sample y ~ q(·|x_t).
    Compute α(x_t, y) from Eq. (11.5).
    Sample u ~ 𝒰(0, 1).
    if u ≤ α(x_t, y) then
        Set x_{t+1} = y
    else
        Set x_{t+1} = x_t.
    end if
end for
```

11.2.3 Properties of Metropolis-Hastings Algorithm

We consider the case where the target distribution is the posterior from Bayes' rule, i.e.,

$$\hat{p}(x) \equiv \pi(x|D) \int p(D|x)\pi(x)\, dx = p(D|x)\pi(x), \tag{11.6}$$

where D denotes the data that we have, $p(D|x)$ is the data likelihood conditional on a value of x, and $\pi(x)$ is the prior on x. The definition of $\alpha(x_t, y)$ from Eq. (11.5) gives

$$\begin{aligned} \alpha(x_t, y) &= \min\left(1, \frac{\pi(y|D)q(x_t|y)}{\pi(x_t|D)q(y|x_t)}\right) \\ &= \min\left(1, \frac{\hat{p}(y)q(x_t|y)}{\hat{p}(x_t)q(y|x_t)}\right). \end{aligned}$$

Notice that the posterior distribution appears in the expression for α because the constant of normalization cancels. Upon manipulating this equation, we can get the equality:

$$\pi(x_t|D)q(x_{t+1}|x_t)\alpha(x_t, x_{t+1}) = \pi(x_{t+1}|D)q(x_t|x_{t+1})\alpha(x_{t+1}, x_t). \tag{11.7}$$

We note that $q(x_{t+1}|x_t)\alpha(x_t, x_{t+1})$ is the probability density of the chain moving from x_t to x_{t+1} (the probability density of proposing x_{t+1} times the acceptance probability). Therefore,

$$P(x_{t+1}|x_t) = q(x_{t+1}|x_t)\alpha(x_t, x_{t+1}).$$

Therefore, Eq. (11.7) leads to the detailed balance equation:

$$\pi(x_t|D)P(x_{t+1}|x_t) = \pi(x_{t+1}|D)P(x_t|x_{t+1}). \tag{11.8}$$

This equation indicates that the probability of transitioning to state x_{t+1} from x_t is the same as transitioning to state x_t from state x_{t+1}. Such a Markov chain is said to be reversible.

If we integrate the detailed balance equation over all values of x_t, we get

$$\int \pi(x_t|D)P(x_{t+1}|x_t)\,dx_t = \pi(x_{t+1}|D)\underbrace{\int P(x_t|x_{t+1})\,dx_t}_{1}. \tag{11.9}$$

The result in Eq. (11.9) gives the posterior evaluated at state x_{t+1} given that state x_t is a sample from the posterior. Therefore, from the property of the stationary distribution of the Markov chain, once one sample x_t is from the posterior distribution, all subsequent samples will be as well. As a result, after a long-enough burn in the samples, x_t will be samples from $\pi(x|D)$.

11.2.4 Further Discussion of Metropolis-Hastings

The original algorithm presented by Metropolis et al. (1953) had a symmetric proposal distribution in that $q(y|x) = q(x|y)$. This makes α an even easier calculation because the proposal distribution cancels in Eq. (11.5). This original algorithm was used to generate possible configurations of atoms for statistical mechanics calculations, a situation where the atoms will obey detailed balance and the distribution of energies is known up to a multiplicative factor. The flexibility to have q be more general can be useful in practice.

There are variations to the MH algorithm that can improve performance. For example, it is possible to propose points for one dimension at a time when attempting to sample from a multivariate distribution. This will improve the acceptance rate of proposals because the likelihood of generating an accepted point in a d-dimensional space is smaller than that for a single dimension. For example, if each dimension in d is proposed from an independent distribution and the probability of acceptance in any single dimension is θ, the probability of accepting the d-dimensional proposal is $\theta^d \leq \theta$.

A popular variation on the MH algorithm with sampling one dimension at a time is known as Gibbs sampling. This method uses a proposal distribution for state $t+1$ for dimension i that is the target distribution conditioned on the values of $x_{j,t+1}$ for $j < i$ and on the values $x_{j,t}$ for $j > i$. This proposal distribution guarantees that the proposal will be accepted. To use this method, it must be possible to evaluate/sample from the conditioned target distribution. Some Bayesian models are well suited to this type of sampling.

It is also possible to generate several Markov chains with MH in parallel. If we have access to several processors, each processor can generate an independent Markov chain that will be sampling the target distribution (and due to the random proposal distribution will produce different samples). Additionally, one can cross over the chains where certain dimensions of the chain state are swapped between the parallel chains. For example, if there are two chains running in parallel and there are p dimensions in the samples, after every τ steps in the Markov chain, the first $p/2$ dimensions from chain 1 are swapped with those from chain 2 for a new proposal distribution. This has the effect of mixing the Markov chains to better explore the space.

The length of time for burn-in is, unfortunately, more of an art than a science. The standard advice is to use 1–2% of the total samples for burn-in (Gilks and Spiegelhalter 1996). There have been diagnostics developed to attempt to inform when burn-in is complete. However, it is possible to show that for a given diagnostic, it is possible to create a chain that can fool it. Therefore, it is necessary to monitor the chain (and look at chains of different lengths by taking sub-chains from a single chain or using parallel chains) to see if stability in the properties of the chain have been obtained.

During the burn-in period, it is often a good idea to monitor the acceptance rate of the chain over some period and adjust the proposal distribution until the acceptance rate is approximately 0.5 for a single variable sampler to 0.23 for a multivariate case with a large number of dimensions (Roberts et al. 1997). This will assure that the chain is balancing exploring the space (taking large steps) versus having too many samples rejected (and wasting the effort to generate the samples).

Finally we note that in practice, the estimator in Eq. (11.4) can be modified to deal with autocorrelation in the chain. The values in the chain are correlated with the previous values because of the ability for the algorithm to reject a proposal and the fact that the proposal may be a function of the current chain state. To account for this in the estimator, it is common to introduce a sampling period s such that only every s states of the chain are included in the estimator:

$$E[g(x)] \approx \frac{1}{n_s} \sum_{t=m+1}^{n} g(x_t)\delta_{0,\, t \bmod s} \tag{11.10}$$

and n_s is the number of points between $m + 1$ and n that were used in the estimator. This estimator helps to counteract the autocorrelation of the samples. The value of s may be chosen so that several acceptances are likely to have occurred between sample points. The autocorrelation of the chain may also be used to select s: the larger the autocorrelation, the larger s needs to be. The effect of a larger autocorrelation is, therefore, to reduce the number of samples used in the estimator. It has been claimed that $s = 5$ is a default value used in practice (Denison et al. 2002).

11.2.5 Example of MCMC Sampling

As an example of the MH algorithm, we consider the task of sampling from a standard normal distribution, $\mathcal{N}(0, 1)$. While there are better approaches, we can use MH to accomplish this task. The interesting feature of our approach is that we will use a different distribution to propose new points. In particular we use a proposal distribution $\mathcal{N}(x_t, \sigma^2)$ where σ has different values. The result is we can sample a standard normal using samples from a nonstandard normal.

The initial behavior of the Markov chain generated by MH with $x_0 = 3$ is shown in Fig. 11.3 for different values of σ in the proposal distribution. At $\sigma = 0.01$ the chain does not move very far from the initial state. This is indicative of a proposal distribution that is not generating samples that are far enough from the current state of the chain. Also, the acceptance rate of the chain is low. When $\sigma = 0.1$, the chain accepts over 89% of the proposals, and the chain does begin to explore the target distribution. However, in comparison to proposals coming from $\sigma = 1$ and the theory discussed above, the $\sigma = 0.1$ results have too high an acceptance rate and not enough dynamic range. When the proposal distribution proposes points far from the current chain state, as in the $\sigma = 10$ results, the acceptance rate can be low. These results do explore the range of the target distribution, at the price of having many samples rejected.

These chains were continued to 10^5 points. Using a burn-in length of 10^4 and using every tenth point, we generate the histograms in Fig. 11.4. In these histograms we see that the resulting distributions for $\sigma \geq 0.1$ appear to be standard normals and the calculated mean and standard deviations give reasonable agreement with

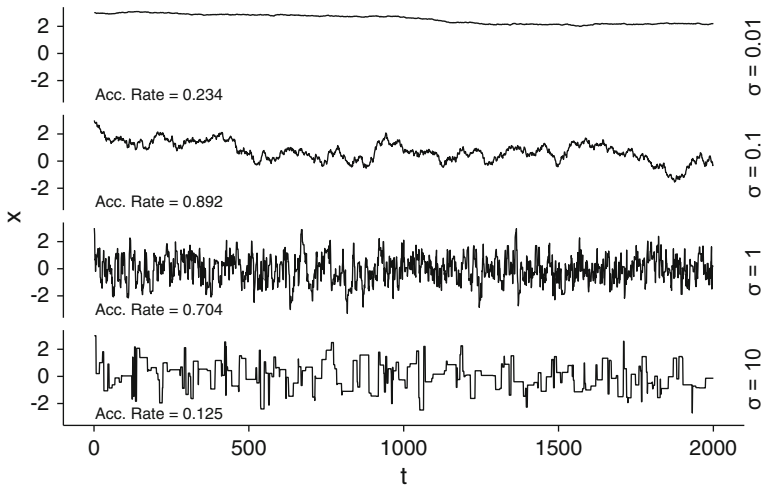

Fig. 11.3 Markov chains generated by the Metropolis-Hastings algorithm using a standard normal for the target distribution, $\hat{p}(x) = \phi(x)$, and a proposal distribution $\mathcal{N}(x_t, \sigma^2)$ for different values of σ. Each chain starts at $x_0 = 3$

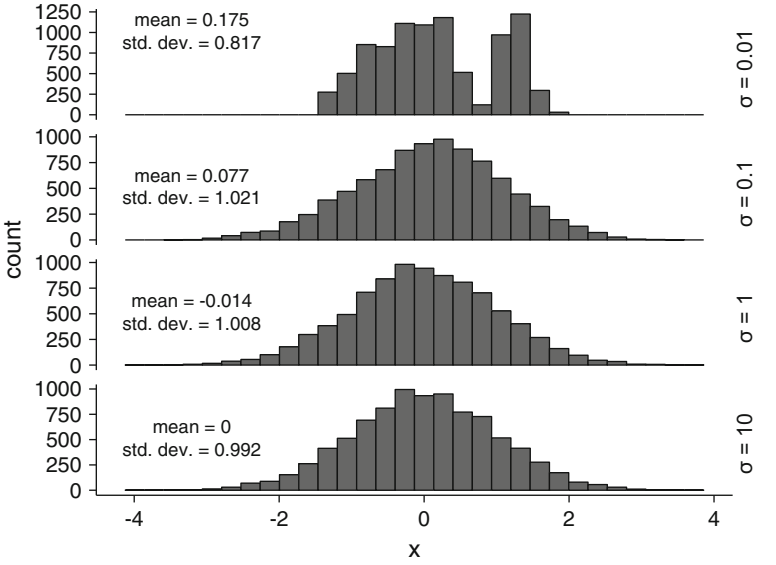

Fig. 11.4 Histogram for Markov chains of length 10^5 with a burn-in period of 10^4 and sampling period of 10 generated by the Metropolis-Hastings algorithm using a standard normal for the target distribution, $\hat{p}(x) = \phi(x)$, and a proposal distribution $\mathcal{N}(x_t, \sigma^2)$ for different values of σ

the expected values of 0 and 1, respectively. However, the $\sigma = 0.01$ results do not generate samples that approximate a standard normal distribution. It is possible to run the chain for much longer to produce a reasonable histogram.

11.3 Calibration Using MCMC

With MCMC we are able to sample from the posterior distribution of the calibration parameters, $\mathbf{t}$. We can use this capability to obtain samples from the posterior with a finite number of simulation outputs. To pose this problem, we consider the case where we have N measurements at points $\mathbf{x}_i$, that is, we have $\{y(\mathbf{x}_1), \ldots, y(\mathbf{x}_N)\}$. At these points we wish to know the value of the calibrated simulation, $\{\eta(\mathbf{x}_1, \mathbf{t}_c), \ldots, \eta(\mathbf{x}_N, \mathbf{t}_c))\}$. We also have M simulations at other points in input space $\{\eta(\mathbf{x}_1^*, \mathbf{t}_1^*), \ldots, \eta(\mathbf{x}_M^*, \mathbf{t}_M^*)\}$; here the asterisks denote simulations that do not necessarily correspond to the experimental measurements.

We combine the measurements and simulations into a single vector:

$$\mathbf{z} = \{y(\mathbf{x}_1), \ldots, y(\mathbf{x}_N), \eta(\mathbf{x}_1^*, \mathbf{t}_1^*), \ldots, \eta(\mathbf{x}_M^*, \mathbf{t}_M^*)\}.$$

Using this vector we can formulate the calibration problem using a Gaussian process regression model as the simulation:

$$\begin{aligned} \mathbf{z}_i &= \hat{\eta}(\mathbf{x}_i, \mathbf{t}_c) + \epsilon_i, \qquad i = 1, \ldots, N, \\ \mathbf{z}_i &= \hat{\eta}(\mathbf{x}^*_{i-N}, \mathbf{t}^*_{i-N}), \qquad i = N+1, \ldots, N+M. \end{aligned} \tag{11.11}$$

where $\hat{\eta}$ denotes a Gaussian process model for the simulation. Notice that $\mathbf{t}_c$ is unknown at this point.

We assume that the measurement uncertainty is normally distributed and that we know the covariance for the observations. In particular, this means that for ϵ, we can write down an $N \times N$ covariance matrix for the measurements, Σ_y. We also will assume a covariance function for the simulation that is a power-exponential kernel, as shown in Sect. 10.2.1. We write the kernel function to explicitly include both the experimental controls and the calibration parameters:

$$k(\mathbf{x}, \mathbf{t}, \mathbf{x}', \mathbf{t}') = \frac{1}{\lambda}\left(\exp\left[-\sum_{k=1}^{p} \beta_k |x_k - x'_k|^\alpha\right] + \exp\left[-\sum_{k=1}^{q} \beta_{k+p} |t_k - t'_k|^\alpha\right]\right). \tag{11.12}$$

Using this kernel we can define a $(N+M)\times(N+M)$ matrix given the measurement and simulation points and a value for $\mathbf{t}_c$ as

$$\Sigma_\eta = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{t}_c, \mathbf{x}_1, \mathbf{t}_c) & k(\mathbf{x}_1, \mathbf{t}_c, \mathbf{x}_2, \mathbf{t}_c) & \ldots & k(\mathbf{x}_1, \mathbf{t}_c, \mathbf{x}_N, \mathbf{t}_c) & k(\mathbf{x}_1, \mathbf{t}_c, \mathbf{x}^*_1, \mathbf{t}^*_1) \ldots & k(\mathbf{x}_1, \mathbf{t}_c, \mathbf{x}^*_M, \mathbf{t}^*_M) \\ k(\mathbf{x}_2, \mathbf{t}_c, \mathbf{x}_1, \mathbf{t}_c) & k(\mathbf{x}_2, \mathbf{t}_c, \mathbf{x}_2, \mathbf{t}_c) & \ldots & k(\mathbf{x}_2, \mathbf{t}_c, \mathbf{x}_N, \mathbf{t}_c) & k(\mathbf{x}_2, \mathbf{t}_c, \mathbf{x}^*_1, \mathbf{t}^*_1) \ldots & k(\mathbf{x}_2, \mathbf{t}_c, \mathbf{x}^*_M, \mathbf{t}^*_M) \\ \vdots & & & & & \\ k(\mathbf{x}_N, \mathbf{t}_c, \mathbf{x}_1, \mathbf{t}_c) & k(\mathbf{x}_N, \mathbf{t}_c, \mathbf{x}_2, \mathbf{t}_c) & \ldots & k(\mathbf{x}_N, \mathbf{t}_c, \mathbf{x}_N, \mathbf{t}_c) & k(\mathbf{x}_N, \mathbf{t}_c, \mathbf{x}^*_1, \mathbf{t}^*_1) \ldots & k(\mathbf{x}_N, \mathbf{t}_c, \mathbf{x}^*_M, \mathbf{t}^*_M) \\ k(\mathbf{x}^*_1, \mathbf{t}^*_1, \mathbf{x}_1, \mathbf{t}_c) & k(\mathbf{x}^*_1, \mathbf{t}^*_1, \mathbf{x}_2, \mathbf{t}_c) & \ldots & k(\mathbf{x}^*_1, \mathbf{t}^*_1, \mathbf{x}_N, \mathbf{t}_c) & k(\mathbf{x}^*_1, \mathbf{t}^*_1, \mathbf{x}^*_1, \mathbf{t}^*_1) \ldots & k(\mathbf{x}^*_1, \mathbf{t}^*_1, \mathbf{x}^*_M, \mathbf{t}^*_M) \\ \vdots & & & & & \\ k(\mathbf{x}^*_M, \mathbf{t}^*_M, \mathbf{x}_1, \mathbf{t}_c) & k(\mathbf{x}^*_M, \mathbf{t}^*_M, \mathbf{x}_2, \mathbf{t}_c) & \ldots & k(\mathbf{x}^*_M, \mathbf{t}^*_M, \mathbf{x}_N, \mathbf{t}_c) & k(\mathbf{x}^*_M, \mathbf{t}^*_M, \mathbf{x}^*_M, \mathbf{t}^*_1) \ldots & k(\mathbf{x}^*_M, \mathbf{t}^*_M, \mathbf{x}^*_M, \mathbf{t}^*_M) \end{pmatrix}.$$

Given the assumptions that the simulation is replaced with a Gaussian process regression model and that the measurements have a normal uncertainty, and given values of the hyperparameters in Eq. (11.12) and a covariance for the measurement error, the vector $\mathbf{z}$ has a likelihood that is a multivariate normal PDF of the form

$$f(\mathbf{z}|\mathbf{t}_c, \beta_k, \lambda, \alpha, \Sigma_y) \propto |\Sigma_z|^{-1/2} \exp\left[-\frac{1}{2}\mathbf{z}^T \Sigma_z^{-1} \mathbf{z}\right]. \tag{11.13}$$

where $|\Sigma_z|$ is the determinant of the $(N+M)\times(N+M)$ matrix Σ_z

$$\Sigma_z = \Sigma_\eta + \begin{pmatrix} \Sigma_y & 0 \\ 0 & 0 \end{pmatrix}. \tag{11.14}$$

In this likelihood we have assumed that $\mathbf{z}$ is standardized to have mean zero. Using this likelihood function, we can specify a posterior for the calibration parameters *and* the Gaussian process regression hyperparameters as

$$\pi(\mathbf{t}_c, \beta_k, \lambda, \alpha|\mathbf{z}, \Sigma_y) \propto f(\mathbf{z}|\mathbf{t}_c, \beta_k, \lambda, \alpha, \Sigma_y)\pi(\mathbf{t}_c)\pi(\beta_k)\pi(\lambda)\pi(\alpha). \tag{11.15}$$

Therefore, if we can sample points from this posterior using MCMC, we perform calibration and build the emulator at the same time. The benefit of this approach is that measurement data is combined with simulation data in the construction of the emulator.

To perform MCMC sampling from the posterior in Eq. (11.15), we need to specify the prior distributions for the hyperparameters and the calibration parameters. For the calibration parameters, we can typically choose these based on valid limits for the models they represent, e.g., the parameter must be in some range or be positive, etc. It is common to set a flat, uniform prior for these variables if we have no preference for one value over another before we look at data. For the hyperparameters we follow the prescriptions of Higdon et al. (2004) and set

$$\pi(\lambda) \propto \lambda^{a-1} e^{-b\lambda} \tag{11.16}$$

$$\pi(\beta) \propto \prod_{k=1}^{p+q} \left(1 - e^{-\beta_k}\right)^{-\frac{1}{2}} e^{-\beta_k}, \tag{11.17}$$

with $a = b = 5$. We will set the power in the covariance function to be 2, i.e., $\alpha = 2$, and treat it as known. This is an assumption, but is often justified, if we believe the QoI should be a smooth function of the inputs.

In Metropolis-Hastings MCMC it is often convenient to work with the logarithm of probabilities. This is because the scale of the probability distributions can vary by orders of magnitude in scale. For numerical efficiency (and to avoid comparing very large or small numbers), we can work in terms of the logarithms so that

$$\begin{aligned} \log \pi(\mathbf{t}_c, \beta_k, \lambda, \alpha|\mathbf{z}, \Sigma_y) + \text{constant} &= \log f(\mathbf{z}|\mathbf{t}_c, \beta_k, \lambda, \alpha, \Sigma_y) \\ &+ \log \pi(\mathbf{t}_c) + \log \pi(\beta_k) + \log \pi(\lambda) + \log \pi(\alpha) \\ &= -\frac{1}{2}\log|\Sigma_z| - \frac{1}{2}\mathbf{z}^T\Sigma_z^{-1}\mathbf{z} \\ &+ (a-1)\log\lambda - b\lambda + \sum_{k=1}^{p+q}\left(\frac{-1}{2}\log\left(1 - e^{-\beta_k}\right) - \beta_k\right). \end{aligned} \tag{11.18}$$

Then in the MH algorithm, the value of the logarithm of the acceptance probability is found from Eq. (11.5) to be

$$\log \alpha(x_t, y) = \min\left(0, \log \hat{p}(y) + \log q(x_t|y) - \log \hat{p}(x_t) - \log q(y|x_t)\right).$$

Using this formulation, we then take the logarithm of a uniform random number between 0 and 1 and accept the proposal if that logarithm is less than the logarithm of the acceptance probability.

To make predictions from the calibrated model, we need to use the MCMC samples generated, after the burn-in, to construct a GP model using the algorithms of the previous chapter. One approach is to draw a sample from the Markov chain and use those values of the hyperparameters to construct a GP model using the data and make predictions. This is repeated, drawing new samples each time, several times to estimate the mean prediction and confidence intervals for the calibrated model. It is also useful to use calibrated parameters to then use in the execution of the simulation code for new predictions.

One important point we have not discussed is how to select the points at which to run the simulation, that is the $\mathbf{x}_i^*$ and $\mathbf{t}_i^*$. A space-filling design, orthogonal array, or pseudo-Monte Carlo technique, such as those discussed in Chap. 7 can be employed. These methods have the benefit of working in a batch mode: one determines the inputs at which to execute the simulation and then can run, in parallel, all the simulations. However, there are approaches to using adaptive sampling of the input space where a batch of simulations is run, a GPR is built, and points with the highest predicted uncertainty are then added as training points. This does limit the amount of batching that can be executed; however, it can greatly improve the number of simulation runs needed to perform calibration.

11.3.1 Application of Calibration on Real Data

We can apply this calibration model to the shock breakout data from Chap. 10. In that data set, the laser energy and disc thickness are experimental parameters, and the other three inputs (Be gamma, wall opacity, and flux limiter) are calibration parameters for approximate models in the calculation. Additionally, there are eight experimental measurements of the shock breakout time. Therefore, we can use these experiments to find appropriate posterior estimates for the calibration parameters. We do this using the priors specified above and 10^4 burn-in samples, and use a flat distribution over the range of the calibration parameters for the prior for the $\mathbf{t}$. In Fig. 11.5 we show the distribution of the β hyperparameters, and Fig. 11.6 compares the calibrated parameters with the prior distribution.

The results of the calibration indicate that the calibration parameters should be set to the lower end of their range to best agree with the experimental data. Furthermore, the disc thickness is the most important parameter in describing the shock breakout time, followed by the Be gamma and the flux limiter.

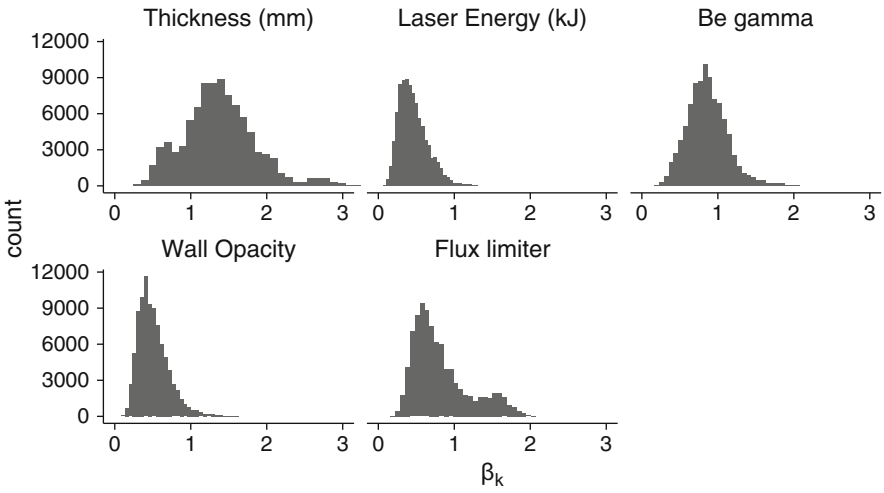

Fig. 11.5 The MCMC samples of the β_k for the five inputs to the simulation; the first two plots are x parameters, and the final three are calibration parameters

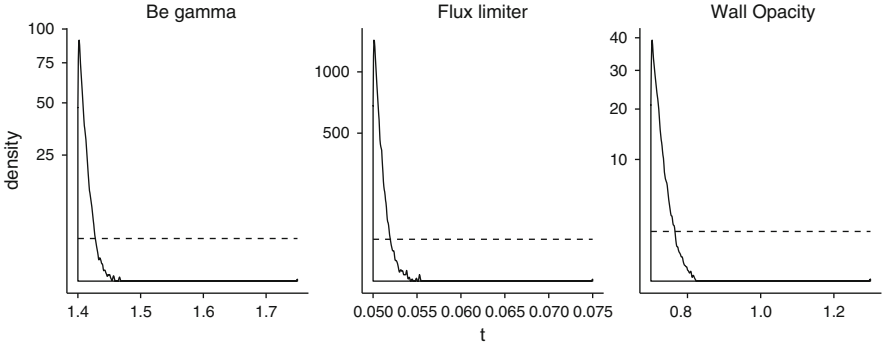

Fig. 11.6 The empirical density function from the MCMC samples of the three calibration parameters for the calibration problem. The flat prior distribution for these parameters is shown with a dashed line

11.4 The Kennedy-O'Hagan Predictive Model

The calibration procedure described above works when the computational model has the ability to reproduce the experimental results. Taking a cynical view of the calibration exercise, we could say that calibration only works if there are enough knobs in the code to turn to get the correct answer. In many cases we might know that the simulation is not an adequate representation of reality, and we want to develop a function for the difference between the code and the experimental results. That is we want to know how to correct the code to match an experiment.

We will use the predictive model originally proposed by Kennedy and O'Hagan (2000) and commonly called the Kennedy-O'Hagan model in a flourish of Hibernian appellation. The idea for the model is that we want to include a term that only depends on experimental parameters (the $\mathbf{x}$ from before), to allow for corrections to the computer model. To this end we write an experimental observation, $y(\mathbf{x}_i)$, as

$$y(\mathbf{x}_i) = \hat{\eta}(\mathbf{x}_i, \mathbf{t}_i) + \delta(\mathbf{x}_i) + \epsilon_i, \qquad i = 1, \ldots, N. \qquad \text{Kennedy-O'Hagan model}$$

The function $\delta(\mathbf{x}_i)$ is known as the discrepancy function. The question now is how to modify the calibration problem to estimate a GP model for the discrepancy and the simulation at the same time.

As before we combine the N measurements and M simulation results into a single vector $\mathbf{z}$ of size $N+M$. With this formulation the only change between fitting the Kennedy-O'Hagan model and the calibration problem is the specification of the covariance matrix Σ_z. For the predictive model, we include an $N \times N$ matrix Σ_δ:

$$\Sigma_z = \Sigma_\eta + \begin{pmatrix} \Sigma_y + \Sigma_\delta & 0 \\ 0 & 0 \end{pmatrix}. \tag{11.19}$$

The elements $(\Sigma_\delta)_{ij}$ are computed by evaluating a kernel function, $k_\delta(\mathbf{x}_i, \mathbf{x}_j)$, at the N inputs corresponding to the measurements. We can use the same form for this kernel function as we used previously:

$$k_\delta(\mathbf{x}, \mathbf{x}') = \frac{1}{\lambda_\delta}\left(\exp\left[-\sum_{k=1}^{p} \beta_k^{(\delta)} |x_k - x_k'|^{\alpha_\delta}\right]\right). \tag{11.20}$$

Given that we have introduced $p+1$ new hyperparameters, we need priors for λ_δ and $\beta_k^{(\delta)}$. Following Higdon et al. (2004) we use

$$\pi(\lambda_\delta) \propto \lambda_\delta^{a-1} e^{-b\lambda_\delta} \tag{11.21}$$

$$\pi(\beta^{(\delta)}) \propto \prod_{k=1}^{p+q} \left(1 - e^{-\beta_k^{(\delta)}}\right)^{-\frac{6}{10}} e^{-\beta_k}; \tag{11.22}$$

to give a flat prior for λ_δ, we set $a = 2$ and $b = 0.001$. The prior for $\beta^{(\delta)}$ is chosen to encourage the discrepancy function to be flatter than the model of the simulation as well; this is manifest in the power of $6/10$ compared with $1/2$ in the prior for β in the simulation covariance. The logic behind this choice is that we would prefer to match the experiment with the simulation (if possible) and make the discrepancy function small.

The question naturally arises regarding using the discrepancy to make a prediction. When I want to apply my code to a new experiment, how do I best apply the predictive model? When the new experiment is interpolation, that is the inputs

are inside the convex hull of previous data used to build the predictive model, the discrepancy function should be used to correct the simulation's prediction. However, extrapolating outside the previous experimental data requires care in applying the discrepancy function. For a point far outside the training data, the GP for the discrepancy function will return to the mean of the function, in this case zero. This should not be interpreted to imply that the simulation should be trusted to be correct in its prediction. In such an extrapolation, we can investigate how the discrepancy function varies for the known measurements: if the discrepancy function is small in magnitude, we can use this fact to give credence to the simulation predictions for extrapolation. Of course it will require expert judgment and considerations of epistemic uncertainty to be completely transparent with the uncertainties in the extrapolation, as we will discuss in a later chapter.

To make a prediction using the Kennedy-O'Hagan model, we have to modify the definition of $\mathbf{k}^*$ from Eq. (10.15) to include the kernel function for the discrepancy function. Each element of the vector is

$$(\mathbf{k}^*)_i = \begin{cases} k(\mathbf{x}_i, \mathbf{t}, \mathbf{x}^*, \mathbf{t}^*) + k_\delta(\mathbf{x}_i, \mathbf{x}^*), & i = 1, \ldots, N \\ k(\mathbf{x}_i, \mathbf{t}, \mathbf{x}^*, \mathbf{t}^*) & i = N+1, \ldots, N+M \end{cases}, \tag{11.23}$$

where $k(\mathbf{x}_i, \mathbf{t}, \mathbf{x}^*, \mathbf{t}^*)$ is the covariance kernel function for the simulations. The prediction requires this definition of $\mathbf{k}^*$ to inform the prediction vector that the covariance between the prediction should have a different form when compared with the simulation training points versus the measurement points. Equation (11.23) is used in Eq. (10.19) to produce the predictions for the predictive model. The evaluation of the expected simulation result from the predictive model can be accomplished by removing the $k_\delta(\mathbf{x}_i, \mathbf{x}^*)$ term from Eq. (11.23) to get a prediction without a discrepancy. This simulation prediction can then be used to evaluate the discrepancy function via subtraction from the full prediction.

11.4.1 Toy Example of Kennedy-O'Hagan Model

To demonstrate the behavior of the predictive model, we consider a simple simulation code that takes a single experimental input and a single calibration parameter given by the equation

$$\eta(x, t) = \sin xt.$$

We also consider measurements generated from the function

$$\begin{aligned} y(x) &= \sin 1.2x + 0.1x + \epsilon \\ &= \eta(x, 1.2) + 0.1x + \epsilon, \end{aligned} \tag{11.24}$$

where ϵ is a measurement error that is normally distributed with mean 0 and standard deviation 0.005. We will use the Kennedy-O'Hagan model to estimate the calibration parameter, which in this case has a true value of $t = 1.2$, and fit a discrepancy function. We know that the true discrepancy function is linear, and we can compare our estimate to the true function.

To build the model, we generated 10 measurements by sampling x from a standard normal distribution using stratified sampling. We also sample the simulation at 40 points using 2-D Latin hypercube sampling of the standard normal for x and a normal variable with mean 1 and standard deviation 0.2 for the t variable. We set $\alpha = 2$ for both the simulator and discrepancy covariance functions. For the calibration parameter, t, we set a flat prior meaning that the value can take on any value with equal likelihood.

We generated 10^4 MCMC samples after a burn-in period of 10^4 samples to fit the predictive model for this data. The total MCMC chains are shown in Fig. 11.7. In this problem the chain centers on the correct value of t in a small number of samples; we also see that the value for β_x is the largest indicating that x is the most important variable as estimated by the model. Furthermore, as suggested by the prior, the estimate for λ_δ is larger than λ for the simulation, indicating that the model was putting more emphasis on making the GP for the simulations match the data rather than making the discrepancy function larger.

To test our predictive model, we generate a test set of 20 new measurements at x values sampled from the uniform distribution between ± 3. To make the predictions, we select 100 samples from the MCMC chain to get the hyperparameters and an estimate for t to produce a prediction for each x in the test set. The results are shown in Fig. 11.8 where the predictions are plotted versus measured values. In this figure we denote the mean of the 100 predictions from the MCMC samples with a point and the range of the samples with an error bar; the measurement uncertainty is smaller than the width of the points. We see that for most of the test set, the predictive model can reproduce the measurement values. There are, however, several predictions that have large estimated uncertainties and do not agree with the measurements.

To understand these inaccurate points, we look at the predictions as a function of x compared with the true underlying function without measurement error in Fig. 11.9. The figure gives the prediction at a range of x values between ± 2.5 using 10 samples from the MCMC chain; the dashed lines are the range of the predictions. We can see that when $x \in [-2, 2]$, the model and the true function are indistinguishable in the figure. This represents the range of the training data, a fact that can be confirmed by looking at the particular training points sampled. The conclusion that we can draw is that the predictive model is very strong at interpolating between known data points, but extrapolation is problematic.

We can also see how the discrepancy function and the GP for the simulation behave as a function of x using the same procedure. In Fig. 11.10a the simulation using 100 samples from the Markov chain is compared with an exactly calibrated simulator. As before, outside the range of the data, the model performs poorly. The estimated discrepancy function, shown in Fig. 11.10b, has a mean that matches

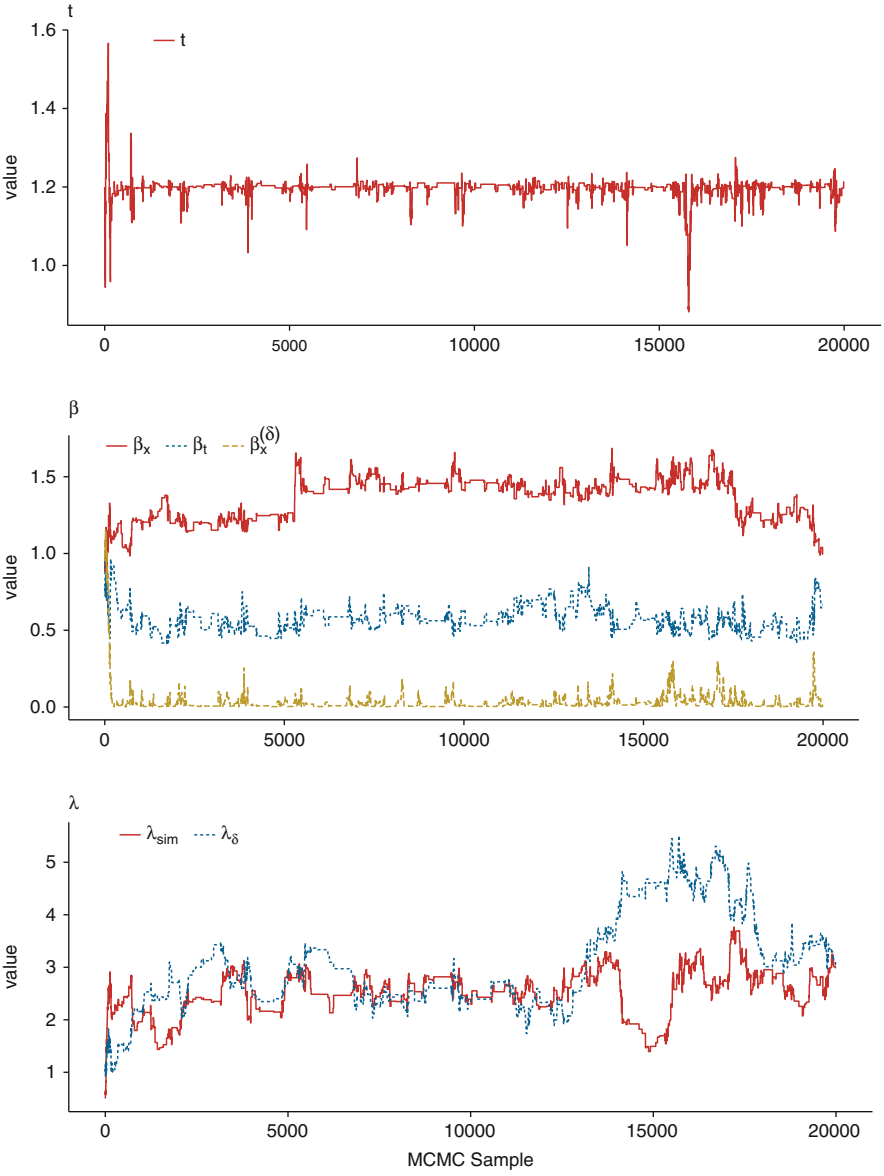

Fig. 11.7 The MCMC samples of the calibration parameter t and the hyperparameters. The burn-in period used was 10^4

the true discrepancy between -2 and 2 with large error outside this range. We do notice that the uncertainty in the discrepancy function is noticeable near $x = \pm 1$. This uncertainty in the discrepancy is mirrored in the simulation estimates, though due to the scale is less noticeable. In this case we have compensating errors

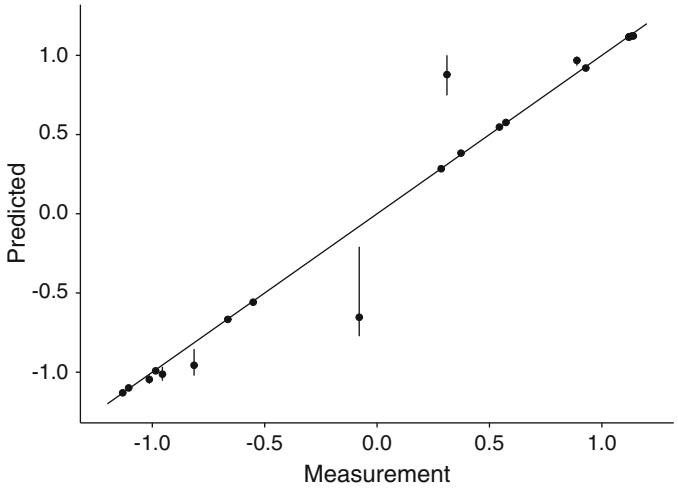

Fig. 11.8 Prediction from the predictive model versus actual at 20 new measurements generated from Eq. (11.24). Each point represents the mean of the estimate generated using 100 different samples from the MCMC chain, and the error bars give the range of those estimates

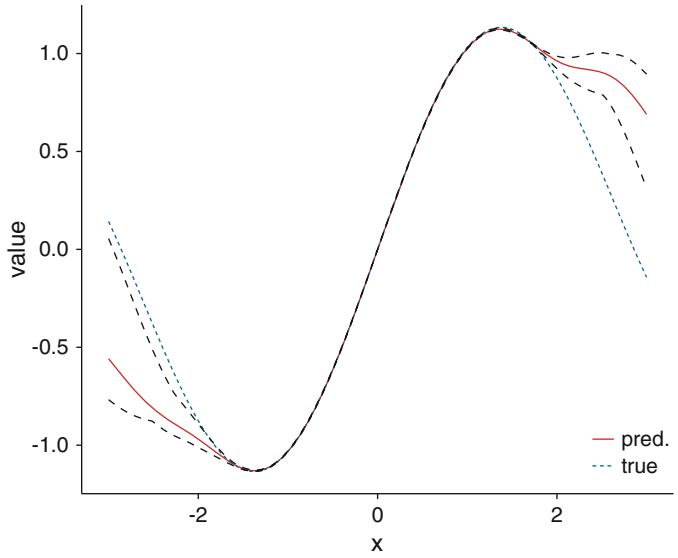

Fig. 11.9 Prediction from the predictive model as a function of x and the underlying true function from Eq. (11.24). The predicted curve represents the mean of the estimate generated using 10 different samples from the Markov chain, and the dashed lines give the range of those estimates

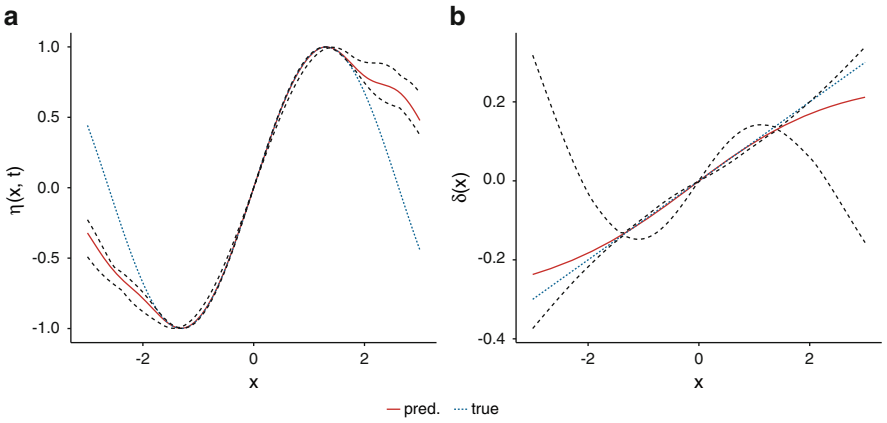

Fig. 11.10 The estimated simulator response $\eta(x, t)$ from the predictive model compared with the function $\sin 1.2x$ (left) and the discrepancy function estimated by the predictive model compared with the true discrepancy (right). The dashed lines represent the range of estimates produced from 100 samples from the Markov chain. (**a**) Simulation. (**b**) Discrepancy

in the simulation and discrepancy estimates: if the simulation is too high, the discrepancy can be decreased to compensate for the error. These errors cancel when the simulation and discrepancy are added to get an overall prediction.

Though this is a simple example, it does point out some important features of predictive models in terms of extrapolation and how the discrepancy function, and GP for the simulation can have compensating effects on the prediction. These phenomena occur beyond just toy problems. We will return to this example later when we want to have a multi-fidelity model.

11.5 Hierarchical Models

In scientific computing we often have a range of models with varying degrees of fidelity to apply to a problem. For example, we may have an analytic model that is known to be an order-of-magnitude approximation, an approximate ODE model for the behavior, and a full 3-D, time-dependent PDE model. In this scenario we may be able to only run the high-fidelity model a few times and the low-fidelity model more, perhaps many more, times, and evaluate the analytic model an arbitrary number of times. We would like to create a predictive model for this type of scenario. This scenario was investigated by Goh et al. (2013).

We consider the case of two simulations to compute a QoI: a high-fidelity simulation $\eta_{\mathrm{H}}(\mathbf{x}_i, \mathbf{t}_i^{\mathrm{H}}, \mathbf{t}_i^{\mathrm{s}})$ and a low-fidelity simulation $\eta_{\mathrm{L}}(\mathbf{x}_i, \mathbf{t}_i^{\mathrm{L}}, \mathbf{t}_i^{\mathrm{s}})$. Notice that there can be different calibration parameters which are different for the simulations, the $\mathbf{t}_i^{\mathrm{L}}$ and $\mathbf{t}_i^{\mathrm{H}}$, as well as shared calibration parameters $\mathbf{t}_i^{\mathrm{s}}$. We begin with the case where we have N experimental measurements, M_{H} evaluations of the high-fidelity

simulation, and M_{L} evaluations of the low-fidelity simulator. It is typically the case that $N \ll M_{\mathrm{H}} \ll M_{\mathrm{L}}$. In this scenario our statistical model is

$$\begin{aligned}
\mathbf{y}(\mathbf{x}_i) &= \hat{\eta}_{\mathrm{L}}(\mathbf{x}_i, \mathbf{t}_{\mathrm{c}}^{\mathrm{L}}, \mathbf{t}_{\mathrm{c}}^{\mathrm{s}}) + \delta(\mathbf{x}_i) + \delta_{\mathrm{L}}(\mathbf{x}_i, \mathbf{t}_{\mathrm{c}}^{\mathrm{H}}, \mathbf{t}_{\mathrm{c}}^{\mathrm{s}}) + \epsilon_i, \qquad i = 1, \ldots, N, \\
\boldsymbol{\eta}_{\mathrm{H}}(\mathbf{x}_i, \mathbf{t}_i^{\mathrm{H}}, \mathbf{t}_i^{\mathrm{s}}) &= \hat{\eta}_{\mathrm{L}}(\mathbf{x}_i, \mathbf{t}_{\mathrm{c}}^{\mathrm{L}}, \mathbf{t}_i^{\mathrm{s}}) + \delta_{\mathrm{L}}(\mathbf{x}_i, \mathbf{t}_i^{\mathrm{H}}, \mathbf{t}_i^{\mathrm{s}}), \qquad i = N+1, \ldots, N+M_{\mathrm{H}}, \\
\boldsymbol{\eta}_{\mathrm{L}}(\mathbf{x}_i, \mathbf{t}_i^{\mathrm{L}}, \mathbf{t}_i^{\mathrm{s}}) &= \hat{\eta}_{\mathrm{L}}(\mathbf{x}_i, \mathbf{t}_i^{\mathrm{L}}, \mathbf{t}_i^{\mathrm{s}}), \qquad i = N+M_{\mathrm{H}}+1, \ldots, N+M_{\mathrm{H}}+M_{\mathrm{L}}.
\end{aligned} \tag{11.25}$$

In this model the subscript c denotes calibrated quantities, and the hats over the η functions represent a GP regression approximation to the simulation.

The form of the multi-fidelity predictive model indicates that we calibrate the low-fidelity model and compute a discrepancy to match the high-fidelity model, and then the high-fidelity model is calibrated along with a discrepancy function to match the measurements. There is the complication that the calibration parameters for the two models are generally different. Therefore, in order for the low-fidelity model to approximate the high-fidelity simulation, we must include these parameters in the discrepancy function.

We construct a single vector to hold the measurements and two types of simulation data: $\mathbf{z}$ will be a length $N + M_{\mathrm{H}} + M_{\mathrm{L}}$ vector containing the left-hand side of Eq. (11.25). We then write the covariance matrix for the data as

$$\Sigma_z = \Sigma_{\eta_{\mathrm{L}}} + \begin{pmatrix} \Sigma_y + \Sigma_\delta & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \Sigma_{\mathrm{L}} & 0 \\ 0 & 0 \end{pmatrix}; \tag{11.26}$$

$\Sigma_{\eta_{\mathrm{L}}}$, the low-fidelity covariance, is a square matrix with size $N + M_{\mathrm{H}} + M_{\mathrm{L}}$ found by evaluating $k(\mathbf{x}_i, \mathbf{x}_j)$ at all of the points in the data set; Σ_y is the square matrix that contains the covariances between the N measurements; Σ_δ is the square matrix of size N found by evaluating $k_\delta(\mathbf{x}_i, \mathbf{x}_j)$ at the N measurement points; Σ_{L}, the low-fidelity discrepancy covariance, is the size $N + M_{\mathrm{H}}$ square matrix found by evaluating $k_{\mathrm{L}}(\mathbf{x}_i, \mathbf{x}_j)$ at the measurement and high-fidelity simulation points.

The kernel covariance functions will take the same form as before, except now we have more hyperparameters. Denote the number of experimental variables as p, the length of $\mathbf{t}^{\mathrm{H}}$ as q_{H}, the length of $\mathbf{t}^{\mathrm{L}}$ as q_{L}, and the length of $\mathbf{t}^{\mathrm{s}}$ as r. The kernels required to evaluate Eq. (11.26) are then

$$\begin{aligned}
k(\mathbf{x}, \mathbf{t}^{\mathrm{L}}, \mathbf{t}^{\mathrm{s}}, \mathbf{x}', \mathbf{t}^{\mathrm{L}'}, \mathbf{t}^{\mathrm{s}'}) = \frac{1}{\lambda} \Bigg(& \exp\left[-\sum_{k=1}^{p} \beta_k |x_k - x_k'|^\alpha \right] \\
& + \exp\left[-\sum_{k=1}^{q_{\mathrm{L}}} \beta_{k+p} |t_k^{\mathrm{L}} - t_k^{\mathrm{L}'}|^\alpha \right] \\
& + \exp\left[-\sum_{k=1}^{r} \beta_{k+p+q_{\mathrm{L}}} |t_k^{\mathrm{s}} - t_k^{\mathrm{s}'}|^\alpha \right] \Bigg),
\end{aligned} \tag{11.27a}$$

$$k_{\mathrm{L}}(\mathbf{x}, \mathbf{t}^{\mathrm{H}}, \mathbf{t}^{\mathrm{s}}, \mathbf{x}', \mathbf{t}^{\mathrm{H}'}, \mathbf{t}^{\mathrm{s}'}) = \frac{1}{\lambda_{\mathrm{L}}} \left(\exp\left[-\sum_{k=1}^{p} \beta_k^{(\mathrm{L})} |x_k - x_k'|^{\alpha_{\mathrm{L}}} \right] + \exp\left[-\sum_{k=1}^{q_{\mathrm{H}}} \beta_{k+p}^{(\mathrm{L})} |t_k^{\mathrm{H}} - t_k^{\mathrm{H}'}|^{\alpha_{\mathrm{L}}} \right] + \exp\left[-\sum_{k=1}^{r} \beta_{k+p+q_{\mathrm{H}}}^{(\mathrm{L})} |t_k^{\mathrm{s}} - t_k^{\mathrm{s}'}|^{\alpha_{\mathrm{L}}} \right] \right), \tag{11.27b}$$

$$k_{\delta}(\mathbf{x}, \mathbf{x}') = \frac{1}{\lambda_{\delta}} \left(\exp\left[-\sum_{k=1}^{p} \beta_k^{(\delta)} |x_k - x_k'|^{\alpha_{\delta}} \right] \right). \tag{11.27c}$$

There are a total of $(p + q_{\mathrm{L}} + r) + (p + q_{\mathrm{H}} + r) + p$ β hyperparameters, 3 λ hyperparameters, and 3 α hyperparameters.

For the prior distributions, we use Eq. (11.16) for the priors of the hyperparameters in Eq. (11.27a) and use the priors in Eq. (11.21) for the hyperparameters in Eqs. (11.27b) and (11.27c). In our examples below, we assume that the α parameters are known. To estimate these parameters and the calibration parameters, we use MCMC sampling to generate samples from the posterior distribution

$$\begin{aligned} &\pi(\mathbf{t}_{\mathrm{c}}^{\mathrm{L}}, \mathbf{t}_{\mathrm{c}}^{\mathrm{H}}, \mathbf{t}_{\mathrm{c}}^{\mathrm{s}}, \beta, \lambda, \alpha | \mathbf{z}, \Sigma_y) \\ &\quad \propto f(\mathbf{z} | \mathbf{t}_{\mathrm{c}}^{\mathrm{L}}, \mathbf{t}_{\mathrm{c}}^{\mathrm{H}}, \mathbf{t}_{\mathrm{c}}^{\mathrm{s}}, \beta, \lambda, \alpha, \Sigma_y) \pi(\mathbf{t}_{\mathrm{c}}^{\mathrm{L}}, \mathbf{t}_{\mathrm{c}}^{\mathrm{H}}, \mathbf{t}_{\mathrm{c}}^{\mathrm{s}}) \pi(\beta_k) \pi(\lambda) \pi(\alpha). \end{aligned} \tag{11.28}$$

with the likelihood function given by

$$f(\mathbf{z} | \mathbf{t}_{\mathrm{c}}^{\mathrm{L}}, \mathbf{t}_{\mathrm{c}}^{\mathrm{H}}, \mathbf{t}_{\mathrm{c}}^{\mathrm{s}}, \beta, \lambda, \alpha, \Sigma_y) \propto |\Sigma_z|^{-1/2} \exp\left[-\frac{1}{2} \mathbf{z}^{\mathrm{T}} \Sigma_z^{-1} \mathbf{z} \right], \tag{11.29}$$

where we have abused notation to write all the β, λ, and α hyperparameters using a single variable each.

It would be possible to extend the hierarchical model to include further levels in a straightforward, if notationally messy, manner. Additionally, one can develop a predictive model that admits several models that do not necessarily have a known hierarchy. In such a model, we may not know which computational model is better, but we would like to use simulation data from each model to make predictions. Such predictive models were studied by Goh (2014).

11.5.1 Prediction with an Inexpensive Low-Fidelity Model

If the low-fidelity model can be evaluated an arbitrary number of times or on demand, we can simplify the multi-fidelity model greatly. Such a low-fidelity model

might be an analytic approximation or a code that can execute in a time comparable to the evaluation of the likelihood function. In this instance we do not need to fit a GP emulator for the low-fidelity model. Rather we specify the model as

$$\begin{aligned}\mathbf{y}(\mathbf{x}_i) &= \eta_{\mathrm{L}}(\mathbf{x}_i, \mathbf{t}_{\mathrm{c}}^{\mathrm{L}}, \mathbf{t}_{\mathrm{c}}^{\mathrm{s}}) + \delta(\mathbf{x}_i) + \delta_{\mathrm{L}}(\mathbf{x}_i, \mathbf{t}_{\mathrm{c}}^{\mathrm{H}}, \mathbf{t}_{\mathrm{c}}^{\mathrm{s}}) + \epsilon_i, \qquad i = 1, \ldots, N, \\ \boldsymbol{\eta}_{\mathrm{H}}(\mathbf{x}_i, \mathbf{t}_i^{\mathrm{H}}, \mathbf{t}_i^{\mathrm{s}}) &= \eta_{\mathrm{L}}(\mathbf{x}_i, \mathbf{t}_{\mathrm{c}}^{\mathrm{L}}, \mathbf{t}_i^{\mathrm{s}}) + \delta_{\mathrm{L}}(\mathbf{x}_i, \mathbf{t}_i^{\mathrm{H}}, \mathbf{t}_i^{\mathrm{s}}), \qquad i = N+1, \ldots, N+M_{\mathrm{H}}.\end{aligned} \tag{11.30}$$

We then define the vector $\mathbf{z}$ to have just the measurements and the high-fidelity simulations making it a $N + M_{\mathrm{H}}$ vector. The covariance matrix for the model becomes

$$\Sigma_z = \Sigma_{\mathrm{L}} + \begin{pmatrix} \Sigma_y + \Sigma_\delta & 0 \\ 0 & 0 \end{pmatrix}. \tag{11.31}$$

The posterior distribution is no longer based on a mean-zero Gaussian process. The likelihood function becomes

$$f(\mathbf{z}|\mathbf{t}_{\mathrm{c}}^{\mathrm{L}}, \mathbf{t}_{\mathrm{c}}^{\mathrm{H}}, \mathbf{t}_{\mathrm{c}}^{\mathrm{s}}, \beta, \lambda, \alpha, \Sigma_y) \propto |\Sigma_z|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{z} - \mathbf{z}_{\mathrm{L}})^{\mathrm{T}} \Sigma_z^{-1} (\mathbf{z} - \mathbf{z}_{\mathrm{L}})\right], \tag{11.32}$$

where $\mathbf{z}_{\mathrm{L}}$ is a vector containing the low-fidelity model evaluated at $\mathbf{x}_i$, $\mathbf{t}_{\mathrm{c}}^{\mathrm{L}}$, and $\mathbf{t}_{\mathrm{c}}^{\mathrm{s}}$ for $i = 1, \ldots, N$ and $\mathbf{x}_i$, $\mathbf{t}_i^{\mathrm{L}}$, and $\mathbf{t}_i^{\mathrm{s}}$ for $i = N+1, \ldots, N+M_{\mathrm{H}}$. Notice that each time we evaluate the likelihood, we will have to evaluate the low-fidelity model N times. The posterior distribution is then given by Eq. (11.28), as before.

11.5.2 Example Hierarchical Model

To demonstrate the application of a hierarchical, multi-fidelity model, we consider a modification of the toy problem from Sect. 11.4.1. In this case the low-fidelity model will be a Taylor series expansion of the high-fidelity simulation function. The high-fidelity model is given by

$$\eta_{\mathrm{H}}(x, t^{\mathrm{H}}, t^{\mathrm{s}}) = \sin\left(x t^{\mathrm{s}} + t^{\mathrm{H}}\right).$$

The low-fidelity model is a Taylor series of the high-fidelity model with two additional calibration parameters:

$$\eta_{\mathrm{L}}(x, t_1^{\mathrm{L}}, t_2^{\mathrm{L}}, t^{\mathrm{s}}) = t_1^{\mathrm{L}} + t^{\mathrm{s}} t_2^{\mathrm{L}} x - \frac{1}{2}\left(t^{\mathrm{s}}\right)^2 t_1^{\mathrm{L}} x^2 - \frac{1}{6}\left(t^{\mathrm{s}}\right)^3 t_2^{\mathrm{L}} x^3.$$

Note that the Taylor series is correct if $t_1^{\mathrm{L}} = \sin t^{\mathrm{H}}$ and $t_2^{\mathrm{L}} = \cos t^{\mathrm{H}}$; these are the values we expect to recover in the calibration procedure. The measurements are generated from

$$\begin{aligned} y(x) &= \sin(1.2x + 0.1) + 0.1x + \epsilon \\ &= \eta_{\mathrm{H}}(x, 0.1, 1.2) + 0.1x + \epsilon, \end{aligned} \tag{11.33}$$

where ϵ is a measurement error that is normally distributed with mean 0 and standard deviation 0.005. In this example, there needs to be a discrepancy function to correct the high-fidelity simulation, and there is a discrepancy function needed to make the low-fidelity simulation agree with the high-fidelity model. Both of these discrepancy functions can be written down analytically.

To build the model, we generated 10 measurements by sampling x from a standard normal distribution using stratified sampling. We also sample the simulation at 40 points using 5-D Latin hypercube sampling of $\mathscr{U}(-2, 2)$for x and a normal variable with mean 1 and standard deviation 0.2 for the t^{s} and t_2^{L} variables, and a standard normal with mean 0 and standard deviation 0.2 for each of the t^{H} and t_1^{L} variables. We set $\alpha = 2$ for both the simulator and discrepancy covariance functions. For the calibration parameters, we set a flat prior meaning that the value can take on any value with equal likelihood. Additionally, we assume that the low-fidelity model can be evaluated an arbitrary number of times so that we can apply the techniques of Sect. 11.5.1.

We generated 10^4 MCMC samples after a burn-in period of 10^4 samples to fit the predictive model for this data. The total MCMC chains are shown in Fig. 11.11. In contrast to the non-hierarchical model, the calibration parameters seem to vary from their true values. As we will see, this is due to the fact that these parameters can compensate for each other.

To test our predictive model we generate a test set of 20 new measurements at x values sampled from the uniform distribution between ± 3. To make the predictions we select 100 samples from the MCMC chain to get the hyperparameters and estimates for the calibration parameters and produce prediction for each x in the test set. The results are shown in Fig. 11.12 where the predictions are plotted versus measured values. In this figure we denote the mean of the 100 predictions from the MCMC samples with a point and the range of the samples with an error bar; the measurement uncertainty is smaller than the width of the points. As in the previous predictive model example, most of the predictions match the measurements except for a few. These are extrapolations as before, though there error is larger in this case.

The predicted value of the measurement as a function of x is shown in Fig. 11.13. In the hierarchical model, the errors due to extrapolation are much larger than the in the standard Kennedy-O'Hagan model case from before. This is due to the fact that we have to estimate the extrapolation values of two discrepancy functions in this case. Nevertheless, the results for smaller magnitudes of x are in line with the true values.

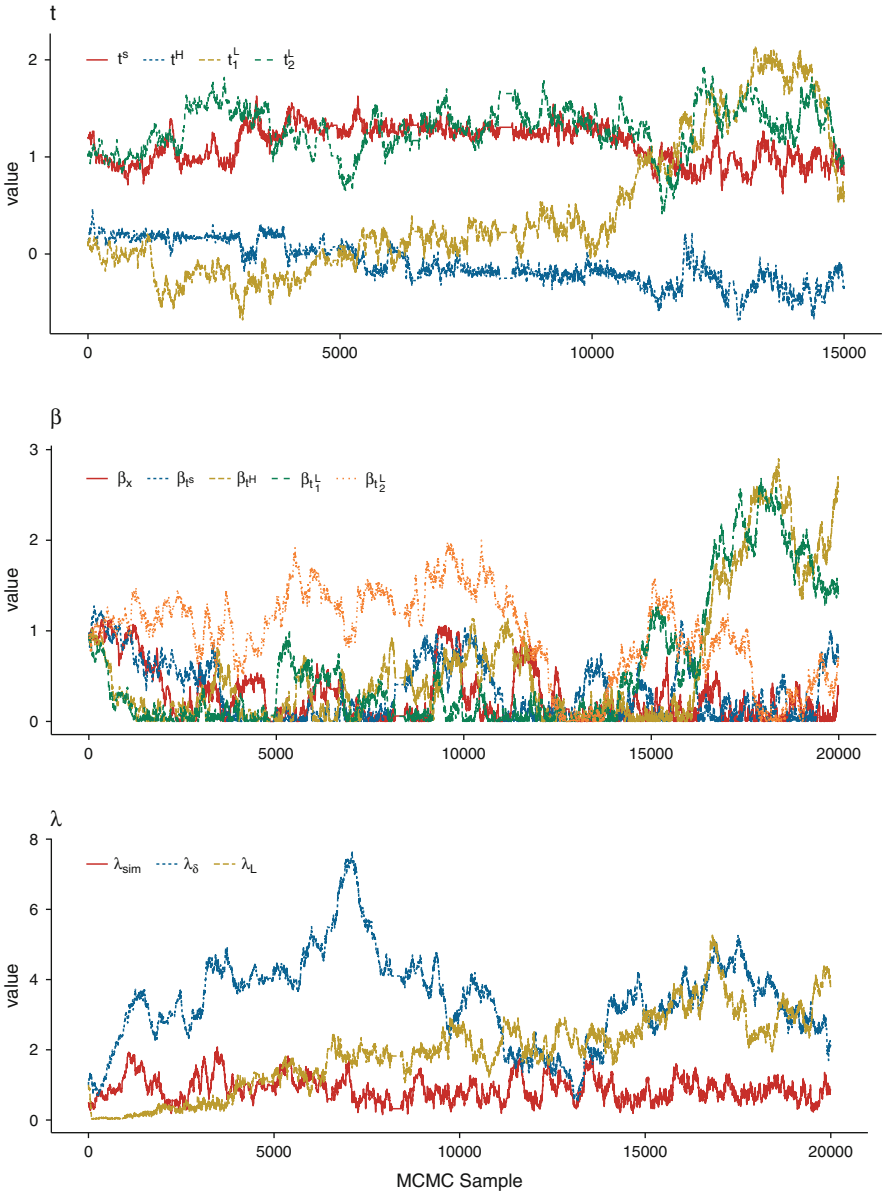

Fig. 11.11 The MCMC samples of the calibration parameter t and the hyperparameters for the hierarchical model. The burn-in period used was 10^4

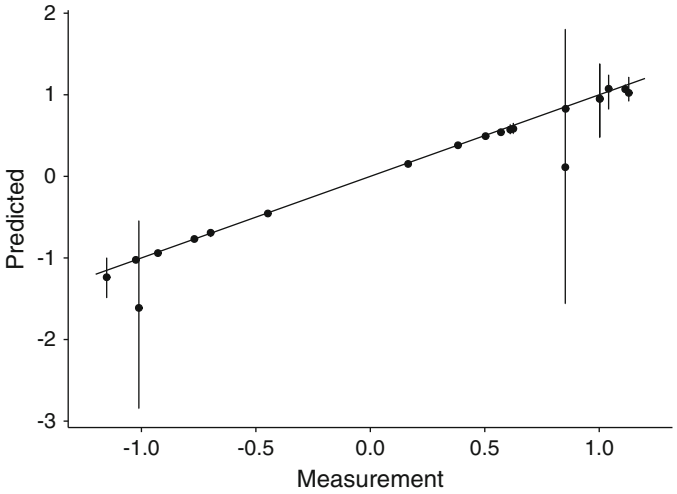

Fig. 11.12 Prediction from the hierarchical predictive model versus actual at 20 new measurements generated from Eq. (11.33). Each point represents the mean of the estimate generated using 100 different samples from the MCMC chain, and the error bars give the range of those estimates

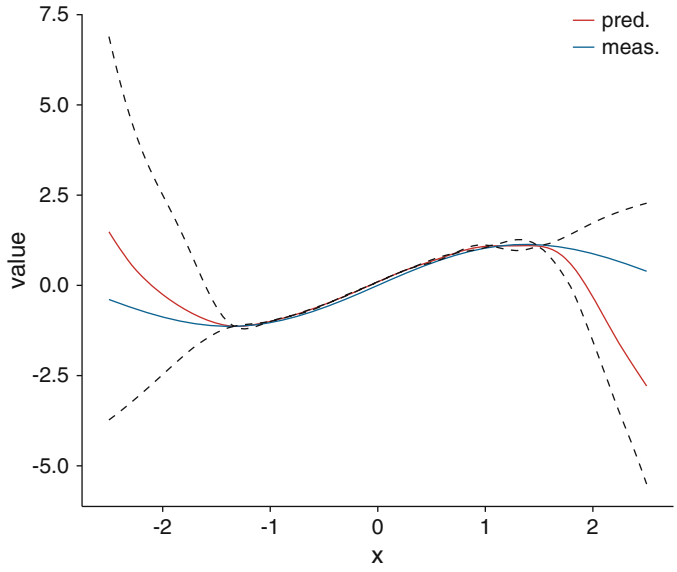

Fig. 11.13 Prediction from the hierarchical predictive model as a function of x and the underlying true function from Eq. (11.33). The predicted curve represents the mean of the estimate generated using 10 different samples from the Markov chain, and the dashed lines give the range of those estimates

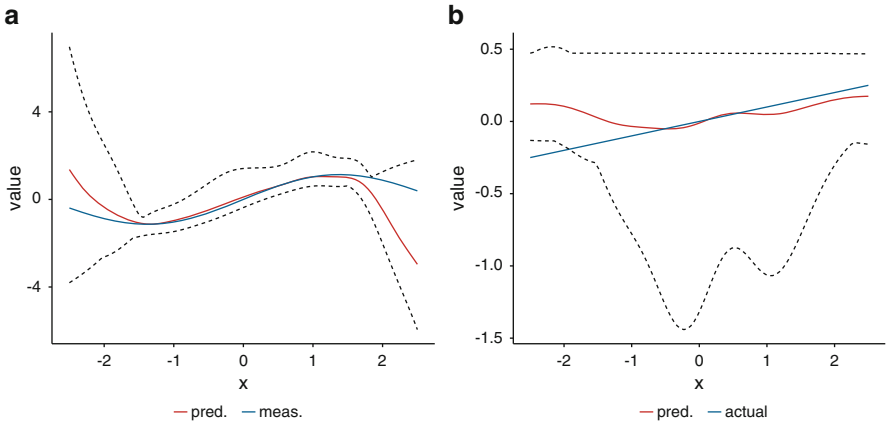

Fig. 11.14 The estimated simulator response for $\eta_{\mathrm{H}}(x, t^{\mathrm{H}}, t^{\mathrm{s}})$ from the hierarchical predictive model compared with the function $\sin(1.2x + 0.1)$ (left), and the discrepancy function, $\delta(x)$ estimated by the predictive model compared with the true discrepancy (right). The dashed lines represent the range of estimates produced from 100 samples from the Markov chain. (**a**) Simulation. (**b**) Discrepancy

The predictive model can be used to estimate the high-fidelity model output at the calibrated inputs, as shown in Fig. 11.14a. Here we can see that the estimate is accurate inside the range of the training data, though the predictions are slightly high for $x \in [-1, 0]$ and low between 1 and 2. For the discrepancy function, the result does not match the true linear discrepancy except near $x = 0$. Additionally, the uncertainty in the discrepancy estimate is much larger than in the previous, single-level predictive model.

Despite the inaccuracy in producing the discrepancy function for this data, the hierarchical predictive model excels at its designed goal: to make a prediction for the measurement. This is noteworthy because we asked the model to accomplish four tasks in a single MCMC procedure: estimate a discrepancy function, calibrate parameters from low- and high-fidelity models, as well as estimate a GP emulator for the high-fidelity model.

11.6 Notes and References

The models that we demonstrated in this chapter all used the same kernel for the covariance function. In the literature there are other covariances commonly used. One to note takes the form

$$k(\mathbf{x}, \mathbf{x}') = \frac{1}{\lambda} \prod_{k=1}^{p} \rho_k^{4(x_k - x_k')^2}, \qquad \rho_k > 0.$$

In this function the smaller the value of ρ_k, the more important the parameter. As a prior for ρ_k, a flat prior with mean near 1 is usually used. This covariance function is widely used; we chose a single function in our examples for ease of exposition.

In addition to the references mentioned above, the use of predictive models can be found in other papers. Holloway et al. (2011) and Gramacy et al. (2015) used a Kennedy-O'Hagan model on the modeling of a radiating shock experiment, Karagiannis and Lin (2017) combined several types of simulation of unknown fidelity to make predictions, Zheng and McClarren (2016) used multiple physical models to calibrate neutron scattering data, and Bayarri et al. (2007) combined a predictive model with wavelet decompositions of functional data. This list is not exhaustive, but does point to papers that can be readily understood with the tools discussed in this chapter.

11.7 Exercises

Problems 1 and 2 deal with an example from Goh et al. (2013):

1. Construct a hierarchical predictive model where the low-fidelity simulations are given by

$$\eta_{\mathrm{L}}(\mathbf{x}, t^{\mathrm{L}}, t^{\mathrm{s}}) = \left(1 - \exp\left(\frac{-1}{2x_2}\right)\right) \frac{1000t^{\mathrm{s}}x_1^3 + 1900x_1^2 + 2092x_1 + 60}{1000t^{\mathrm{L}}x_1^3 + 500x_1^2 + 4x_1 + 20},$$

   the high-fidelity simulations are given by

$$\eta_{\mathrm{H}}(\mathbf{x}, t^{\mathrm{H}}, t^{\mathrm{s}}) = \eta_{\mathrm{L}}(\mathbf{x}, 0.1, t^{\mathrm{s}}) + 5e^{-t^{\mathrm{s}}} \frac{x_1^{t^{\mathrm{H}}}}{100(x_2^{2+t^{\mathrm{H}}} + 1)},$$

   and the measurements are

$$y(\mathbf{x}) = \eta_{\mathrm{H}}(\mathbf{x}, 0.3, 0.2) + \frac{10x_1^2 + 4x_2^2}{50x_1x_2 + 10} + \epsilon,$$

   where $\epsilon \sim \mathcal{N}(\mu = 0, \sigma^2 = 0.5^2)$. All parameters and the components of $\mathbf{x}$ are said to lie on the unit interval. Use 5 measurements, 10 high-fidelity simulations, and 40 low-fidelity simulations to build a predictive model. Use the model to make predictions at 10 new points. Additionally, compare your predictions for the high-fidelity model to the actual high-fidelity model and demonstrate the accuracy (or inaccuracy) of your discrepancy functions. How well does the model calibrate the values of t in the model?
2. Repeat the previous exercise by considering only the high-fidelity model and the measurements.

3. You perform a measurement of a beam of radiation hitting a slab and somehow are able to measure the particle intensity, $\phi(x)$ at $x = 1, 1.5, 3, 5$:

$$\phi(1) = 0.201131, \quad \phi(1.5) = 0.110135, \quad \phi(3) = 0.0228748,$$
$$\phi(5) = 0.00328249.$$

Using the simple model for the particle intensity $\phi(x) = E_2(\sigma x)$, where

$$\mathrm{E}_n(x) = \int_1^\infty \frac{e^{-xt}}{t^n}\, dt.$$

With $\sigma \sim \mathcal{G}(8, .1)$, and the experimental data just given, derive a posterior distribution for σ (i.e., calibrate σ). You may assume that the measurement has an error distributed by $N(0, \sigma = 0.001)$. Do your answers change if you add a discrepancy function?

Chapter 12
Epistemic Uncertainties: Dealing with a Lack of Knowledge

I think I was aware that something had happened, but I'm not fully aware.

—Brady Hoke

In this chapter we change how we interpret uncertain variables from having a known distribution to instead reflect that certain random variables may not have a known distribution. These uncertainties could arise from, for example, model error, discretization error, or the choices an analyst makes in specifying parameters in input random variables (e.g., assuming that the distribution was normal). All of these errors arise from a lack of knowledge and are therefore called *epistemic* uncertainties: if the model or numerical solution has error, we do not know what that error is. We may be able to bound the error, and this is the goal of solution verification in scientific simulation. Additionally, these epistemic uncertainties arise when we extrapolate models (both simulation and predictive models) beyond known experimental data. When dealing with an uncertainty where we only know bounds, we cannot simply assume that the uncertainty is a uniform distribution. With epistemic uncertainties, such as the numerical error, there is a value—we simply do not know its value. Therefore, we should treat the uncertainties differently.

Figure 12.1 shows a variable x that we only know takes on values in the interval $[a, b]$. If we interpret this to mean that x is uniformly distributed between a and b, the resulting distribution of a QoI that is a function of x is based on the fact that every point within the interval is equally likely. However, a highly peaked distribution inside the interval is possible, and the resulting distribution of the QoI could be very different. In the figure we see that the actual QoI distribution is peaked in the tail of the distribution when x is uniform.

12.1 Model Uncertainty and the L_1 Validation Metric

Before we include epistemic uncertainties, we introduce the idea of a validation metric based on the L_1 norm. We consider a QoI that has aleatory uncertainties that generate a CDF for the QoI. This CDF could be produced using any of the

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0_12

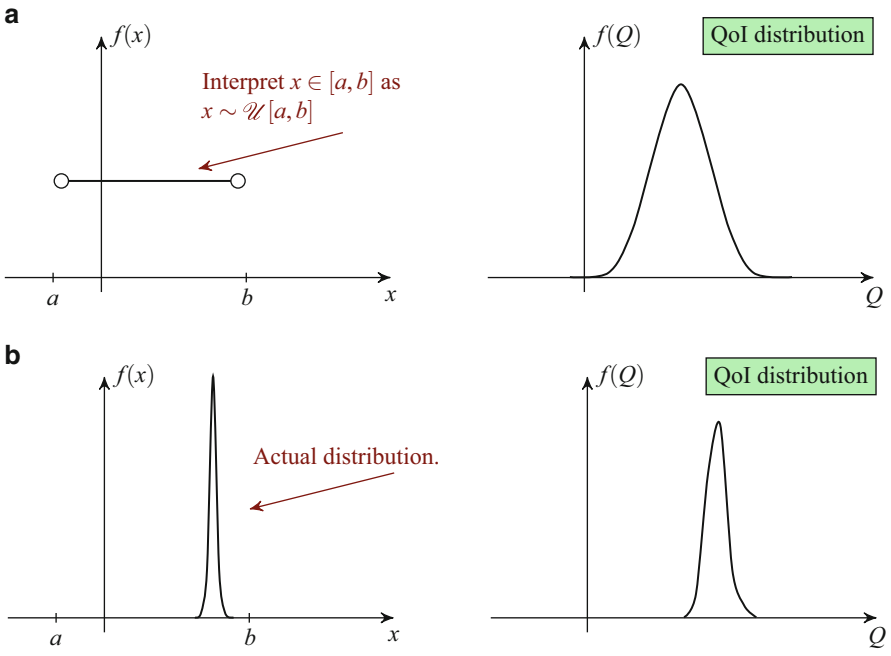

Fig. 12.1 The different results that can be obtained when an interval uncertainty is interpreted as a uniform distribution (**a**). In reality the distribution of x is highly peaked in the right part of the interval leading to a QoI distribution peaked in the tail of the distribution inferred from a uniform distribution. (**b**) Actual distribution is not uniform, but does lie within the interval

techniques we discussed previously: Monte Carlo, polynomial chaos, surrogates, etc. In addition to the simulated CDF, we also consider that we have a number of experimental measurements of the QoI. We are interested in knowing the degree to which the CDF for the QoI agrees with the measured data. We call the CDF of the simulation $F_{\text{sim}}(Q)$ and the CDF from the measurements (observations) $F_{\text{obs}}(Q)$.

This quantification can be made using the Minkowski L_1 metric:

$$d(F_{\text{sim}}, F_{\text{obs}}) = \int_{-\infty}^{\infty} |F_{\text{sim}}(Q) - F_{\text{obs}}(Q)|\, dQ. \tag{12.1}$$

The quantity d, sometimes called a validation metric, measures the amount of disagreement between the observed and calculated CDFs. If d were zero, there would be perfect agreement between the simulation and observation. Additionally, we can think of d as encompassing all the possible uncertainty between the simulation and observation. This includes unknown uncertainties such as model form uncertainty, numerical error, etc. However, the size of d is strongly influenced by the number of experimental observations.

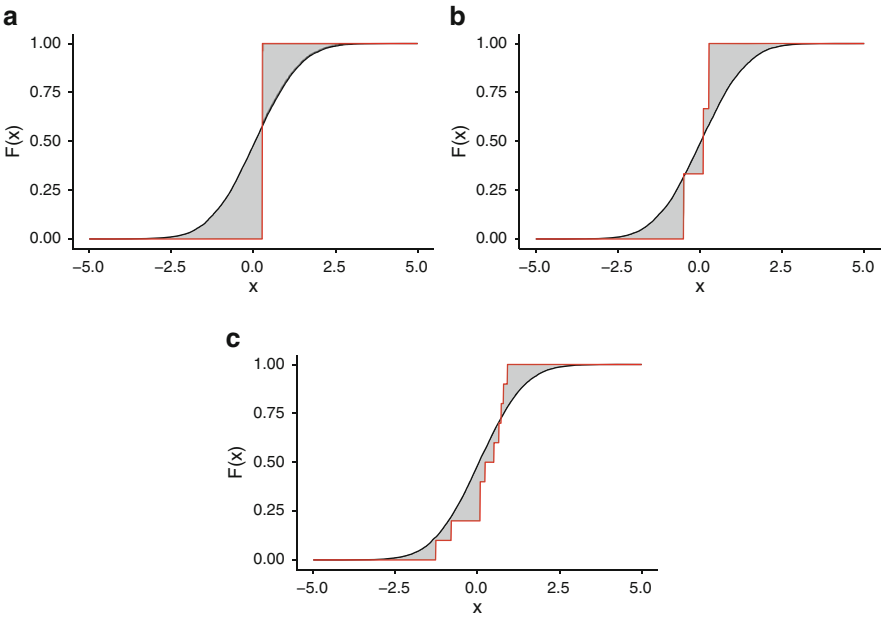

Fig. 12.2 Examples of comparison of observations and the CDF of a QoI estimated from simulations (black line). The number of observations affects the degree to which we can conclude the two are in agreement. The shaded area in each figure is the validation metric d. (**a**) One measurement. (**b**) Three measurements. (**c**) Ten measurements

The validation metric is illustrated in Fig. 12.2 where the shaded area represents the value of d. As we can see in the left panel, when there is a single measurement to compare with the computational results, the value of d is not small despite the fact that the measurement "agrees" with the prediction. If the single measurement were shifted to the left or right, it would be possible to increase d. When further measurements are added, they can reduce the magnitude of d. This is a feature of the validation metric: it naturally indicates the greater confidence in the simulation model when there are more measurements.

The realization that d makes it possible to estimate the impact of unknown uncertainties suggests how we might use it to make a prediction. Given that with the experimental data the empirical CDF had a given amount of area between it and the simulated CDF, when we make another prediction, we can assume that the computed CDF could fall within a range of possible CDFs that have an area between them and the nominal simulated CDF equal to d. That is, we have extrapolated d to the prediction. This is our first example of a probability box, a topic of discussion below.

We have not included the uncertainty in the experimental measurement of the QoI in the construction of the validation metric: our definition implicitly assumes that the observation is exact. If the uncertainty in the observation is small, then this is

a safe assumption. However, in practice the measurement uncertainty may be large. Therefore, there will be a distribution of the value of d. In this case the CDF for the observation will not be a piecewise constant function. Nevertheless, the calculation of d is the same.

12.2 Horsetail Plots and Second-Order Sampling

The previous section discussed using CDFs derived from simulations based on propagating aleatory uncertainty. Now we consider the case where there are epistemic and aleatory uncertainties in the QoI. In this case for each possible value of the epistemic uncertainties, there is a CDF at that value. This arises from the fact that the epistemically uncertain inputs have a value that is unknown. At that value there is a CDF of the QoI based on the aleatory uncertainties. Nevertheless, we do not know which CDF of all possible CDF functions that is.

To estimate the range of possible outcomes due to epistemic uncertainty, we can use a technique known as second-order sampling. We assume that it is possible to produce a CDF of the QoI when the epistemically uncertain inputs are fixed; in practice this is a strong assumption because this involves uncertainty propagation of all the aleatory uncertainties. We sample from the epistemic uncertainties *as though they were uniform distributions between their minimum and maximum values*. For each sample, we have a CDF of the QoI. If we plot these, we get what is known as a horsetail plot. The range of CDFs in the horsetail plot is an estimate of the bounds of the actual CDF. It is an estimate because we have a finite number of values of the epistemic uncertainties.

A horsetail plot for a normal distribution where the mean and standard deviation of the distribution are epistemic uncertainties is shown in Fig. 12.3. In the figure

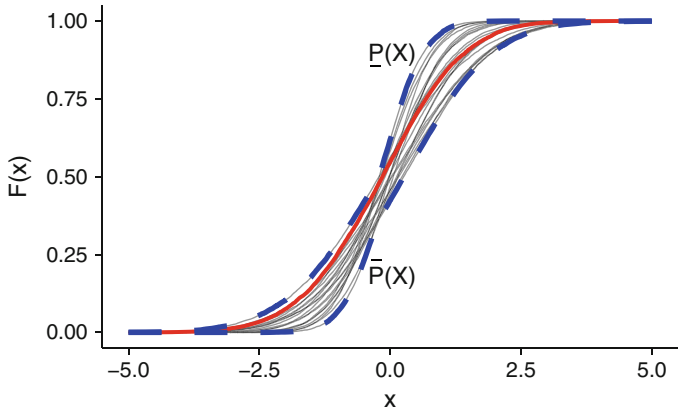

Fig. 12.3 Horsetail plot of 20 potential CDFs of a distribution $x \sim \mathcal{N}(\mu, \sigma)$ where $\mu \in [-0.25, 0.25]$ and $\sigma \in [0.55, 1.45]$. The dashed lines show the upper and lower bound of the sampled CDFs. The thick solid line is the actual distribution with $\mu = -0.168$ and $\sigma = 1.25$

there are 20 different potential CDFs obtained by sampling a value of $\mu \in [-0.25, 0.25]$ and $\sigma \in [0.55, 1.45]$ using uniform distributions and then plotting the resulting CDF. For this example we say that the "correct" CDF has $\mu = -0.168$ and $\sigma = 1.25$; in the figure, it can be seen that this CDF falls within the sampled CDFs. It is possible for the true CDF to fall outside these bounds if there are not enough CDFs sampled. Notice that the bounding values of the sampled CDFs are not each a single CDF from the samples, rather a combination different CDFs. These bounding functions will be discussed below when defining a bounding box.

It is important to keep in mind that though we are sampling from a uniform distribution, we are not interpreting the outputs as coming from a distribution. That is why for N samples we have N different CDFs and not a single CDF. Because we are using sampling, the process of second-order sampling can benefit from stratified sampling and other techniques to improve simple random sampling. Indeed, when there are many epistemic uncertainties, these techniques are essential because of the cost of creating each CDF.

12.3 P-Boxes and Model Evidence

We denote the upper bound of the CDFs in the horsetail plot as $\underline{P}(x)$ because the true CDF is below this function; the lower bound of the CDFs in the horsetail plot is $\overline{P}(x)$ because the true CDF is above this function. These functions are shown in Fig. 12.3. The functions $\overline{P}(x)$ and $\underline{P}(x)$ define an area between them called a probability box, or p-box for short. Now we are in the position where we have a range of possible CDFs that could represent the system. The question that arises is how can we quantify the agreement between the model and experiment in this case.

One possible solution is to generalize the validation metric to include the discrepancy between the experimental values and the p-box. To this end we can define the validation metric as

$$d(F_{\text{sim}}, F_{\text{obs}}) = \int_{-\infty}^{\infty} D(\overline{P}(Q), \underline{P}(Q), F_{\text{obs}}(Q))\, dQ, \tag{12.2}$$

where

$$D(\overline{P}(Q), \underline{P}(Q), F_{\text{obs}}) = \begin{cases} 0 & F_{\text{obs}}(Q) \in [\overline{P}(Q), \underline{P}(Q)], \\ \min\left(|F_{\text{obs}}(Q) - \overline{P}(Q)|, |F_{\text{obs}}(Q) - \underline{P}(Q)|\right) & F_{\text{obs}}(Q) \notin [\overline{P}(Q), \underline{P}(Q)] \end{cases}. \tag{12.3}$$

From this definition of d, we get a validation metric that gives a measure of the agreement of a measurement with the possible range of CDFs given epistemic

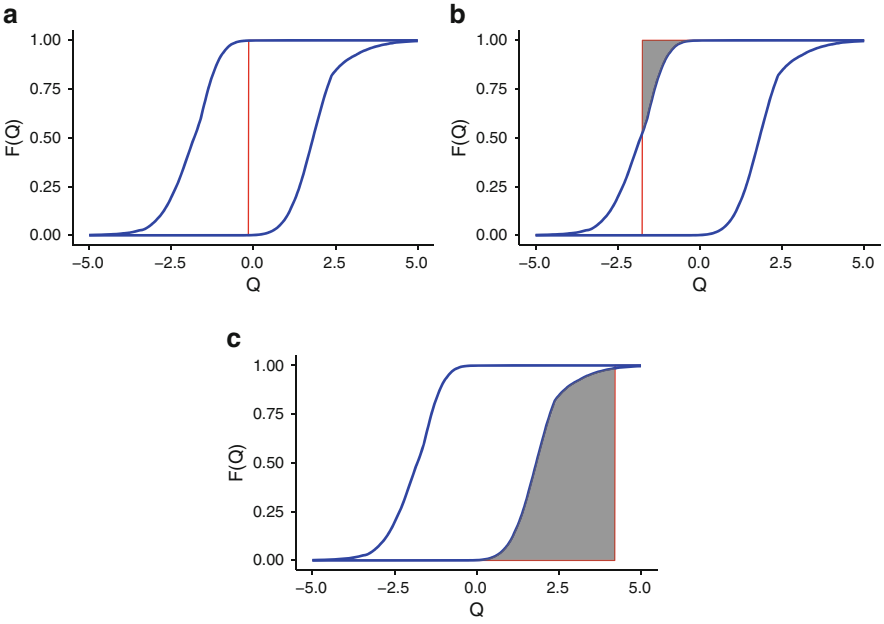

Fig. 12.4 Three examples of the validation metric d when the CDF of the simulation is a p-box and there is a single experimental measurement. The shaded area is equivalent to the value of d for each case. In each figure the p-box is the same, but the observed value differs. (**a**) $d \approx 0$. (**b**) $d \approx 0.2$. (**c**) $d \approx 2$

uncertainty. The value of d when there is a p-box is illustrated in Fig. 12.4. In the left panel, the value of d is approximately 0. This does not mean that the simulation is perfect. Rather, we can conclude that there is no evidence of disagreement between the simulation and measurement. That is, there are values of the epistemically uncertain parameters that would agree with the measurement. In the second and third panels of Fig. 12.4, that value of d is larger and represents the area between the nearest edge of the p-box and the measurement. As before, the validation metric can improve as more experimental measurements are added that "agree" with the data, that is, if d is not already approximately zero.

When p-boxes are involved, the value of d does not contain any information about the precision of the computational model when epistemic uncertainty is concluded. As shown in Fig. 12.5, a large p-box, which indicates a large degree of epistemic uncertainty, could easily have a small value of d because of the large range of potential results to bound the measurement. Nevertheless, a small p-box indicative of a model that has a small amount of epistemic uncertainty could have the same value of d. The value of d cannot pick out the method/process that has a smaller degree of epistemic uncertainty; that quantification requires different analysis perhaps by computing the area of the p-box.

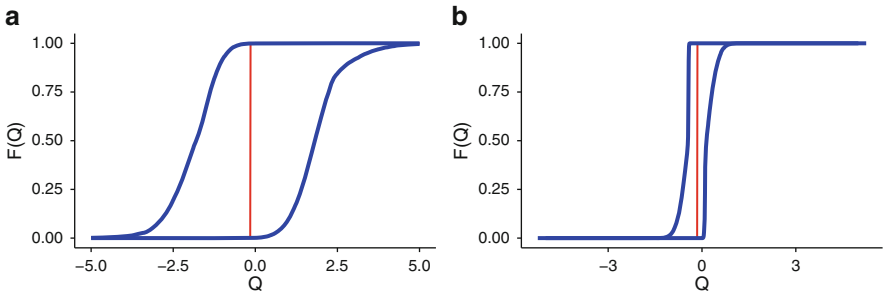

Fig. 12.5 Demonstration that $d = 0$ does not mean that the amount of uncertainty in the prediction is small: in the left plot, the simulation results can agree with the measurement, but epistemic uncertainty is quite large. (**a**) $d \approx 0$ with low precision. (**b**) $d \approx 0$ with higher precision

12.4 Predictions Under Epistemic Uncertainty

The validation metric d measures the agreement between the simulation and observations. To apply this metric to a prediction, that is, to a new scenario where we do not have observations, we want to quantify how much we should adjust the CDF or p-box obtained from a UQ analysis. To aid in this, we consider that the definition of d yields a quantity that is equal to the average difference between the inverse of the simulation CDF and the inverse of the observation CDF (or p-boxes). Therefore, we could add d to both sides of the CDF or p-box obtained from simulations as an estimate for the possible range of outputs of the prediction. This procedure assumes that we can extrapolate the result and that d will not be larger in the prediction. Therefore, given a p-box for the prediction, the adjusted p-box for the prediction would be defined by

$$\underline{P}_{\text{pred}}(Q) \equiv \underline{P}(Q + d), \qquad \overline{P}_{\text{pred}}(Q) \equiv \overline{P}(Q - d). \tag{12.4}$$

In the case of the CDF, we would apply the shift to the CDF to create a p-box. If the CDF is given by $F(Q)$, the resulting p-box is

$$\underline{P}_{\text{pred}}(Q) \equiv \underline{F}(Q - d), \qquad \overline{P}_{\text{pred}}(Q) \equiv \overline{F}(Q + d). \tag{12.5}$$

Other types of extrapolation are also possible. For instance, one could compute the portion of d that is to the left or right of the CDF/p-box and then create a shift that is not symmetrically applied. For a p-box these shifts would be given by

$$d_{\text{left}}(F_{\text{sim}}, F_{\text{obs}}) = d(\underline{P}(Q), F_{\text{obs}}(Q)), \tag{12.6}$$

and

$$d_{\text{right}}(F_{\text{sim}}, F_{\text{obs}}) = d(\overline{P}(Q), F_{\text{obs}}(Q)). \tag{12.7}$$

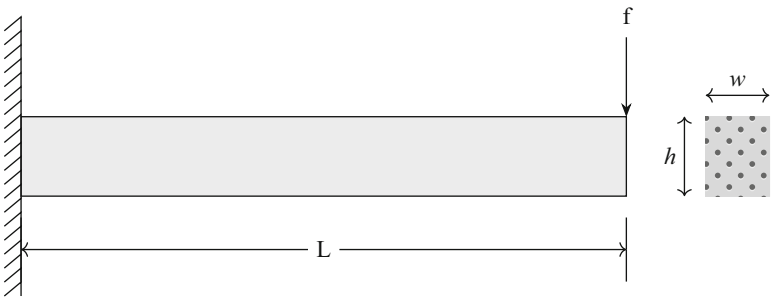

Fig. 12.6 Cantilevered beam with a load of f placed at the edge. The shape of the beam is an aleatory uncertainty, and the elastic modulus of the beam is an epistemic uncertainty

With these results we can then define $\underline{P}_{\text{pred}}(Q)$ and $\overline{P}_{\text{pred}}(Q)$ using d_{left} and d_{right}, respectively. There are obvious benefits in calculating the portion of d to the left/right of the p-box. The downside is that the structure of the discrepancy may not hold up upon extrapolation: a model that consistently predicts values too low in one scenario may not give low predictions in other case.

As an example of adding a p-box to a prediction, we consider the deflection of an end-loaded cantilevered beam, as shown in Fig. 12.6. The deflection of the beam, y, is given by the formula

$$y = \frac{4\text{f}L^3}{Ewh^3}, \tag{12.8}$$

where f is a force in Newtons, E is the elastic modulus, and the dimensions L, w, and h are shown in the figure. Due to manufacturing tolerances, the dimensions of the beam are normally distributed with

$$\mu_L = 1\,\text{m}, \sigma_L = 0.05\,\text{m},$$

$$\mu_w = 0.01\,\text{m}, \sigma_w = 0.0005\,\text{m},$$

and

$$\mu_h = 0.02\,\text{m}, \sigma_h = 0.0005\,\text{m}.$$

For this particular material, the elastic modulus is only known to be in an interval: $E \in [69, 100]\,\text{GPa}$.

We begin by performing ten measurements of the deflection at $\text{f} = 75\,\text{N}$ and compare the resulting CDF to the p-box computed from the model given by Eq. (12.8) in Fig. 12.7. Using the measurements at 75 N, we compute a value of $d \approx 0.00572\,\text{m}$ and $d_{\text{left}} \approx 0.556\,\text{m}$ and $d_{\text{right}} \approx 0.0001$. Using these values of d, we then predict the deflection of the beam at 100 N by computing the p-box from

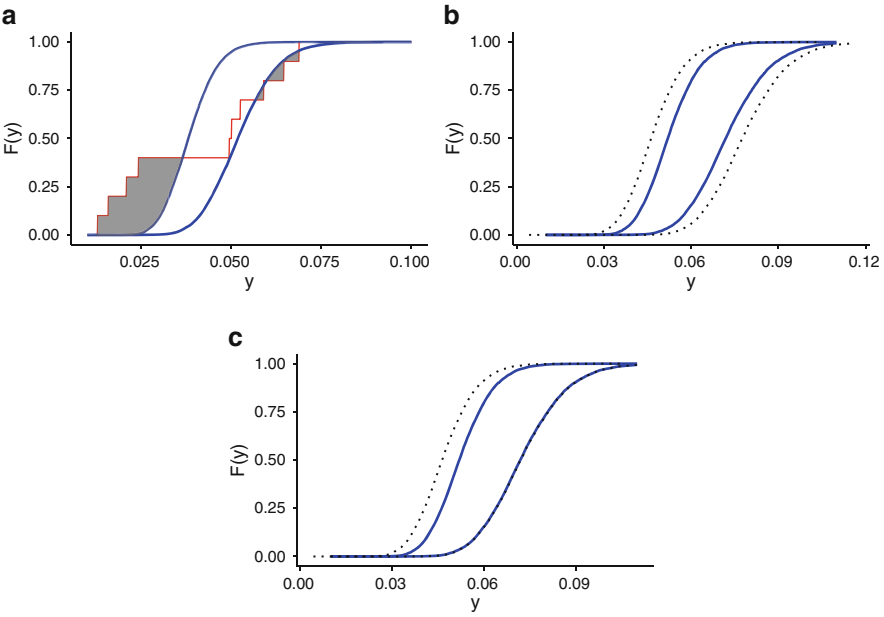

Fig. 12.7 Example of using d to adjust p-box for prediction for the cantilevered beam example. The model was tested at f = 75 N, and the resulting d is extrapolated to predict performance at f = 100 N by adding d to the p-box and adding d_{left} and d_{right} to the p-box (the dotted lines). (**a**) f = 75 N. (**b**) f = 100 N adding d. (**c**) f = 100 N adding d_{left}, d_{right}

using second-order sampling. The p-box is then adjusted by adding d symmetrically (see central panel of the triptych in Fig. 12.7) or by adding d_{left} and d_{right} in the right panel. In this problem it is clear that the model tends to give too high a value of the deflection (based on the experimental data). The result is that when asymmetrically adjusting the p-box there is almost no adjustment to $\overline{P}(y)$.

12.5 Beyond Interval Uncertainties with Expert Judgment

To this point we have only dealt with epistemic uncertainties that have a simple interval. There are cases where we have more information than just an interval, but not enough information to have a true distribution. The ideas we use are based on Dempster-Shafer evidence theory in a simplified form. The approach detailed discusses how to include information from experts that disagree and only have an indication of what values of the quantity are more likely.

Consider a parameter $\theta \in [a, b]$ that represents an epistemic uncertainty in a model. An expert is solicited to give a basic probability assignment (BPA) to different ranges of the interval. The BPAs each must be in the range [0, 1] and must

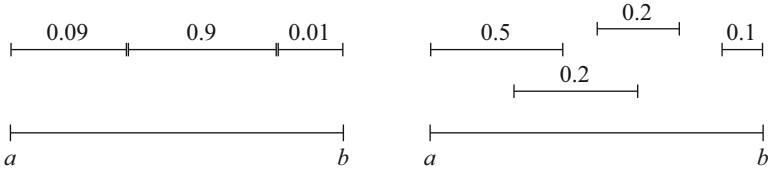

Fig. 12.8 Two illustrative BPA structures for $\theta \in [a, b]$: the left example is a simple interval considered to be more probable in the middle with a left skew, and the right example has overlapping regions and gaps. Note that in both cases the BPAs sum to one

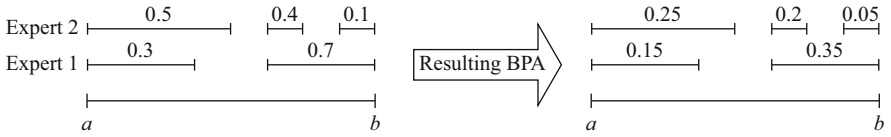

Fig. 12.9 Example of combining BPA structures from two experts: the BPAs are divided by the number of experts to get a single BPA structure

sum to 1. For example, if $\theta \in [0, 1]$ and the expert believes that values in the middle of the interval are more likely and the value of θ has a small likelihood of being close to 1, the BPA might be

$$\text{BPA}(\theta) = \begin{cases} 0.09 & \theta \in [0, 0.35) \\ 0.9 & \theta \in [0.35, 0.8] \\ 0.01 & \theta \in (0.8, 1) \end{cases}.$$

This BPA structure is shown in the left part of Fig. 12.8. The BPA may have overlaps or gaps depending on the situation; such a BPA structure is also shown in Fig. 12.8.

The BPA is then used in second-order sampling by selecting a BPA interval based on its value (higher BPAs being more likely to be selected) and then picking a uniform random number in that interval. This procedure will construct a p-box as before, but the construction is informed by the BPAs assigned by the expert. The BPAs only inform the sampling of θ and do not give any weighting to the resulting CDFs in the construction of the p-box. Therefore, in the limit of an infinite number of samples of θ in second-order sampling, the result will be the same as treating θ as a simple interval, but in practice when samples are limited, the resulting p-box will be informed by the expert opinion.

If there are multiple experts that assign BPAs to θ, their knowledge can be fused by taking the BPAs from each expert and dividing by the number of experts. This will result in a set of BPAs that sum to 1 and can then be used in second-order sampling as before. The resulting BPA structure will have high values where the experts agree while still including information where disagreement exists. An example of this combination is shown in Fig. 12.9. Here two experts have different

BPA structures that are combined by taking the union of the BPA structures and dividing by 2 (the number of experts).

The resulting BPA after combining experts would be used in second-order sampling as though the BPA came from a single expert. The properties of the resulting p-box would be the same as for a single expert.

12.6 Kolmogorov-Smirnoff Confidence Bounds

In the preceding sections, we discussed using second-order sampling to estimate p-boxes when epistemic uncertainties were present. It is reasonable to ask what are the uncertainties in the p-box due to the finite number of samples used to generate the CDFs. An approach to add a confidence interval uses the Kolmogorov distribution and is based on the Kolmogorov-Smirnov (KS) test. This approach is distribution free, that is, it will work for any continuous distribution and is therefore appropriate for the distribution of a QoI where we typically have minimal knowledge of the form of the distribution. The resulting confidence intervals are, nevertheless, fairly large because this technique does not use any of the data we have to inform the confidence interval, other than the number of samples. The confidence intervals would also be extended if there was a validation metric d being extrapolated into the scenario.

The KS test statistic δ_N is the maximum vertical distance between the true (but unknown) CDF, $F(x)$, and the empirical CDF derived from N samples, $F_N(x)$:

$$\delta_N = \sup_x |F_N(x) - F(x)| . \tag{12.9}$$

This distance is illustrated in Fig. 12.10.

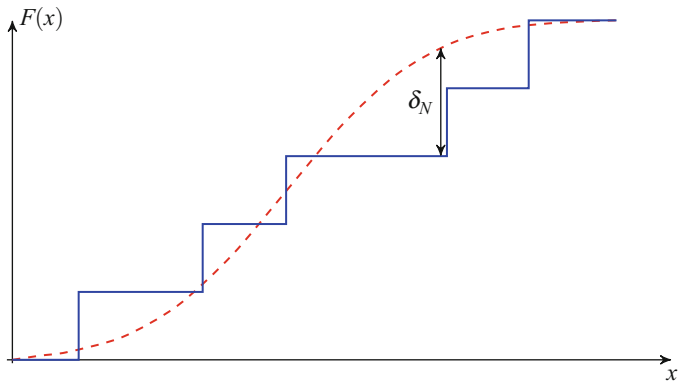

Fig. 12.10 Illustration of δ_N: the maximum vertical distance between the true CDF $F(x)$, dashed line, and empirical CDF, $F_N(x)$, solid line, with N samples

We know that as $x \to \pm\infty$ that the true and empirical CDF will go to 0 and 1. This means that the value of δ_N will be at some finite value of x. It also means that the difference between the empirical CDF and the true CDF has the character of a random walk with fixed endpoints: the difference is 0 at the endpoints ($\pm\infty$) and a random value between 0 and 1 in between the endpoints. It can be shown using the theory of random walks that if F_N converges to F as $N \to \infty$, then $\sqrt{N}\delta_N$ converges to a Kolmogorov distribution as $N \to \infty$. The Kolmogorov distribution has a CDF given by

$$F_K(x) \approx \frac{\sqrt{2\pi}}{x} \sum_{i=1}^{\infty} \exp\left(\frac{-(2i-1)^2\pi^2}{8x^2}\right). \tag{12.10}$$

Therefore, if $\sqrt{N}\delta_N$ is described by a Kolmogorov distribution, we want to know when $F_K(\sqrt{N}\delta_N) = 1 - \alpha$ for some confidence α in order to estimate. In other words, if we wanted to know what δ_N was with 95% confidence ($\alpha = 0.05$), we would solve $F_K(\sqrt{N}\delta_N) = 0.95$. This task is made difficult when we account for the fact that at small values of N, δ_N is not well approximated by Eq. (12.10). Formulas exist for the distribution as a function of N (Marsaglia et al. 2003) and can be used for determining various confidence levels for δ_N given a number of samples. A useful approximate formula for the 95% confidence value of δ_N is

$$\delta_N^{95\%} \approx \begin{cases} \dfrac{1.1897N^{3/2} + 0.00863443N^2 + 1.04231N}{-3.893\sqrt{N} + 4.32736} \dfrac{1}{N^2} & 5 \le N \le 50 \\ \dfrac{1.3581}{\sqrt{N}} & N > 50 \end{cases}. \tag{12.11}$$

Using this formula for δ_N, we can extend a p-box to account for the finite number of samples in the construction of the CDFs by defining $\hat{\underline{P}}$ and $\hat{\overline{P}}$ as

$$\hat{\underline{P}}(Q) = \min\left(\underline{P}(Q) + \delta_N, 1\right) \tag{12.12a}$$

$$\hat{\overline{P}}(Q) = \max\left(\overline{P}(Q) - \delta_N, 0\right). \tag{12.12b}$$

The extended p-box is constrained by the fact that the range of the CDF is [0, 1]. Other adjustments may also be possible due to physical restrictions (e.g., the value of Q is always positive or in some range).

To demonstrate the extension of a p-box using δ_N, we return to the cantilevered beam example. At $f = 75$ N, we construct a p-box using second-order sampling. We use 20 sampled values of the elastic modulus, E, and at each value, we use only 10 samples of the aleatory uncertainties. Using Eq. (12.11) we compute $\delta_N^{95\%}(10) = 0.4092477$. The resulting distribution and its adjusted p-box, $[\hat{\underline{P}}, \hat{\overline{P}}]$, are shown in Fig. 12.11. Also, in the figure, the p-box with 10^4 samples used in each CDF is shown. This example demonstrates that the 95% KS confidence interval does bound

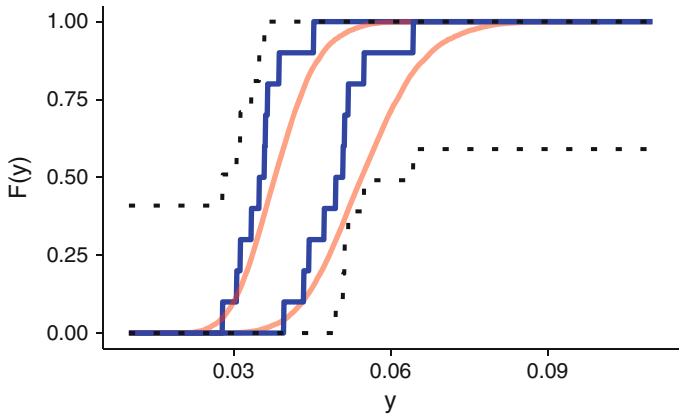

Fig. 12.11 A p-box for the cantilevered beam problem at $f = 75$ N where there are $N = 10$ samples used to construct the CDFs in second-order sampling (solid, piecewise constant line). The Kolmogorov-Smirnov 95% confidence interval for the p-box is shown with a dotted line. The smooth p-box is constructed with 10^4 samples in each CDF

the true p-box, though near the extremes of the original p-box the confidence interval is very conservative.

Overall, the KS confidence interval for the p-box is useful when the model is expensive to evaluate and only rough CDFs can be constructed using second-order sampling. The results from the method may be large p-boxes, but they can be used in a wide variety of applications to understand if performance limits or regulations are potentially at risk in a given scenario.

12.7 The Method of Cauchy Deviates

When there are many dimensions of epistemic uncertainty, the construction of horsetail plots and p-boxes will require many evaluations to adequately explore the space of epistemic uncertainty. To help with this, we use a Cauchy distribution, which has nonfinite mean and variance, to estimate the range of the response due to epistemic uncertainty (Kreinovich and Ferson 2004; Kreinovich et al. 2004; Kreinovich and Nguyen 2009). This is another apparition of the Witch of Agnesi,[1]

[1]The Witch of Agnesi is the name of the curve $y = 1/(1 + x^2)$ studied by Maria Agnesi in the oldest extant mathematics text written by a woman (Agnesi 1748). This curve was called a *averisera* the Latin for "versed sine curve." This being one letter off of the Italian word *avversiera*, a term synonymous with "witch," it was mistranslated into English. The resemblance between the representation of the millinery of witches in popular depictions and the graph of the function is seemingly coincidental.

which appeared previously in the problems of Chap. 9, as the PDF of a Cauchy distribution is equivalent to this function. In this case the witch is here to help.

The Cauchy distribution is described by a single parameter $\Delta > 0$, and its PDF is

$$f(x) = \frac{\Delta}{\pi(x^2 + \Delta^2)}; \tag{12.13}$$

the CDF is

$$F(x) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{x}{\Delta}\right). \tag{12.14}$$

The mean and variance of a Cauchy distribution are undefined. The mode of the distribution is at $x = 0$. Because the mean and variance are undefined, the central limit theorem does not apply to a sum of Cauchy random variables: the sum of N Cauchy random variables will not limit to a normal distribution as $N \to \infty$.

Given a set of N samples, x_n, from a Cauchy distribution, the maximum likelihood estimate of Δ is given by the relation

$$\sum_{n=1}^{N} \frac{1}{1 + \frac{x_n^2}{\Delta^2}} = \frac{N}{2}. \tag{12.15}$$

Note that the function on the LHS is monotonically increasing and that the solution is in the interval $\Delta \in [0, \max_n x_n]$, so that closed root finding methods can be used.

Also, a linear combination of p-independent Cauchy random variables,

$$\hat{x} = c_1 x_1 + c_2 x_2 + \cdots + c_p x_p,$$

where the c_i are real numbers and each Cauchy random variable x_i has a parameter Δ_i, is also a Cauchy random variable with parameter

$$\Delta = |c_1|\Delta_1 + |c_2|\Delta_2 + \cdots + |c_p|\Delta_p.$$

It is this property of Cauchy random variables that we will use to quantify epistemic uncertainty. We consider a QoI, $Q(x, \theta)$, where x is a vector of aleatory uncertainties and θ is a vector of p epistemic uncertainties that are normalized so that $\theta_i \in [\underline{\theta}_i, \overline{\theta}_i]$. We also define $\hat{\theta}_i = 0.5(\underline{\theta}_i + \overline{\theta}_i)$ and $\Delta\theta_i = 0.5(\overline{\theta}_i - \underline{\theta}_i)$. We then make an approximation that Q can be written as a linear function of the epistemic uncertainties at each value of x:

$$Q(x, \hat{\theta}_1 + \delta\theta_1, \ldots, \hat{\theta}_p + \delta\theta_p) = Q(x, \hat{\theta}_1, \ldots, \hat{\theta}_p) + \delta Q, \tag{12.16}$$

where $\delta\theta_i \in [-\Delta\theta_i, \Delta\theta_i]$, and

$$\delta Q = c_1 \delta\theta_1 + c_2 \delta\theta_2 + \cdots + c_p \delta\theta_p, \tag{12.17}$$

Algorithm 12.1 Algorithm for second-order sampling procedure to determine the distribution of $\delta Q(x, \theta)$ as described in Eq. (12.16)

for m = 1 to M **do**
  Sample value x_m as appropriate for aleatory uncertainties.
  Compute $Q_{m,\text{mid}} = Q(x_m, \hat{\theta}_1, \ldots, \hat{\theta}_p)$.
  for n = 1 to N **do**
    Sample Cauchy RV for each θ_i: $d_{in} = \tan\left(\pi\left(\xi_i + \frac{1}{2}\right)\right)$, where $\xi \sim \mathscr{U}(0, 1)$.
    Compute $K_n = \max_i |d_{in}|$.
    Set $\delta\theta_{in} = d_{in}/K_n \Delta\theta_i$.
    Compute $\delta Q_{nm} = K_n(Q(x_m, \hat{\theta}_1 + \delta\theta_{1n}, \ldots, \hat{\theta}_p + \delta\theta_{pn}) - Q_{m,\text{mid}})$.
  end for
  Compute Δ_m by solving

$$\sum_{n=1}^{N} \frac{1}{1 + \frac{\delta Q_{nm}^2}{\Delta_m^2}} = \frac{N}{2}.$$

end for

with $c_i = \frac{\partial Q}{\partial \theta_i}$. Using the properties discussed above, if the $\delta\theta_i$ are Cauchy distributed, then δQ will be Cauchy distributed with a computable Δ parameter. Notice that δQ will be a function of x; this makes the linear approximation less restrictive.

In Algorithm 12.1 a procedure for determining the parameter of a Cauchy-distributed δQ at M different samples of the aleatory uncertainties x. The algorithm will require $M(N + 1)$ evaluations of the QoI. Also, each iteration will require the solution of a single nonlinear equation to determine Δ_m, the parameter of the distribution at x_m. This equation can be solved with a simple closed method, such as bisection, and does not require any further evaluations of the QoI. In the algorithm the values of $\delta\theta_i$ are scaled so that their values are always between $\pm\Delta\theta_i$. By the way that we defined δQ and normalized the sampling, the value of $Q(x_m, \theta) \in [Q_{m,\text{mid}} - \Delta_m, Q_{m,\text{mid}} + \Delta_m]$. Therefore, the algorithm gives an estimate of the bounds of Q at a particular value of x.

The value of Δ_m that we obtain from Algorithm 12.1 is an approximation due to the finite number of samples. As discussed by Kreinovich and Ferson (2004), using a finite number of samples to calculate an approximate Δ will result in an overestimation of the true range of the QoI by a factor $(1 + 2\sqrt{2/N})$, where N is the number of samples. Therefore, using 50 samples will yield an up to 40% overestimation of the true interval.

One consideration in using the Cauchy deviates method is the number of parameters, p. In principle, the range of the interval could be estimated using numerical differentiation to compute the c_i and then use the linear approximation to compute the range of Q. The number of QoI evaluations to estimate the derivatives is $p + 1$, whereas the Cauchy deviates method does not depend on p. Therefore, when p is large and we can afford to overestimate the interval size, the Cauchy deviates method is appropriate.

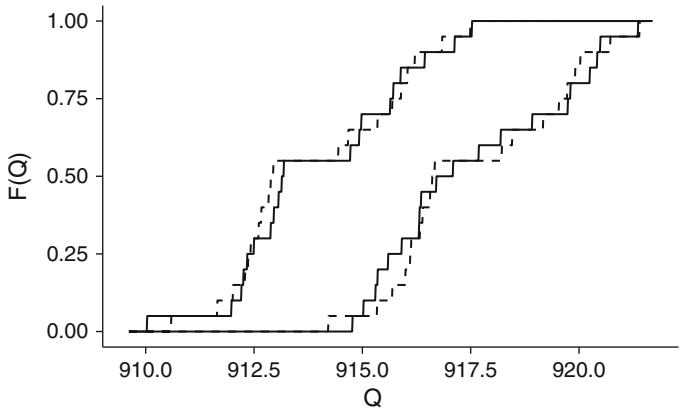

Fig. 12.12 A p-box for the oscillator problem with 1200 epistemic uncertainties. The solid lines are the p-box derived from the method of Cauchy deviates, and the dashed lines are the result of second-order sampling using 10^5 samples of the epistemic uncertainties at each value of x

To demonstrate the Cauchy deviates method, we adapt a problem from Kreinovich and Ferson (2004). We consider a multiple oscillator problem where the QoI is given by

$$Q(x, k_i, m_i, c_i) = \sum_{i=1}^{400} \frac{k_i}{\sqrt{(k_i - m_i x^2)^2 + c_i x^2}}. \tag{12.18}$$

The epistemic uncertainties are

- $k_i \in [\underline{k}_i, \overline{k}_i]$ where the intervals are the range [60, 230] divided into 400 equal pieces, i.e., $k_1 \in [60, 60.425]$, $k_2 \in [60.425, 60.85]$, ...,
- $m_i \in [\underline{m}_i, \overline{m}_i]$ where the intervals are the range [10, 12] divided into 400 equal pieces, i.e., $m_1 \in [10, 10.005]$, $m_2 \in [10.005, 10.01]$, ...,
- $c_i \in [\underline{c}_i, \overline{c}_i]$ where the intervals are the range [5, 25] divided into 400 equal pieces, i.e., $c_1 \in [5, 5.05]$, $c_2 \in [5.05, 5.1]$, ...

The aleatory uncertain variable is $x \sim \mathcal{N}(\mu = 2.75, \sigma = 0.01)$. This large number of epistemically uncertain parameters (1200) would require many evaluations of the QoI. Using the Cauchy deviates method, we can estimate a p-box for Q using only 1020 function evaluations by setting $M = 20$ and $N = 50$. Under these conditions, we obtain the p-box shown in Fig. 12.12. At each of the 20 samples of x, we evaluate the QoI 50 times to get values of δQ_m. We then use the values $\bar{Q}_m \pm \Delta_m$ to construct the upper and lower bounds of the p-box by computing the empirical CDF for the bounding cases.

The resulting p-box from the Cauchy deviates method is compared with second-order sampling using 10^5 samples at each value of x. The Cauchy deviates method

does give a slightly smaller p-box than that derived from second-order sampling. Nevertheless, the Cauchy deviates result required about 100 times fewer evaluations of the QoI.

12.8 Notes and References

A detailed discussion of p-boxes, validation metrics, and second-order sampling can be found in Oberkampf and Roy (2010). That work also provides examples from real engineering systems, how to combine QoIs into a single metric, and other nuances. Other works, often associated with Sandia National Laboratories, give more description of Dempster-Shafer structures (Ferson et al. 2003) and interval uncertainties (Ferson et al. 2007).

The exploration of epistemic uncertainties can be extended to include the impact of perturbations to the input probabilities and prior distributions on the results of a UQ analysis. The study of these perturbations is considered in Chowdhary and Dupuis (2013) and Owhadi et al. (2013, 2015).

12.9 Exercises

1. Construct a p-box by sampling from a normal distribution $x \sim \mathcal{N}(\mu = a, \sigma = b)$ using second-order sampling where each CDF is constructed with 100 points and the epistemic uncertainties have 10 and 100 samples, where

   - $a \in [-0.2, 0.2]$ and $b \in [0.9, 1.1]$.
   - a is given by the BPA structure

$$\text{BPA}(a) = \begin{cases} 0.1 & a \in [-0.2, -0.05) \\ 0.8 & a \in [-0.05, 0.05] \\ 0.1 & a \in (0.05, 0.2) \end{cases},$$

   and b is given by the BPA structure

$$\text{BPA}(b) = \begin{cases} 0.4 & b \in [0.9, 0.95) \\ 0.6 & b \in [1, 1.1] \end{cases}.$$

2. Using a discretization of your choice, solve the equation

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = D\frac{\partial^2 u}{\partial x^2} - \omega u,$$

for $u(x,t)$ on the spatial domain $x \in [0, 10]$ with periodic boundary conditions $u(0^-) = u(10^+)$ and initial conditions

$$u(x,0) = \begin{cases} 1 & x \in [0, 2.5] \\ 0 & \text{otherwise} \end{cases}.$$

Use the solution to compute the total reactions

$$\int_5^6 dx \int_0^5 dt\, \omega u(x,t).$$

Sample values of parameters using a uniform distribution centered at the mean with upper and lower bounds $\pm 10\%$ for the following variables:

(a) $\mu_v = 0.5$,
(b) $\mu_D = 0.125$,
(c) $\mu_\omega = 0.1$,

and treat the mesh spacing in space and time as an epistemic uncertainty:

(a) $\Delta x \in [0.001, 0.5]$,
(b) $\Delta t \in [0.001, 0.5]$.

Derive p-boxes for Q using second-order sampling.

(c) Repeat the previous problem using Kolmogorov-Smirnov confidence bounds.
(d) For the PDE and QoI in problem 2, consider ω as an independent epistemic uncertainty in each mesh zone i with range $\omega_i \in [0.05, 0.15]$. Assume v and D are uniform in the domain and distributed in the same form as in problem 2. Compute p-boxes using the Cauchy deviates method, and compare with second-order sampling.

Appendix A
Cookbook of Distributions

We've a first-class assortment of magic;
And for raising a posthumous shade
With effects that are comic or tragic,
Theres no cheaper house in the trade.

—from the opera *The Sorcerer* by W.S. Gilbert and Arthur Sullivan

This appendix gives the definitions and properties of a variety of typical distributions for random variables. Most of these distributions are discussed elsewhere in the text, but having the definitions in a central location can be useful for reference. The notation used here is the same as can be found in other standard references, except where indicated otherwise.

A.1 Bernoulli Distribution

This is a discrete distribution where the random variable takes the value of 1 with probability p and the value of 0 with probability $1 - p$. If we consider the toss of a fair coin, and we assign the outcome of heads the value of $x = 1$ and tails $x = 0$, then x is Bernoulli distributed with $p = 0.5$. For simplicity we also define $q = 1 - p$.

A.1.1 Probability Mass Function (PMF)

$$f(x|p) = \begin{cases} 1-p & x = 0 \\ p & x = 1 \end{cases}$$

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0

A.1.2 Cumulative Distribution Function (CDF)

$$F(x|p) = \begin{cases} 0 & x < 0 \\ 1-p & 0 \leq x < 1 \\ 1 & x \geq 1 \end{cases}$$

A.1.3 Properties

- Mean: $E[x] = p$
- Median:

$$\text{Median} = \begin{cases} 0 & q > p \\ 0.5 & q = p \\ 1 & q < p. \end{cases}$$

- Mode:

$$\text{Mode} = \begin{cases} 0 & q > p \\ \{0, 1\} & q = p \\ 1 & q < p. \end{cases}$$

- Variance: $pq = p(1-p)$
- Skewness:

$$\gamma_1 = \frac{1-2p}{\sqrt{pq}}$$

- Excess kurtosis:

$$\text{Kurt} = \frac{1-6pq}{pq}$$

A.2 Binomial Distribution

The binomial distribution is a discrete distribution that gives the number of binary events that are successes (i.e., the outcome is 1), out of $n \in \mathbb{N}$ trials when each trial has probability p of success. As an example, if I flip a fair coin ($p = 0.5$) ten times ($n = 10$), then the number of heads, x, in those ten tosses will be binomially

distributed. The Bernoulli distribution is a special case of the binomial distribution with $n = 1$.

A.2.1 PMF

$$f(x|n, p) = \binom{n}{x} p^x (1-p)^{n-x},$$

where the binomial coefficient is given by

$$\binom{n}{x} = \frac{n!}{x!(n-x)!}.$$

A.2.2 CDF

$$F(x|n, p) = I_{1-p}(n-x, 1+x) = (n-x)\binom{n}{x} \int_0^{1-p} t^{n-x-1}(1-t)^x \, dt,$$

where I_{1-p} is the regularized incomplete beta function.

A.2.3 Properties

- Mean: $E[x] = np$
- The median for a binomial distribution does not have a simple formula but it lies between the integer part of np and the value of np rounded up to the nearest integer, i.e., the median is in between $\lfloor np \rfloor$ and $\lceil np \rceil$.
- Mode:

$$\text{mode} = \begin{cases} \lfloor (n+1)\, p \rfloor & (n+1)p \text{ is 0 or a noninteger} \\ (n+1)\, p \text{ and } (n+1)\, p - 1 & (n+1)p \in \{1, \ldots, n\}, \\ n & (n+1)p = n+1 \end{cases}$$

- Variance: $np(1-p)$
- Skewness:

$$\gamma_1 = \frac{1-2p}{\sqrt{np(1-p)}}$$

- Excess kurtosis:

$$\text{Kurt} = \frac{1 - 6p(1-p)}{np(1-p)}$$

A.3 Poisson Distribution

The Poisson distribution is a discrete distribution on the non-negative integers that has a single parameter $\lambda > 0$ that gives the probability of an event occurring x times, if the events occur independently and at a known average rate.

A.3.1 PMF

$$f(x|\lambda) = \frac{\lambda^x e^{-x}}{x!}$$

A.3.2 CDF

$$F(x|\lambda) = e^{-\lambda} \sum_{i=0}^{\lfloor x \rfloor} \frac{\lambda^i}{i!}$$

A.3.3 Properties

- Mean: $E[x] = \lambda$
- The median is greater than or equal to $\lambda - \log 2$ and less than $\lambda + \frac{1}{3}$.
- There are two modes: $\lfloor \lambda \rfloor$ and $\lceil \lambda \rceil - 1$.
- Variance: λ
- Skewness:

$$\gamma_1 = \frac{1}{\sqrt{\lambda}}$$

- Excess kurtosis:

$$\text{Kurt} = \lambda^{-1}$$

A.4 Normal Distribution, Gaussian Distribution

The normal, or Gaussian, distribution is the most well known continuous distribution. It has two parameters, $\mu \in \mathbb{R}$ and $\sigma^2 > 0$, that correspond to the mean and variance for the distribution. We write a random variable x that is normally distributed with parameters μ and σ^2 as $x \sim \mathcal{N}(\mu, \sigma^2)$.

A.4.1 Probability Density Function (PDF)

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

A.4.2 CDF

$$F(x|\mu, \sigma^2) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]$$

where the error function $\operatorname{erf}(x)$ is defined as

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt.$$

A.4.3 Properties

- The mean, median, and mode is μ
- The variance is σ^2
- The skewness and excess kurtosis are 0.

The *standard normal* distribution has $\mu = 0$ and $\sigma = 1$. Any normal distribution can be transformed into a standard normal by centering and scaling. If $x \sim \mathcal{N}(\mu, \sigma^2)$ then $z \sim \mathcal{N}(0, 1)$ with

$$z = \frac{x-\mu}{\sigma}.$$

A.5 Multivariate Normal Distribution

The multivariate normal distribution is multidimensional generalization of the normal distribution. Here $\mathbf{x}$ is a k-dimensional vector, $\mathbf{x} = (x_1, x_2, \ldots, x_k)^{\mathrm{T}}$, $\boldsymbol{\mu}$ is a vector of the expected value, or mean of each of the random variables X_i:

$$\boldsymbol{\mu} = (E[x_1], E[x_2], \ldots, E[x_k])^{\mathrm{T}} = (\mu_1, \mu_2, \ldots, \mu_k)^{\mathrm{T}},$$

the covariance matrix Σ is a symmetric positive definite matrix with the determinant of the matrix written as $|\Sigma|$. A vector that is distributed as a multivariate normal with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ is written as $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$.

A.5.1 Probability Density Function (PDF)

$$f(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right).$$

A.5.2 CDF

There is no closed form expression for the CDF.

A.5.3 Properties

- The mean and mode is $\boldsymbol{\mu}$.
- The variance is the diagonal of Σ.

A.6 Student's t-Distribution, t-Distribution

The t-distribution (also known as Student's t-distribution[1]). This distribution resembles a standard normal distribution but it has an additional, positive, real parameter $\nu > 0$. In the limit $\nu \to \infty$ the distribution limits to a standard normal. The parameter ν is often called the number of degrees of freedom. With $\nu = 1$, the

[1]This name arises because the distribution was popularized by William Sealy Gosset under the pseudonym "Student" (Student 1908) to hide, for competitive reasons, the fact that it was used on samples from the beer making process at the Guinness brewery in Dublin, Ireland. *Brilliant!*

distribution is equivalent to the Cauchy distribution (see below). The smaller the value of ν the thicker the tails in the distribution.

Other than the thick tales, the distribution is also used to model the possible errors of having a small number of samples from a normal distribution. Given n samples from a normal distribution, the difference between the sample mean and the true mean of the distribution is a t-distribution with $\nu = n - 1$.

A.6.1 Probability Density Function (PDF)

$$f(x|\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}},$$

where $\Gamma(x)$ is the gamma function.

A.6.2 CDF

$$F(x|\nu) = \frac{1}{2} + x\Gamma\left(\frac{\nu+1}{2}\right) \times \frac{{}_2F_1\left(\frac{1}{2}, \frac{\nu+1}{2}; \frac{3}{2}; -\frac{x^2}{\nu}\right)}{\sqrt{\pi\nu}\,\Gamma\left(\frac{\nu}{2}\right)}$$

where ${}_2F_1(x)$ is the hypergeometric function.

A.6.3 Properties

- The median and mode are 0. The mean is also 0 for $\nu > 1$ and undefined for $\nu \leq 1$
- The variance has three different cases: it can be undefined, infinite, or finite depending on ν:

$$\text{Var} = \begin{cases} \text{Undefined} & \nu \leq 1 \\ \infty & 1 < \nu \leq 2 \\ \frac{\nu}{\nu-2} & \nu > 2 \end{cases}$$

- The skewness is 0 for $\nu > 3$ and undefined otherwise.
- The excess kurtosis is $6(\nu - 3)$ for $\nu > 4$ and undefined otherwise.

The t-distribution can be changed so that as $\nu \to \infty$ it goes to a normal distribution with mean μ and variance σ^2. If z is t-distributed with parameter ν,

then $x = \mu + z\sigma$ will be a shifted and rescaled random variable so that it becomes normal with mean μ and variance σ^2 as $\nu \to \infty$.

A multivariate t-distribution also exists. In analogous fashion to the multivariate normal, there is a mean vector $\boldsymbol{\mu}$, a positive-definite matrix Σ, and $\nu > 0$ parameter. This distribution has PDF

$$f(\mathbf{x}|\boldsymbol{\mu}, \Sigma, \nu) = \frac{\Gamma\left[(\nu + p)/2\right]}{\Gamma(\nu/2)\nu^{p/2}\pi^{p/2}\left|\Sigma\right|^{1/2}}\left[1 + \frac{1}{\nu}(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right]^{-(\nu+p)/2}.$$

As $\nu \to \infty$, this distribution goes to a multivariate normal with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ.

A.7 Logistic Distribution

The logistic distribution resembles a normal distribution but it has thicker tails (i.e., the excess kurtosis is not zero). The distribution gets its name from the fact that its CDF is the logistic function. The logistic distribution has two parameters, the real-valued μ and the positive, real s.

A.7.1 Probability Density Function (PDF)

$$f(x|\mu, s) = \frac{e^{-\frac{x-\mu}{s}}}{\left(1 + e^{-\frac{x-\mu}{s}}\right)^2 s} = \frac{1}{4s}\,\mathrm{sech}^2\left(\frac{x - \mu}{2s}\right).$$

A.7.2 CDF

$$F(x|\mu, s) = \frac{1}{1 + e^{-\frac{x-\mu}{s}}} = \frac{1}{2} + \frac{1}{2}\,\tanh\left(\frac{x - \mu}{2s}\right).$$

A.7.3 Properties

- The mean, median, and mode are μ.
- The variance is proportional to s:

$$\mathrm{Var} = \frac{\pi^2}{3}s^2$$

- The skewness is 0.
- The excess kurtosis is 1.2.

A.8 Cauchy Distribution, Lorentz Distribution, or Breit-Wigner Distribution

The Cauchy distribution is a special case of the t-distribution with $\nu = 1$. It has a PDF that is finite everywhere, but has undefined mean, variance, skewness, and excess kurtosis. The distribution has two parameters, $x_0 \in \mathbb{R}$ and $\gamma > 0$. The median and mode of the distribution are at x_0.

A.8.1 Probability Density Function (PDF)

$$f(x|x_0, \gamma) = \frac{1}{\pi\gamma}\left[1 + \left(\frac{x - x_0}{\gamma}\right)^2\right]^{-1}.$$

A.8.2 CDF

$$F(x|x_0, \gamma) = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{x - x_0}{\gamma}\right).$$

A.9 Gumbel Distribution

The Gumbel distribution is often used to model the maximum of a random variable. It has two parameters $m \in \mathbb{R}$ and $\beta > 0$. It has positive skew and excess kurtosis. The CDF has one of the few occurrences of the exponential of an exponential.

A.9.1 Probability Density Function (PDF)

$$f(x|m, \beta) = \frac{1}{\beta}e^{-(z+e^{-z})}, \qquad \text{where} \quad z = \frac{x - m}{\beta}.$$

A.9.2 CDF

$$F(x|m,\beta) = e^{e^{-(z-\mu)/\beta}}.$$

A.9.3 Properties

- The mean of the Gumbel distribution is $\mu = m + \beta\,\gamma$, where $\gamma \approx 0.5772$ is the Euler-Mascheroni constant.
- The median is $m - \beta \log(\log 2)$.
- The mode is m.
- The variance is proportional to β^2:

$$\text{Var} = \frac{\pi^2}{6}\beta^2$$

- The skewness is positive:

$$\gamma_1 = \frac{12\sqrt{6}\,\zeta(3)}{\pi^3} \approx 1.14,$$

  where $\zeta(3) \approx 1.20205$ is Apéry's constant.
- The excess kurtosis is $\frac{12}{5}$.

A.10 Laplace Distribution, Double Exponential Distribution

The Laplace distribution resembles the normal distribution except that it has an absolute value in the exponential, rather than the quadratic exponent. It takes two parameters, $m \in \mathbb{R}$ and $b > 0$. It is a symmetric distribution about m and has nonzero excess kurtosis.

A.10.1 Probability Density Function (PDF)

$$f(x|m,b) = \frac{1}{2b}e^{-\frac{|x-m|}{b}}$$

A.10.2 CDF

$$F(x|m,b) = \begin{cases} \frac{1}{2}e^{\frac{x-m}{b}} & x < m \\ \frac{1}{2} + \frac{1}{2}e^{-\frac{x-m}{b}} & x \geq m \end{cases}$$

A.10.3 Properties

- The mean, median, and mode are all m.
- The variance is proportional to b^2:

$$\text{Var} = 2b^2$$

- The skewness is 0.
- The excess kurtosis is 3.

A.11 Uniform Distribution

A uniform random variable is equally likely to take on any value in the interval $[a, b]$, with $b > a$. If x is a uniformly-distributed random variable, we write $x \sim \mathcal{U}(a, b)$.

A.11.1 Probability Density Function (PDF)

$$f(x|a,b) = \begin{cases} \frac{1}{b-a} & x \in [a,b] \\ 0 & \text{otherwise} \end{cases}.$$

A.11.2 CDF

$$F(x|a,b) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & x \in [a,b) \\ 1 & x \geq b \end{cases}.$$

A.11.3 Properties

- The mean and median are $(a + b)/2$
- The mode is any value in $[a, b]$.
- The variance is $(b - a)^2/12$
- The skewness is 0.
- The excess kurtosis is $-6/5$.

We can define a standardized uniform random variable that has support on $[-1, 1]$ that we call $z \sim \mathscr{U}(-1, 1)$, by taking $x \sim \mathscr{U}(a, b)$ and defining

$$z = \frac{a + b - 2x}{a - b}.$$

A.12 Beta Distribution

The beta distribution describes random variables that take on values in the interval $[-1, 1]$ and can be described by two parameters $\alpha > -1$ and $\beta > -1$. If z is a beta-distributed random variable, we write $x \sim \mathscr{B}(\alpha, \beta)$.

A.12.1 Probability Density Function (PDF)

$$f(x|\alpha, \beta) = \frac{2^{-(\alpha+\beta+1)}}{\alpha + \beta + 1} \frac{\Gamma(\alpha + 1) + \Gamma(\beta + 1)}{\Gamma(\alpha + \beta + 1)} (1 + x)^{\beta} (1 - x)^{\alpha} \qquad x \in [-1, 1].$$

The PDF can also be expressed using the beta function,

$$\mathrm{B}(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha + \beta)},$$

as

$$f(x|\alpha, \beta) = \frac{2^{-(\alpha+\beta+1)}}{\mathrm{B}(\alpha + 1, \beta + 1)} (1 + z)^{\beta} (1 - z)^{\alpha} \qquad z \in [-1, 1].$$

A.12.2 CDF

$$F(x|\alpha, \beta) = I_x(\alpha + 1, beta + 1),$$

where $I_x(\alpha, \beta)$ is the regularized incomplete beta function:

$$I_x(a, b) = \frac{\mathrm{B}(x;\, a, b)}{\mathrm{B}(a, b)},$$

and

$$\mathrm{B}(x;\, a, b) = \int_0^x t^{a-1}\,(1-t)^{b-1}\,dt.$$

A.12.3 Properties

- The mean is

$$E[x] = \frac{-(\alpha+1) + (\beta+1)b}{\alpha+\beta+2}.$$

- The variance is

$$\mathrm{Var}(x) = \frac{4(\alpha+1)(\beta+1)}{(\alpha+\beta+2)^2(\alpha+\beta+3)}$$

We can scale a beta random variable, $z \sim \mathscr{B}(\alpha, \beta)$, to have support on the interval $[a, b]$ by writing

$$x = \frac{b-a}{2}z + \frac{a+b}{2}.$$

Note: the more common definition for beta random variables uses $\alpha' = \alpha + 1$ and $\beta' = \beta + 1$ and has the distribution supported on $[0, 1]$. In this work we choose our definition so that the PDF for the standardized gamma random variable is the weighting function in the orthogonality relation for Jacobi polynomials, which have a domain of $[-1.1]$.

A.13 Gamma Distribution

The gamma distribution describes random variables that take on values on the positive real line and can be described by two parameters $\alpha > -1$ and $\beta > 0$. If x is a gamma-distributed random variable, we write $x \sim \mathscr{G}(\alpha, \beta)$.

A.13.1 Probability Density Function (PDF)

$$f(x|\alpha,\beta) = \frac{\beta^{(\alpha+1)}x^{\alpha}e^{-\beta x}}{\Gamma(\alpha+1)}, \qquad x \in (0,\infty), \quad \alpha > -1, \ \beta > 0.$$

A.13.2 CDF

$$F(x|\alpha,\beta) = \frac{\gamma(\alpha+1,\beta x)}{\Gamma(\alpha+1)},$$

where $\gamma(a,b)$ is the lower incomplete Gamma function.

A.13.3 Properties

- The mean is $(\alpha+1)\beta^{-1}$.
- There is no simple formula for the median.
- The mode is $\alpha\beta^{-1}$ for $\alpha > 0$.
- The variance is $(\alpha+1)\beta^{-2}$.
- The skewness is

$$\gamma_1 = \frac{2}{\sqrt{\alpha+1}}.$$

- The excess kurtosis is

$$\text{Kur} = \frac{6}{\alpha+1}.$$

A standardized gamma random variable can be defined as $z = \beta x$ where $x \sim \mathcal{G}(\alpha,\beta)$ and $z\mathcal{G}(\alpha,1)$. Now the PDF for z is

$$f(z|\alpha) = \frac{z^{\alpha}e^{-z}}{\Gamma(\alpha+1)}, \qquad z \in (0,\infty), \quad \alpha > -1.$$

Note: the more common definition for gamma random variables uses $\alpha' = \alpha + 1$ but the same parameter β. In this work we choose our definition so that the PDF for the standardized gamma random variable is the weighting function in the orthogonality relation for generalized Laguerre polynomials.

A.14 Inverse Gamma Distribution

The inverse gamma distribution describes random variables whose reciprocal is a gamma random variable. Inverse gamma random variables take on values on the positive real line and can be described by two parameters $\alpha > 0$ and $\beta > 0$. If x is an inverse gamma-distributed random variable, we write $x \sim \text{IG}(\alpha, \beta)$. In this case we also have $x^{-1} \sim \mathcal{G}(\alpha + 1, \beta)$.

A.14.1 Probability Density Function (PDF)

$$f(x|\alpha, \beta) = \frac{\beta^{(\alpha)} x^{-\alpha-1} e^{-\frac{\beta}{x}}}{\Gamma(\alpha)}, \qquad x \in (0, \infty), \quad \alpha > 0, \ \beta > 0.$$

A.14.2 CDF

$$F(x|\alpha, \beta) = \frac{\Gamma\left(\alpha, \frac{\beta}{x}\right)}{\Gamma(\alpha)},$$

where $\Gamma(a, b)$ is the upper incomplete Gamma function.

A.14.3 Properties

- The mean is $(\alpha - 1)^{-1}\beta$ for $\alpha > 1$
- There is no simple formula for the median.
- The mode is $(\alpha + 1)^{-1}\beta$.
- The variance is $(\alpha - 1)^{-2}(\alpha - 2)^{-1}\beta^2$ for $\alpha > 2$.
- The skewness is

$$\gamma_1 = \frac{4\sqrt{\alpha - 21}}{\alpha - 3}, \qquad \alpha > 3.$$

- The excess kurtosis is

$$\text{Kur} = \frac{30\alpha - 66}{(\alpha - 3)(\alpha - 4)}, \qquad \alpha > 4.$$

A.15 Exponential Distribution

The exponential distribution is used for nonnegative random variables with a single, positive parameter λ. The exponential distribution is a special case of the gamma distribution with $\alpha = 0$ and $\beta = \lambda$. The exponential distribution is used to describe the distance between collisions for subatomic particles (e.g., light, neutrons, electrons) traveling in a given media with λ^{-1} being the average distance traveled between collisions.

A.15.1 Probability Density Function (PDF)

$$f(x|\lambda) = \lambda e^{-\lambda x}$$

A.15.2 CDF

$$F(x|m, b) = 1 - e^{-\lambda x}$$

A.15.3 Properties

- The mean is λ^{-1}.
- The median is $\lambda^{-1} \log 2$.
- The mode is 0.
- The variance is λ^{-2}.
- The skewness is 2.
- The excess kurtosis is 6.

References

Agnesi M (1748) Instituzioni analitiche ad uso della gioventú italiana. Nella Regia-Ducal Corte

Barth A, Schwab C, Zollinger N (2011) Multi-level Monte Carlo finite element method for elliptic PDEs with stochastic coefficients. Numer Math 119(1):123–161

Bastidas-Arteaga E, Soubra AH (2006) Reliability analysis methods. In: Stochastic analysis and inverse modelling, ALERT Doctoral School 2014, pp 53–77

Bayarri MJ, Berger JO, Cafeo J, Garcia-Donato G, Liu F, Palomo J, Parthasarathy RJ, Paulo R, Sacks J, Walsh D (2007) Computer model validation with functional output. Ann Stat 35(5):1874–1906

Bernoulli J (1713) Ars conjectandi, opus posthumum. Accedit Tractatus de seriebus infinitis, et epistola gallic scripta de ludo pilae reticularis. Thurneysen Brothers, Basel

Boyd JP (2001) Chebyshev and fourier spectral methods. Dover Publications, Mineola

Bratley P, Fox BL, Niederreiter H (1992) Implementation and tests of low-discrepancy sequences. ACM Trans Model Comput Simul 2(3):195–213

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Cacuci DG (2015) Second-order adjoint sensitivity analysis methodology (2nd-ASAM) for computing exactly and efficiently first- and second-order sensitivities in large-scale linear systems: I. Computational methodology. J Comput Phys 284:687–699

Carlin BP, Louis TA (2008) Bayesian methods for data analysis. Chapman & Hall/CRC texts in statistical science, 3rd edn. CRC Press, Boca Raton

Carpentier A, Munos R (2012) Adaptive stratified sampling for Monte-Carlo integration of differentiable functions. In: Advances in neural information processing systems, vol. 25, pp 251–259

Chowdhary K, Dupuis P (2013) Distinguishing and integrating aleatoric and epistemic variation in uncertainty quantification. ESAIM Math Model Numer Anal 47(3):635–662

Cliffe KA, Giles MB, Scheichl R, Teckentrup AL (2011) Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients. Comput Vis Sci 14(1):3–15

Collaboration OS et al (2015) Estimating the reproducibility of psychological science. Science 349(6251):aac4716

Collier N, Haji-Ali AL, Nobile F, Schwerin E, Tempone R (2015) A continuation multilevel Monte Carlo algorithm. BIT Numer Math 55(2):1–34

Collins GP (2009) Within any possible universe, no intellect can ever know it all. Scientific American

Constantine PG (2015) Active subspaces: emerging ideas for dimension reduction in parameter studies. SIAM spotlights, vol 2. SIAM, Philadelphia. ISBN 1611973864, 9781611973860

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0

Cook AH (1965) The absolute determination of the acceleration due to gravity. Metrologia 1(3):84–114
Denison DG, Mallick BK, Smith AF (1998) Bayesian MARS. Stat Comput 8(4):337–346
Denison DGT, Holmes CC, Mallick BK, Smith AFM (2002) Bayesian methods for nonlinear classification and regression. Wiley, Chichester
Der Kiureghian A, Ditlevsen O (2009) Aleatory or epistemic? Does it matter? Struct Saf 31(2):105–112
Farrell PE, Ham DA, Funke SW, Rognes ME (2013) Automated derivation of the adjoint of high-level transient finite element programs. SIAM J Sci Comput 35(4):C369–C393
Faure H (1982) Discrépance de suites associées à un système de numération (en dimension s). Acta Arith 41(4):337–351
Ferson S, Kreinovich V, Ginzburg L, Myers D, Sentz K (2003) Constructing probability boxes and dempster-shafer structures. Tech. Rep. SAND2002-4015, Sandia National Laboratories
Ferson S, Kreinovich V, Hajagos J, Oberkampf W, Ginzburg L (2007) Experimental uncertainty estimation and statistics for data having interval uncertainty. Tech. Rep. SAND2007-0939, Sandia National Laboratories
Fox BL (1986) Algorithm 647: implementation and relative efficiency of quasirandom sequence generators. ACM Trans Math Softw 12(4):362–376
Gal Y, Ghahramani Z (2016) Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In: International conference on machine learning, pp 1050–1059
Ghanem RG, Spanos PD (1991) Stochastic finite elements: a spectral approach. Springer, Berlin
Giles MB (2013) Multilevel monte carlo methods. In: Monte Carlo and Quasi-Monte Carlo methods 2012. Springer, Berlin, pp 83–103
Gilks W, Spiegelhalter D (1996) Markov chain Monte Carlo in practice. Chapman & Hall, London
Goh J (2014) Prediction and calibration using outputs from multiple computer simulators. PhD thesis, Simon Fraser University
Goh J, Bingham D, Holloway JP, Grosskopf MJ, Kuranz CC, Rutter E (2013) Prediction and computer model calibration using outputs from multifidelity simulators. Technometrics 55(4):501–512
Gramacy RB, Apley DW (2015) Local Gaussian process approximation for large computer experiments. J Comput Graph Stat 24(2):561–578
Gramacy RB, Lee HKH (2008) Bayesian treed Gaussian process models with an application to computer modeling. J Am Stat Assoc 103(483):1119–1130
Gramacy RB et al (2007) TGP: an R package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed Gaussian process models. J Stat Softw 19(9):6
Gramacy RB, Niemi J, Weiss RM (2014) Massively parallel approximate Gaussian process regression. SIAM/ASA J Uncertain Quantif 2(1):564–584
Gramacy RB, Bingham D, Holloway JP, Grosskopf MJ, Kuranz CC, Rutter E, Trantham M, Drake RP et al (2015) Calibrating a large computer experiment simulating radiative shock hydrodynamics. Ann Appl Stat 9(3):1141–1168
Griewank A, Walther A (2008) Evaluating derivatives: principles and techniques of algorithmic differentiation, vol 105. SIAM, Philadelphia
Gunzburger MD, Webster CG, Zhang G (2014) Stochastic finite element methods for partial differential equations with random input data. Acta Numer 23:521–650
Haldar A, Mahadevan S (2000) Probability, reliability, and statistical methods in engineering design. Wiley, New York
Halpern JY (2017) Reasoning about uncertainty. MIT Press, Cambridge
Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning. Data Mining, Inference, and Prediction, 2nd edn. Springer Science & Business Media, New York
Higdon D, Kennedy M, Cavendish JC, Cafeo JA, Ryne RD (2004) Combining field data and computer simulations for calibration and prediction. SIAM J Sci Comput 26(2):448
Hoerl AE, Kennard RW (1970) Ridge regression: biased estimation for nonorthogonal problems. Technometrics 12(1):55–67

Holloway JP, Bingham D, Chou CC, Doss F, Drake RP, Fryxell B, Grosskopf M, van der Holst B, Mallick BK, McClarren R, Mukherjee A, Nair V, Powell KG, Ryu D, Sokolov I, Toth G, Zhang Z (2011) Predictive modeling of a radiative shock system. Reliab Eng Syst Saf 96(9):1184–1193

Holtz M (2011) Sparse grid quadrature in high dimensions with applications in finance and insurance. Lecture notes in computational science and engineering, vol 77. Springer, Berlin

Humbird KD, McClarren RG (2017) Adjoint-based sensitivity analysis for high-energy density radiative transfer using flux-limited diffusion. High Energy Density Phys 22:12–16

Humbird K, Peterson J, McClarren R (2017) Deep jointly-informed neural networks. arXiv:170700784

John LK, Loewenstein G, Prelec D (2012) Measuring the prevalence of questionable research practices with incentives for truth telling. Psychol Sci 23(5):524–532. https://doi.org/10.1177/0956797611430953

Jolliffe I (2002) Principal component analysis. Springer series in statistics. Springer, Berlin

Jones S (2009) The formula that felled Wall St. The Financial Times

Kalos M, Whitlock P (2008) Monte Carlo methods. Wiley-Blackwell, Hoboken

Karagiannis G, Lin G (2017) On the Bayesian calibration of computer model mixtures through experimental data, and the design of predictive models. J Comput Phys 342:139–160

Kennedy MC, O'Hagan A (2000) Predicting the output from a complex computer code when fast approximations are available. Biometrika 87(1):1–13

Knupp P, Salari K (2002) Verification of computer codes in computational science and engineering. Discrete mathematics and its applications. CRC Press, Boca Raton

Kreinovich V, Ferson SA (2004) A new Cauchy-based black-box technique for uncertainty in risk analysis. Reliab Eng Syst Saf 85(1–3):267–279

Kreinovich V, Nguyen HT (2009) Towards intuitive understanding of the Cauchy deviate method for processing interval and fuzzy uncertainty. In: Proceedings of the 2015 conference of the international fuzzy systems association and the european society for fuzzy logic and technology conference, pp 1264–1269

Kreinovich V, Beck J, Ferregut C, Sanchez A, Keller G, Averill M, Starks S (2004) Monte-Carlo-type techniques for processing interval uncertainty, and their engineering applications. In: Proceedings of the workshop on reliable engineering computing, pp 15–17

Kurowicka D, Cooke RM (2006) Uncertainty analysis with high dimensional dependence modelling. Wiley, Chichester

Lahman S (2017) Baseball database. http://wwwseanlahmancom/baseball-archive/statistics

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444

Ling J (2015) Using machine learning to understand and mitigate model form uncertainty in turbulence models. In: 2015 IEEE 14th international conference on machine learning and applications (ICMLA). IEEE, Piscataway, pp 813–818

Lyness JN, Moler CB (1967) Numerical differentiation of analytic functions. SIAM J Numer Anal 4(2):202–210

Marsaglia G, Tsang WW, Wang J (2003) Evaluating Kolmogorov's distribution. J Stat Softw 8(18):1–4. https://doi.org/10.18637/jss.v008.i18

McClarren RG, Ryu D, Drake RP, Grosskopf M, Bingham D, Chou CC, Fryxell B, van der Holst B, Holloway JP, Kuranz CC, Mallick B, Rutter E, Torralva BR (2011) A physics informed emulator for laser-driven radiating shock simulations. Reliab Eng Syst Saf 96(9):1194–1207

Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculations by fast computing machines. J Chem Phys 21(6):1087–1092. https://doi.org/10.1063/1.1699114

National Academy of Science (2012) Building confidence in computational models: the science of verification, validation, and uncertainty quantification. National Academies Press, Washington

Oberkampf WL, Roy CJ (2010) Verification and validation in scientific computing, 1st edn. Cambridge University Press, New York

Owhadi H, Scovel C, Sullivan TJ, McKerns M, Ortiz M (2013) Optimal uncertainty quantification. SIAM Rev 55(2):271–345

Owhadi H, Scovel C, Sullivan T (2015) Brittleness of Bayesian inference under finite information in a continuous world. Electron J Stat 9(1):1–79
Peterson J, Humbird K, Field J, Brandon S, Langer S, Nora R, Spears B, Springer P (2017) Zonal flow generation in inertial confinement fusion implosions. Phys Plasmas 24(3):032702
Rackwitz R, Flessler B (1978) Structural reliability under combined random load sequences. Comput Struct 9(5):489–494
Raissi M, Karniadakis GE (2018) Hidden physics models: machine learning of nonlinear partial differential equations. J Computat Phys 357:125–141
Rasmussen CE, Williams CKI (2006) Gaussian processes for machine learning. MIT Press, Cambridge
Roache PJ (1998) Verification and validation in computational science and engineering. Hermosa Publishers, Albuquerque
Robert C, Casella G (2013) Monte Carlo statistical methods. Springer Science & Business Media, New York
Roberts GO, Gelman A, Gilks WR (1997) Weak convergence and optimal scaling of random walk metropolis algorithms. Ann Appl Probab 7(1):110–120
Saltelli A, Ratto M, Andres T, Campolongo F, Cariboni J, Gatelli D, Saisana M, Tarantola S (2008) Global sensitivity analysis: the primer. Wiley, Chichester
Saltelli A, Annoni P, Azzini I, Campolongo F, Ratto M, Tarantola S (2010) Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. Comput Phys Commun 181(2):259–270
Santner TJ, Williams BJ, Notz WI (2013) The design and analysis of computer experiments. Springer Science & Business Media, New York
Schilders WH, Van der Vorst HA, Rommes J (2008) Model order reduction: theory, research aspects and applications, vol 13. Springer, Berlin
Sobol IM (1967) On the distribution of points in a cube and the approximate evaluation of integrals. USSR Comput Math Math Phys 7(4):86–112
Spears BK (2017) Contemporary machine learning: a guide for practitioners in the physical sciences. arXiv:171208523
Stein M (1987) Large sample properties of simulations using latin hypercube sampling. Technometrics 29(2):143
Stripling HF, McClarren RG, Kuranz CC, Grosskopf MJ, Rutter E, Torralva BR (2013) A calibration and data assimilation method using the Bayesian MARS emulator. Ann Nucl Energy 52:103–112
Student (1908) The probable error of a mean. Biometrika 6:1–25
Tate DR (1968) Acceleration due to gravity at the national bureau of standards. J Res Natl Bur Stand Sect C Eng Instrum 72C(1):1
Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc Ser B (Methodological) 58(1):267–288
Townsend A (2015) The race for high order Gauss–Legendre quadrature. SIAM News, pp 1–3
Trefethen LN (2013) Approximation theory and approximation practice. Other titles in applied mathematics. SIAM, Philadelphia
Wagner JC, Haghighat A (1998) Automated variance reduction of Monte Carlo shielding calculations using the discrete ordinates adjoint function. Nucl Sci Eng 128(2):186–208
Wang Z, Navon IM, Le Dimet FX, Zou X (1992) The second order adjoint analysis: theory and applications. Meteorol Atmos Phys 50(1–3):3–20
Wilcox LC, Stadler G, Bui-Thanh T, Ghattas O (2015) Discretely exact derivatives for hyperbolic PDE-constrained optimization problems discretized by the discontinuous Galerkin method. J Sci Comput 63(1):138–162
Wolpert DH (2008) Physical limits of inference. Phys D Nonlinear Phenom 237(9):1257–1281
Zheng W, McClarren RG (2016) Emulation-based calibration for parameters in parameterized phonon spectrum of ZrH_x in TRIGA reactor simulations. Nucl Sci Eng 183(1):78–95
Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. J R Stat Soc Ser B (Stat Methodol) 67(2):301–320

Index

R. G. McClarren, *Uncertainty Quantification and Predictive Computational Science*, https://doi.org/10.1007/978-3-319-99525-0