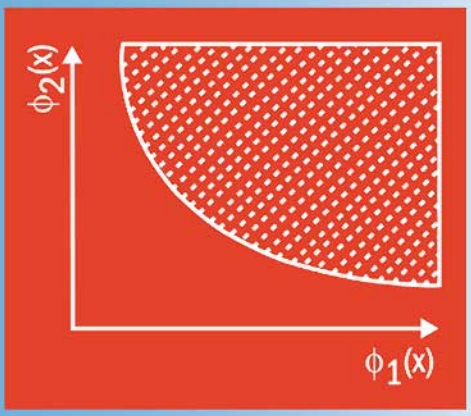

Multiobjective Optimisation and Control

G.P. Liu
J.B. Yang
J.F. Whidborne

Multiobjective Optimisation and Control

ENGINEERING SYSTEMS MODELLING AND CONTROL SERIES

Series Editor: **Professor D H Owens**
University of Sheffield, UK

3. Controllability Analysis and Control Structure Selection
Y Cao, D Rossiter and **D H Owens ***
4. Multiobjective Optimisation and Control
G P Liu, J B Yang and **J F Whidborne**

* forthcoming

Multiobjective Optimisation and Control

G P Liu, *University of Nottingham, UK*
J B Yang, *UMIST, UK*
J F Whidborne, *King's College London, UK*

RESEARCH STUDIES PRESS LTD.
Baldock, Hertfordshire, England

RESEARCH STUDIES PRESS LTD.

16 Coach House Cloisters, 10 Hitchin Street, Baldock, Hertfordshire, SG7 6AE, England
www.research-studies-press.co.uk

and

Institute *of* **Physics** PUBLISHING, Suite 929, The Public Ledger Building,

150 South Independence Mall West, Philadelphia, PA 19106, USA

Marketing:

Institute *of* **Physics** PUBLISHING, Dirac House, Temple Back, Bristol, BS1 6BE, England
www.bookmarkphysics.iop.org

Distribution:

NORTH AMERICA

AIDC, 50 Winter Sport Lane, PO Box 20, Williston, VT 05495-0020, USA
Tel: 1-800 632 0880 or outside USA 1-802 862 0095, Fax: 802 864 7626, E-mail: orders@aidcvt.com

UK AND THE REST OF WORLD

Marston Book Services Ltd, P.O. Box 269, Abingdon, Oxfordshire, OX14 4YN, England
Tel: + 44 (0)1235 465500 Fax: + 44 (0)1235 465555 E-mail: *direct.order@marston.co.uk*

Library of Congress Cataloguing-in-Publication Data

Liu, G.P.(Guo Ping), 1962 –
Multiobjective optimisation and control / G.P. Liu, J.B. Yang, J.F. Whidborne.
p. cm. – (Engineering systems modelling and control series ; 4)
Includes bibliographical references and index.
ISBN: 0-86380-264-8
1. Mathematical optimisation. 2. Nonlinear programming. 3. Multiple criteria decision making. I. Yang, Jian-Bo, 1961 – II Whidborne, J.F. (James Ferris), 1960- III. Title. IV. Series.

QA402.5 .L57 2001
519.3--dc21 2001019195

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library.

ISBN 0 86380 264 8

Printed in Great Britain by SRP Ltd., Exeter

Dedication

To Weihong and Louise
(G.P. Liu)

To Dong-Ling and Lin
(J.B. Yang)

To Cécile
(J.F. Whidborne)

Contents

Preface **xiii**

Symbols and Abbreviations **xv**

1 Introduction **1**
1.1 Multiobjective Optimisation 1
1.1.1 Constrained Optimisation 1
1.1.2 Conventional Multiobjective Optimisation 2
1.1.3 Method of Inequalities 7
1.1.4 Multiobjective Genetic Algorithms 9
1.2 Multiobjective Control 10
1.2.1 Conflicts and Trade-offs in Control Systems 11
1.2.2 Multiobjective Robust Control 14
1.2.3 Multiobjective Critical Control 15
1.2.4 Multiobjective Eigenstructure Assignment 16
1.2.5 Multiobjective PID Control 16
1.2.6 Multiobjective Optimisation of Controller Implementations 17
1.2.7 Multiobjective Nonlinear Identification 18
1.2.8 Multiobjective Fault Detection 19
1.3 Outline of the Book 20

2 Nonlinear Optimisation **23**
2.1 One-Dimensional Optimisation 23
2.1.1 The Dichotomy Method with Derivatives 23
2.1.2 The Dichotomy Method without Derivatives 25
2.1.3 The Fibonacci Method 26
2.1.4 The Golden Section Search Method 31
2.2 Optimisation Conditions 32
2.2.1 Necessary Conditions for Local Optimality 32
2.2.2 Sufficient Conditions for Local Optimality 34

2.3 Unconstrained Optimisation Methods 34
2.3.1 Steepest Decent Method 34
2.3.2 Newton's Method 38
2.3.3 Quasi-Newton's Methods 41
2.4 Summary 44

3 Constrained Optimisation 45
3.1 Introduction 45
3.2 Optimality Conditions 46
3.2.1 Basic Concepts 46
3.2.2 Kuhn-Tucker Necessary Condition 47
3.2.3 Second Order Sufficient Conditions 48
3.3 Primal Methods 52
3.3.1 Sequential Linear Programming 52
3.3.2 Sequential Quadratic Programming 55
3.4 Dual Methods 60
3.4.1 Lagrangean Methods 61
3.4.2 Method of Exterior Penalties 64
3.4.3 Method of Interior Penalties 68
3.5 Summary 71

4 Multiple Objective Optimisation 73
4.1 Introduction 73
4.2 Basic Concepts and Methods 74
4.2.1 Concepts and Definitions 74
4.2.2 Method Classification 77
4.2.3 Simple Weighting Method 78
4.3 p-Norm Methods 82
4.3.1 Minimax (Ideal Point) Method 82
4.3.2 Goal Attainment Method 89
4.3.3 Goal Programming 91
4.3.4 The Minimax Reference Point Method 95
4.4 Interactive Methods 103
4.4.1 Geoffrion's Method 103
4.4.2 The STEM Method 108
4.4.3 The ISTM Method 112
4.4.4 The Gradient Projection Method 116
4.5 Summary 123

5 Genetic Algorithms and Optimisation 125
5.1 Introduction 125

5.2 What are Genetic Algorithms 125
5.3 Basic Structure of Genetic Algorithms 127
5.4 Population Representation and Initialisation 129
5.4.1 Binary Representation 129
5.4.2 Real-Valued Representation 129
5.4.3 Initialisation 130
5.5 Fitness Functions 130
5.6 Selection 132
5.6.1 Roulette Wheel Selection Methods 133
5.6.2 Stochastic Universal Sampling 134
5.7 Crossover 135
5.7.1 Single-Point Crossover 135
5.7.2 Multi-Point Crossover 135
5.7.3 Uniform Crossover 136
5.7.4 Other Crossover Operators 137
5.7.5 Intermediate Recombination 137
5.7.6 Line Recombination 138
5.8 Mutation 138
5.9 Reinsertion and Termination 140
5.9.1 Reinsertion 140
5.9.2 Termination 141
5.10 Multiobjective Optimisation with GAs 141
5.10.1 Constrained Optimisation 141
5.10.2 Non-Pareto Optimisation 142
5.10.3 Pareto-Based Optimisation 143
5.11 An Example 143
5.12 Summary 146

6 Robust Control System Design by Mixed Optimisation 147
6.1 Introduction 147
6.2 An H_∞ Loop Shaping Design Procedure 148
6.2.1 Overview 148
6.2.2 Preliminaries 149
6.2.3 Normalised Left Coprime Factorisation 150
6.2.4 Coprime Factor Robust H_∞ Stability Problem 151
6.2.5 A Loop-Shaping Design Procedure (LSDP) 153
6.2.6 Example – The Inverted Pendulum 156
6.3 Mixed-Optimisation for the LSDP 160
6.3.1 MATLAB Implementation - The MODCONS Toolbox 162
6.4 Example – The Distillation Column 163

6.5 Example – High Speed EMS Maglev Vehicle 167
6.6 Summary 175

7 Multiobjective Control of Critical Systems 177
7.1 Introduction 177
7.2 Critical Control Systems 178
7.3 Critical System Descriptions 180
7.4 Input Spaces of Systems 183
7.5 Multiobjective Critical Control 184
7.6 Control Design of SISO Critical Systems 186
7.7 Control Design of MIMO Critical Systems 191
7.8 An Example 197
7.9 Summary 198

8 Multiobjective Control Using Eigenstructure Assignment 199
8.1 Introduction 199
8.2 What is Eigenstructure Assignment 200
8.3 Allowable Eigenvector Subspaces 203
8.4 Parametric Eigenstructure Assignment 206
8.5 Multiobjective Eigenstructure Assignment 210
8.6 Controller Design Using the Method of Inequalities 214
8.7 Controller Design Using Genetic Algorithms 217
8.8 Summary 221

9 Multiobjective PI Controller Design for a Gasifier 223
9.1 Introduction 223
9.2 Modelling of the Gasifier 224
9.3 System Specifications of the Gasifier 226
9.4 Multiobjective PI Control Formulation 228
9.5 Multiobjective Optimal-Tuning PI Control 230
9.6 Simulation Results and Discussions 231
9.7 Summary 238

10 Multiobjective PID Controller Implementation Design 239
10.1 Introduction 239
10.2 FWL Fixed-Point Representation 241
10.2.1 A Linear System Equivalence Completion Problem 242
10.3 MOGA for Optimal FWL Controller Structures 245
10.3.1 Multiobjective Genetic Algorithm 245
10.3.2 Procedure Outline 248
10.3.3 Encoding of Solution Space 249

10.4 Example – Steel Rolling Mill System 249
10.4.1 Performance indices 250
10.4.2 Nominal Plant Model 251
10.4.3 Controller 251
10.4.4 Design Results 252
10.5 Example – IFAC93 Benchmark Design 252
10.5.1 Performance Indices 253
10.5.2 Nominal Plant Model and Controller 253
10.5.3 Design Results 254
10.6 Summary 257

11 Multiobjective Nonlinear Identification 259
11.1 Introduction 259
11.2 Neural Networks 261
11.3 Gaussian Radial Basis Function Networks 263
11.4 Nonlinear Modelling with Neural Networks 264
11.5 Modelling Selection by Genetic Algorithms 265
11.6 Multiobjective Identification Criteria 266
11.7 Multiobjective Identification Algorithm 268
11.8 Examples 271
11.8.1 Example 1 272
11.8.2 Example 2 277
11.9 Summary 279

12 Multiobjective Fault Diagnosis 281
12.1 Introduction 281
12.2 Overview of Robust Fault Diagnosis 282
12.3 Observer Based Fault Diagnosis 284
12.4 Multiple Objectives of Fault Diagnosis 286
12.5 Disturbance Distribution and Fault Isolation 287
12.6 Parameterisation of Fault Diagnosis 288
12.7 Multiobjective Fault Diagnosis 290
12.8 An Example 291
12.9 Summary 295

Bibliography 297

Index 317

Preface

Multiobjective optimisation has been widely applied to control systems to achieve a number of design objectives that are often conflicting. This book is intended to cover the central concepts of multiobjective optimisation and control techniques. It is designed for either self-study by professionals or classroom work at the undergraduate or postgraduate level for students who have a technical background in control engineering and engineering mathematics. Like the field of optimisation and control, which involves many classical and advanced disciplines, the book should be useful to control system designers and researchers and other specialists from the host of disciplines from which practical optimisation and control applications are drawn.

The prerequisites for convenient use of the book are relatively modest; the prime requirement being some familiarity with introductory elements of linear algebra and linear control systems. Certain sections and developments do assume some knowledge of more advanced concepts of control systems, but the text is structured so that the mainstream of the development can be faithfully pursued without reliance on this more advanced background material.

Although the book covers primarily material that is now fairly standard, it is intended to reflect theoretical and practical insights of multiobjective optimisation and control. These provide structure to what might otherwise be simply a collection of techniques and results, and this is valuable both as a means for learning existing material and for developing new results. One major insight of this type is the connection between the purely analytical character of an optimisation problem and the behaviour of control techniques used to design a practical system.

Much of the work described in this book is based on a series of publications by the authors. The following publishers are gratefully acknowledged for permission to publish aspects of the authors' work appeared in their journals: The Institution of Electrical Engineers, Taylor and Francis Ltd., The Institute of Electrical and Electronics Engineers, and The Institute of Mechanical Engineers. The authors would like to thank the following people whom they have

each collaborated with on some of the material for the book: Professor Steve Billings, Dr Jie Chen, Dr Steve Daley, Dr Roger Dixon, Dr Dawei Gu, Dr Visakan Kadirkamanathan, Professor Duan Li, Professor Ron Patton, Professor Ian Postlethwaite and Dr Vladimir Zakian. Guoping Liu wishes to thank his wife Weihong and daughter Louise for their constant encouragement, understanding and tolerance during the preparation of the manuscript. Jian-Bo Yang wishes to thank his wife Dong-Ling and daughter Lin for their sharing his interests and providing support in writing the book. James Whidborne wishes to thank his wife Cécile and daughters Gwenaëlle and Camille for their support, love and understanding. He would also like to thank Raza Samar, Aamer Bhatti and the other organisers of the 25th International Nathiagali Summer College for the opportunity to present some of the material contained in this book to the college in 2001.

G. P. Liu
School of Mechanical, Materials, Manufacturing
Engineering and Management
University of Nottingham
Nottingham NG7 2RD
United Kingdom

J. B. Yang
Manchester School of Management
University of Manchester Institute
of Science and Technology
Manchester M60 1QD
United Kingdom

J. F. Whidborne
Department of Mechanical Engineering
King's College London
London WC2R 2LS
United Kingdom

January 2002

Symbols and Abbreviations

The symbols and abbreviations listed here are used unless otherwise stated.

$\mathcal{C}$	field of complex numbers
$(\cdot)^*$	complex conjugate
DM	decision maker
$\delta x(t)$	$\dot{x}(t)$ for continuous time and $x(t+1)$ for discrete time
EA	eigenstructure assignment
FDI	fault detection and isolation
FWL	finite word-length
GA	genetic algorithm
GAs	genetic algorithms
GRBF	Gaussian radial basis function
$\| H \|_n$	n-norm of the function $H(s)$
ISTM	interactive step trade-off method
im	imaginary part of a complex number
$\inf\{\cdot\}$	infimum
j	imaginary indictor of a complex
K	system controller
L	left eigenvector matrix
$\mathcal{L}[.]$	Laplace transform operation
λ	eigenvalue
λ_i	i-th eigenvalue
Λ	closed-loop eigenvalue set
LHP	left half-plane
LSDP	loop-shaping design procedure
LQG	linear quadratic Gaussian
MBP	moving boundaries process
MIMO	multi-input multi-output
MLP	multilayer perceptron
MOGA	multiobjective genetic algorithm
MoI	method of inequalities
Max	Maximum
$\max\{\cdot\}$	maximum
Min	Minimum

$\min\{\cdot\}$	minimum
$\|\cdot\|$	modulus
$\mathcal{N}$	integer numbers
$\mathcal{N}^+$	non-negative integer numbers
ω	angular frequency
PI	proportional-integral
PID	proportional-integral-derivative
p	design parameter
$\frac{\partial}{\partial x}$	partial derivative with respect to x
ϕ	performance function
R	right eigenvector matrix
RBF	radial basis function
$\mathcal{R}$	field of real numbers $(-\infty, \infty)$
$\mathcal{R}^+$	field of non-negative real numbers $[0, \infty)$
RHP	right half-plane
re	real part of a complex number
SISO	single-input single-output
SLP	sequential linear programming
STEM	step method
s	Laplace operator
$s.t.$	satisfy
$\bar{\sigma}(M)$	maximum singular value of the matrix M
$\underline{\sigma}(M)$	minimum singular value of the matrix M
$\sup\{\cdot\}$	supremum
T	sampling interval in sampled-data systems
t	time
u	system control input
x	system state vector
y	system output

Chapter 1

Introduction

In control system design there are often a number of design objectives to be considered. The objectives are sometimes conflicting and no design exists which can be considered best with respect to all objectives. Hence, there is an inevitable trade-off between design objectives, for example, between an output performance objective and stability robustness. These considerations have led to the study of multiobjective optimisation methods for control systems.

1.1 Multiobjective Optimisation

The general multiobjective optimisation problem has been studied in great detail, both in the control community (*e.g.* Zadeh, 1963; Gembicki and Haimes, 1975; Lin, 1976; Giesy, 1978; Tabak *et al.*, 1979; for a review, see Ng, 1989) and the operational research community (*e.g.* Cohon, 1978). Constraint satisfaction and multiobjective optimisation are two aspects of the same problem. Both of them involve the simultaneous optimisation of many objective functions. Constraints can often be considered as hard objectives, which need to be satisfied before the optimisation of the remaining soft objectives takes place. On the other hand, problems characterised by a number of soft objectives are often re-formulated as constrained optimisation problems to find their solutions. Both multiobjective and constraint optimisation have a long research history.

1.1.1 Constrained Optimisation

Practical problems are often constrained by a number of restrictions imposed on the decision variables. Generally speaking, there are two types of constraints. The first one is the domain constraints which express the domain of definition of the objective function. In control systems, for example, closed-loop system stability is a domain constraint, because most control perfor-

mance measures are not defined for unstable systems. The second is the preference constraints that impose further restrictions on the solution of the problem according to knowledge at a higher level. For example, a given stability margin expresses a preference of the designer.

Constraints can usually be described in terms of function inequalities of the type

$$\phi(p) \leq \varepsilon \tag{1.1.1}$$

where $\phi(.)$ is a generally nonlinear real-valued function of the decision variable vector p and ε is a constant value. The inequalities may also be strict by $<$ rather than $\leq$. Equality constraints can be seen as particular cases of inequality constraints.

Without loss of generality, the constrained optimisation problem is that of minimising a scalar function ϕ_0 of some decision variable vector p in a universe U, subject to n conditions involving p. Then, the constrained optimisation problem can eventually be expressed as the following:

$$\min_{p \in U} \phi_0(p) \tag{1.1.2}$$

subject to

$$\phi_i(p) \leq \varepsilon_i, \quad \text{for} \quad i = 1, 2, ..., n \tag{1.1.3}$$

where it is assumed that there is at least one point in U which satisfies all constraints.

In many cases, it is very difficult to satisfy all constraints. When constraints cannot be all simultaneously satisfied, the problem is often deemed to admit that there is no solution. According to the extent to which each constraint is violated, the violated constraints then need to be considered in order to relax the preference constraints. It is clear that the problem of satisfying a number of violated inequality constraints is the multiobjective problem of minimising the associated functions until given values (goals) are reached.

1.1.2 Conventional Multiobjective Optimisation

A number of problems are also characterised by several non-commensurable and often competing measures of performance, or objectives. Without loss of generality, the multiobjective optimisation problem is the problem of simultaneously minimising the n objectives $\phi_i(p)$, $k = 1, ..., n$ of a variable vector p in a universe U, that is

$$\min_{p \in U} (\phi_1(p), \phi_2(p), ..., \phi_n(p)) \tag{1.1.4}$$

In general, the problem has no optimal solution that could optimise all objectives simultaneously. But there exist a set of equally efficient, or non-inferior,

alternative solutions, known as the Pareto-optimal set (Ben-Tal, 1980). To select a suitable compromise solution from all non-inferior alternatives, a decision process is necessary. Depending on how the computation and the decision processes are combined in the search for compromise solutions, there exist three broad classes of multiobjective optimisation methods (Hwang and Masud, 1979): *a priori* articulation of preferences, *a posteriori* articulation of preferences, and progressive articulation of preferences.

In *a priori* articulation of preferences, the decision-maker expresses preferences in terms of an aggregating function, which combines individual objective values into a single utility value and ultimately makes the problem single-objective prior to optimisation.

In *a posteriori* articulation of preferences, the decision-maker is presented by the optimiser with a set of non-inferior solution candidates before expressing any preferences. The trade-off solution is chosen from that set.

In progressive articulation of preferences, decision making and optimisation occur at interactive steps. At each step, the decision-maker provides partial preference information to the optimiser, which in turn generates better alternatives according to the information received.

Preference articulation implicitly defines a so-called utility function that discriminates between candidate solutions. Although it can be very difficult to formalise such a utility function in every detail, approaches based on weighting coefficients, priorities and goal values have been widely used.

Weighting coefficients are real values that express the relative importance of the objectives and balance their involvement in the overall utility measure. For example, the weighted sum approach is based on objective weighting (Hwang and Masud, 1979).

Priorities are integer values that determine the order of objectives to be optimised, according to their importance. The lexicographic method (Ben-Tal, 1980), for example, requires to assign different priorities to all objectives.

Goal values give an indication of desired levels of performance in each objective dimension. Goals can be interpreted in various ways. They may represent minimum levels of performance to be attained, utopian performance levels to be approximated, or ideal performance levels to be matched as closely as possible. Generally speaking, goals are easier to set than weights and priorities because they relate more closely to the final solution of the problem.

Pareto Optimality

As we have seen, control systems design problems, along with the majority of engineering design problems are multiobjective, in that there are several conflicting design aims which need to be simultaneously achieved. If these design aims are expressed quantitatively as a set of n design objective functions $\{\phi_i(p) : i = 1 \ldots n\}$, where p denotes the design parameters chosen by the designer, the design problem could be formulated as a multiobjective op-

timisation problem:

$$\min_{p \in \mathcal{P}} \{\phi_i(p), \text{ for } i = 1 \ldots n\} \tag{1.1.5}$$

where $\mathcal{P}$ denotes the set of possible design parameters p.

In most cases, the objective functions are in conflict, so the reduction of one objective function leads to the increase in another. Subsequently, the result of the multiobjective optimisation is known as a Pareto-optimal solution (Pareto, 1906). A Pareto-optimal solution has the property that it is not possible to reduce any of the objective functions without increasing at least one of the other objective functions.

A point $p^* \in \mathcal{P}$ is defined as being Pareto-optimal if and only if there exists no other point $p \in \mathcal{P}$ such that

a) $\phi_i(p) \leq \phi_i(p^*)$ for all $i = 1, \ldots, n$ and
b) $\phi_j(p) < \phi_j(p^*)$ for at least one j.

The Pareto-optimal set is illustrated in Figure 1.1 for the case where there are two objective functions, *i.e.* $n = 2$. A point lying in the interior of the attainable set is sub-optimal, since both ϕ_1 and ϕ_2 can be both reduced. A point lying on the boundary of the set, *i.e.* in the Pareto-optimal set, requires ϕ_1 to be increased if ϕ_2 is to be decreased and *vice versa*. A solution to a multiobjective optimisation problem must lie on this boundary.

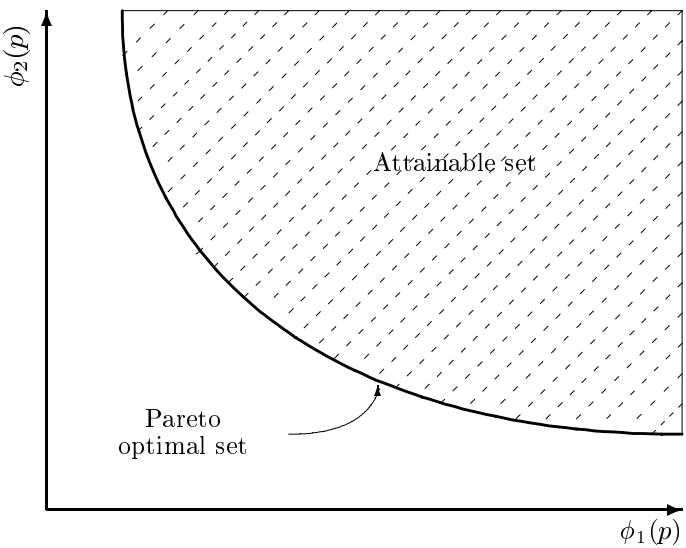

Figure 1.1: A Pareto-optimal set

Weighted Sum Method

The weighted sum method converts the multiobjective problem of minimising the objectives into a scalar one by constructing a weighted sum of all the objectives, that is,

$$\min_{p \in U} \sum_{k=1}^{n} w_k \phi_k(p) \tag{1.1.6}$$

where w_k is the weighting coefficient. This problem can then be solved using a standard unconstrained optimisation algorithm. The main point in the problem here is to attach the weighting coefficients to each of the objectives. The weighting coefficients do not necessarily represent the relative importance of the objectives or allow trade-offs between the objectives to be expressed. Further, the non-inferior solution boundary may be non-convex so that certain solutions are not accessible using this method.

ε-Constraint Method

ε-constraint method is a procedure which overcomes some of the convexity problems of the weighted sum technique. It involves minimising a primary objective and expressing the other objectives in the form of inequality constraints

$$\min_{p \in U} \phi_i(p) \tag{1.1.7}$$

subject to

$$\phi_k(p) \leq \varepsilon_k, \quad \text{for} \quad k = 1, 2, ..., n, k \neq i \tag{1.1.8}$$

This method is able to identify many non-inferior solutions on a non-convex boundary, which are not obtainable using the weighted sum approach. However, a problem with this approach is a suitable selection of ε_k to ensure a feasible solution. A further disadvantage of this method is that the use of hard constraints is rarely adequate for expressing true design objectives. There are also some similar methods, such as that of Waltz (1967). They prioritise the objectives and the optimisation proceeds with reference to these priorities and allowable bounds. Here, it is difficult to express such information at an early stage of the optimisation cycle.

Goal Attainment Method

The goal attainment method involves expressing a set of design goals, which is associated with a set of objectives (Gembicki, 1974). The problem formulation allows the objectives to be under- or over-achieved so that the initial design goals can be set to be relatively imprecise by the designer. The relative degree of under- or over-achievement of the goals is controlled by a weighting

coefficient vector. A standard optimisation problem using this method is formulated as

$$\min_{f \in \mathcal{R}, p \in U} f \tag{1.1.9}$$

$$\phi_k(p) - w_k f \leq \phi_k^*, \quad k = 1, 2, ..., n \tag{1.1.10}$$

where ϕ_k is the objective, ϕ_k^* the design goal, and w_k the weighting coefficient. The goal attainment method provides a convenient intuitive interpretation of the design problem, which is solvable using standard optimisation procedures. An advantage of the goal attainment method is that it can be posed as a nonlinear programming problem. Illustrative examples of using goal attainment method in control system design can be found in Fleming (1986).

Nonlinear Programming

Many approaches based on nonlinear programming to solve variations on the general problem given by (1.1.5) have been proposed. For example, Kreisselmeier and Steinhauser (1983) propose an approach with some similarities to the Method of Inequalities (see section 1.1.3, Maciejowski, 1989, pp 346-351). The approaches by Polak, Mayne and co-authors (*e.g.* Mayne *et al.*, 1981; Polak and Mayne, 1976; see Polak *et al.* (1984) for a review) provide some efficient methods for solving many multiobjective problems.

Convex Optimisation

It has been recognised (Boyd and Barratt, 1991; Polak and Salcudean, 1989) that via a Youla parameterisation, many multiobjective control problems can be posed as convex optimisation problems. Efficient methods have been developed to solve such problems. In addition, a number of problems can be posed as linear matrix inequalities (Scherer *et al.*, 1997).

Interactive Multiobjective Programming

It has been recognised by many researchers that, generally, the multiobjective design process is interactive, with the computer providing information to the designer about conflicting design requirements, and the designer adjusting the problem specification to explore the various possible solutions to the problem. The design process is thus a two way process, with the computer providing information to the designer about conflicting design requirements, and the designer making decisions about the 'trade-offs' between design requirements based on this information as well as on the designer's knowledge, experience and intuition about the particular problem. The designer can be supported in this role by various graphical displays (Ng, 1989) which provide information about the progress of the search algorithm and about the conflicting design

requirements. Some packages have been developed to provide such an environment, such as DELIGHT (Nye, 1983; Nye and Tits, 1986), ANDECS (Grubel *et al.*, 1993) and MODCONS (Whidborne *et al.*, 1995, 1996).

1.1.3 Method of Inequalities

The problem with multiobjective optimisation is that there is generally a very large set of Pareto-optimal solutions. Subsequently there is a difficulty in representing the set of Pareto-optimal solutions and in choosing the solution which is the best design.

To overcome this difficulty, the design problem can be formulated as the method of inequalities (MoI) (Zakian and Al-Naib, 1973) (see also, Patel and Munro, 1982; Maciejowski, 1989; Whidborne and Liu, 1993). In the method of inequalities, the problem is expressed as a set of algebraic inequalities which need to be satisfied for a successful design. The design problem is expressed as: find p such that inequalities

$$\phi_i(p) \leq \varepsilon_i, \quad \text{for} \quad i = 1,\ 2, \ldots,\ n \tag{1.1.11}$$

are satisfied, where ε_i are real numbers, $p \in \mathcal{P}$ is a real vector $(p_1, p_2, \ldots, p_q)$ chosen from a given set $\mathcal{P}$ and ϕ_i are real functions of p.

The design goals ε_i are chosen by the designer and represent the largest tolerable values of the objective functions ϕ_i. The aim of the design is to find a p that simultaneously satisfies the set of inequalities.

For control system design, the functions $\phi_i(p)$ may be functions of the system step response, for example the rise-time, overshoot or the integral absolute error, or functions of the frequency response, such as the bandwidth. They can also represent measures of the system stability and robustness, such as the maximum real part of the closed-loop poles. Additional inequalities which arise from the physical constraints of the system can also be included, to restrict for example, the maximum control signal. The design parameter, p, may parameterise a controller with a particular structure (*e.g.* Whidborne, 1992 or Whidborne, 1993). For example, $p = (p_1, p_2)$ could parameterise a PI controller $p_1 + p_2/s$. Alternatively, p, may parameterise the weighting functions required by analytical optimisation methods (Whidborne *et al.*, 1994b, 1995b; Postlethwaite *et al.*, 1994) to provide a mixed optimisation approach.

The actual solution to the set of inequalities (1.1.11) may be obtained by means of numerical search algorithms. Generally, the design process is interactive, with the computer providing information to the designer about conflicting design requirements, and the designer adjusting the inequalities to explore the various possible solutions to the problem. The progress of the search algorithm should be monitored, and, if a solution is not found, the designer may either change the starting point, relax some of the desired bounds ε or change the design configuration. Alternatively, if a solution is found easily, to improve the quality of the design, the bounds could be tightened or additional

design objectives could be included in (1.1.11). The design process is thus a two way process, with the MoI providing information to the designer about conflicting design requirements, and the designer making decisions about the 'trade-offs' between design requirements based on this information as well as on the designer's knowledge, experience and intuition about the particular problem. The designer should be supported in this role by various graphical displays (Ng, 1989) which provide information about the progress of the search algorithm and about the conflicting design requirements.

The original algorithm for the MoI, proposed by Zakian and Al-Naib (1973), is known as the moving boundaries process (MBP). This algorithm uses Rosenbrock's hill-climbing (Rosenbrock, 1960) to perform a local search to try and improve on at least one of the unsatisfied performance indices. This algorithm is simple, robust and effective and has worked well over the years, however, it does rely on a great deal of user-interaction to provide assistance when local minima are approached. The success of the algorithm is very dependent on being provided with a good starting point. This does have the advantage of forcing the user to carefully analyse the problem before the design is completed, and hence guarding against 'unreasonable' solutions.

Ng (1989) has proposed another algorithm, which is also based on a hill-climbing method, namely Nelder and Mead's modified simplex method (Nelder and Mead, 1965). It provides a dynamic minimax formulation which makes all indices with unsatisfied bounds equally active at the start of each iteration of the Nelder-Mead algorithm, so that at the k-th iteration, *one step* of the following minimax problem is solved:

$$\min_{p} \psi(p) \tag{1.1.12}$$

where

$$\psi(p) = \max_{i} \left\{ \bar{\phi}_i = \frac{\phi_i(p) - \phi_i^g}{\phi_i^b - \phi_i^g},\ i = 1, 2, \ldots, n \right\} \tag{1.1.13}$$

$$\phi_i^g = \begin{cases} \varepsilon_i & \text{if } \phi_i(p^k) > \varepsilon_i \\ \phi_i(p^k) - \delta & \text{if } \phi_i(p^k) \leq \varepsilon_i \end{cases} \tag{1.1.14}$$

$$\phi_i^b = \begin{cases} \phi_i(p^k) & \text{if } \phi_i(p^k) > \varepsilon_i \\ \varepsilon_i & \text{if } \phi_i(p^k) \leq \varepsilon_i \end{cases} \tag{1.1.15}$$

and δ is set to a small positive number. This algorithm also appears to work well, but is also very dependent on being provided with a good starting point.

There are a number of other search algorithms for the MoI. An algorithm based on simulated annealing can be found in Whidborne *et al.* (1996a) or Chipperfield *et al.* (1999). The goal attainment method (Gembicki and Haimes, 1975; Fleming and Pashkevich, 1986) can also be used for solving inequalities and this algorithm has been included in the MATLAB Optimisation Toolbox (Grace, 1994). Other algorithms have been developed for the solution of functional inequalities (see for example, Becker *et al.*, 1979).

1.1.4 Multiobjective Genetic Algorithms

Genetic algorithms (GAs) are search procedures based on the evolutionary process in nature. They differ from other approaches in that they use probabilistic and not deterministic procedures for progressing the search. The idea is that the GA operates a population of individuals, each individual representing a potential solution to the problem, and applies the principle of survival of the fittest on the population, so that the individuals evolve towards better solutions to the problem.

The individuals are given a chromosoidal representation, which corresponds to the genotype of an individual in nature. Three operations can be performed on individuals in the population, *selection*, *crossover* and *mutation*. These correspond to the selection of individuals in nature for breeding, where the fitter members of a population breed and so pass-on their genetic material. The crossover corresponds to the combination of genes by mating, and mutation to genetic mutation in nature. The selection is weighted so that the 'fittest' individuals are more likely to be selected for crossover, the fitness being a function of the function which is being minimised. By means of these operations, the population will evolve towards a solution.

Most GAs have been used for single objective problems, although several multiobjective schemes have been proposed (*e.g.* Schaffer, 1985; Wienke *et al.*, 1992). In particular, Patton *et al.* (1994) have used a GA to solve the MoI by converting the problem to a minimax optimisation problem. Fonseca and Fleming (1995) have used an approach called the multiobjective genetic algorithm (MOGA), which is an extension on an idea by Goldberg (1989). This formulation maintains the genuine multiobjective nature of the problem, and is essentially the scheme used here. Further details of the MOGA can be found in Fonseca and Fleming (1993a,b, 1994).

The design philosophy of the MOGA differs from the MoI, in that a set of simultaneous solutions is sought, and the designer then selects the best solution from the set. The idea behind the MOGA is to develop a population of Pareto-optimal or near Pareto-optimal solutions. However, to restrict the size of the near Pareto-optimal set and to give a more practical setting to the MOGA, the problem has been formulated in a similar way to the MoI in Fonseca and Fleming (1993a). This formulation maintains the genuine multiobjective nature of the problem. The aim is to find a set of solutions which are non-dominated and which satisfy a set of inequalities. An individual j with a set of objective functions $\phi^j = (\phi_1^j, \ldots, \phi_n^j)$ is said to be *non-dominated* if for a population of N individuals, there are no other individuals $k = 1, \ldots, N, k \neq j$ such that

$$\begin{array}{lll} \text{a)} & \phi_i^k \leq \phi_i^j & \text{for all } i = 1, \ldots, n \quad \text{and} \\ \text{b)} & \phi_i^k < \phi_i^j & \text{for at least one } i. \end{array}$$

The MOGA is set into a multiobjective context by means of the fitness function. The individuals are ranked on the basis of the number of other individ-

uals they are dominated by for the unsatisfied inequalities. Each individual is then assigned a fitness according to their rank. The mechanism is described in detail in Fonseca and Fleming (1993a). An alternative, less computationally demanding, ranking scheme has been proposed by Liu *et al.* (1994).

To summarise, the problem addressed by the MOGA could be stated as: find a set of M admissible points $p^j, j = 1, \dots M$ such that

$$\phi_i^j \leq \varepsilon_i, \quad j = 1, \dots M, \ i = 1, \dots, n \tag{1.1.16}$$

and such that $\phi^j (j = 1, \dots M)$ are non-dominated.

Genetic algorithms are naturally parallel and hence lend themselves well to multiobjective settings. They also work well on non-smooth objective functions. In addition, it is very easy to extend GAs to solve mixed continuous/integer problems. Thus the GA can be used to search over controller structures as well as over the controller parameters. For more information, see Dakev *et al.* (1997), Chipperfield *et al.* (1999) and Tang *et al.* (1996).

1.2 Multiobjective Control

Most control design techniques have only paid attention to optimal solutions for one special performance index, *e.g.*, H^∞ norm on a closed-loop system transfer function, the eigenvalue sensitivity function or the linear quadratic index. However, many practical control systems are required to have the ability to fit simultaneously different and often conflicting performance objectives as best as possible, for instance, closed-loop stability, low feedback gains and insensitivity to model parameter variations. A number of traditional approaches to control system design objectives make use of scalar summation of all weighted objectives in one cost function. Though this method simplifies the approach to optimisation, it is not clear how each objective is affected by the controller. On the other hand, if all of the objectives are considered through the use of the individual cost functions, then the action of each objective on the structure of the controller can be determined.

During the last two decades, multiobjective control has been considered in the design process. The control system objectives are described by a set of performance indices. This type of control problem appears in flight control design, in the control of space structures and in industrial process control. The concept of a generic multiobjective control problem which involves different types of performance indices is a very interesting idea. For example, the multiple objectives can be considered as different types of norms on transfer functions. In this case, one may take H^2 norms on some of the closed-loop transfer functions, H^∞ norms on others, and L^1 norms on some others. Such a multiple objective problem would naturally arise in considering trade-offs between nominal performance and robust stability.

Although there has been considerably more research into multiobjective decision making in the field of systems engineering than into multiobjective

optimisation design of multivariable control, more and more research in multiobjective control has been carried out. The main research areas are multiobjective robust control (Whidborne *et al.*, 1995a, 1996b), multiobjective critical control (Liu *et al.*, 1995), multiobjective eigenstructure assignment (Liu and Patton, 1996a), mulitobjective PID control (Whidborne *et al.*, 1995b; Liu *et al.*, 1998; Liu and Daley, 2001), multiobjective identification (Liu and Kadirkamanathan, 1999), multiobjective fault detection (Chen *et al.*, 1996), and multiobjective linear quadratic Gaussian control (Tabak *et al.*, 1979; Skelton and DeLorenzo, 1985; Koussoulas and Leondes, 1986; Toivonen and Makila, 1989; Khargonekar and Rotea, 1991).

1.2.1 Conflicts and Trade-offs in Control Systems

In general, engineering design also consists of obtaining the right balance between conflicting cost, design and performance requirements. This idea also extends to control systems design. That there are trade-offs in control systems is well-known and usually taught at an undergraduate level. For example, consider a simple plant, which could be a model of a servo system

$$G(s) = \frac{1}{s(s+a)} \tag{1.2.1}$$

with a proportional controller k with negative feedback. The closed loop system is

$$T(s) = \frac{k}{s^2 + as + k} \tag{1.2.2}$$

and the maximum overshoot to a step response is given by

$$M_p = e^{-a\pi/\sqrt{4k-a^2}} \tag{1.2.3}$$

and the time-to-peak by

$$t_p = \frac{2\pi}{\sqrt{4k-a^2}} \tag{1.2.4}$$

for $k > (a^2/4)$. Assuming that $a = 1$, Figure 1.2 shows the overshoot and the time-to-peak plotted against k. Immediately, it can be seen that there is a trade-off between M_p and t_p. Figure 1.3 shows the overshoot/time-to-peak trade-off curve.

Another well-known trade-off often taught on graduate level courses is the trade-off between the sensitivity and complimentary sensitivity functions (see, for example, Maciejowski, 1989, pp 10-22). Consider the control system shown in Figure 1.4. In this general standard form, the control problem can include consideration of the effects of reference input, plant disturbances and measurement noise. The plant is assumed fixed and known, and represented by the strictly proper transfer function $G(s)$, with a disturbance signal $D(s)$. The plant has an input, also known as the control, $U(s)$, and an output

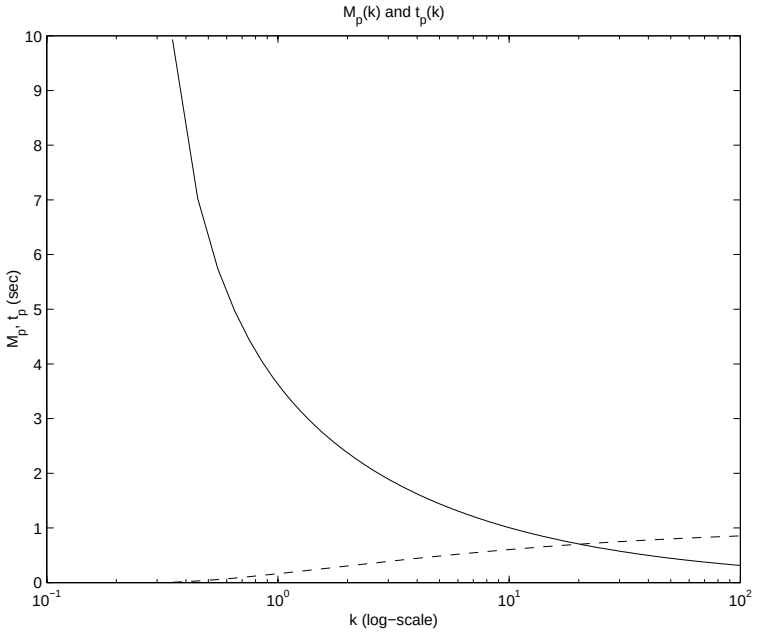

Figure 1.2: Maximum overshoot M_p (- - -) and time-to-peak t_p (—) against k

$Y(s)$. The controller is fixed and known with a proper transfer function $K(s)$. The output is subjected to measurement errors $M(s)$, and the system has a reference signal $R(s)$ which is to be followed by the plant output.

From Figure 1.4,

$$Y(s) = D(s) + G(s)K(s)\left[R(s) - M(s) - Y(s)\right] \tag{1.2.5}$$

$$\left[1 + G(s)K(s)\right]Y(s) = D(s) + G(s)K(s)\left[R(s) - M(s)\right] \tag{1.2.6}$$

The sensitivity function $S(s)$ is defined as

$$S(s) = \left[1 + G(s)K(s)\right]^{-1} \tag{1.2.7}$$

and the closed-loop transfer function $T(s)$ is defined as

$$T(s) = S(s)G(s)K(s) \tag{1.2.8}$$

so

$$Y(s) = S(s)D(s) + T(s)\left[R(s) - M(s)\right] \tag{1.2.9}$$

The error $e(t)$ is defined as

$$e(t) = r(t) - y(t) \tag{1.2.10}$$

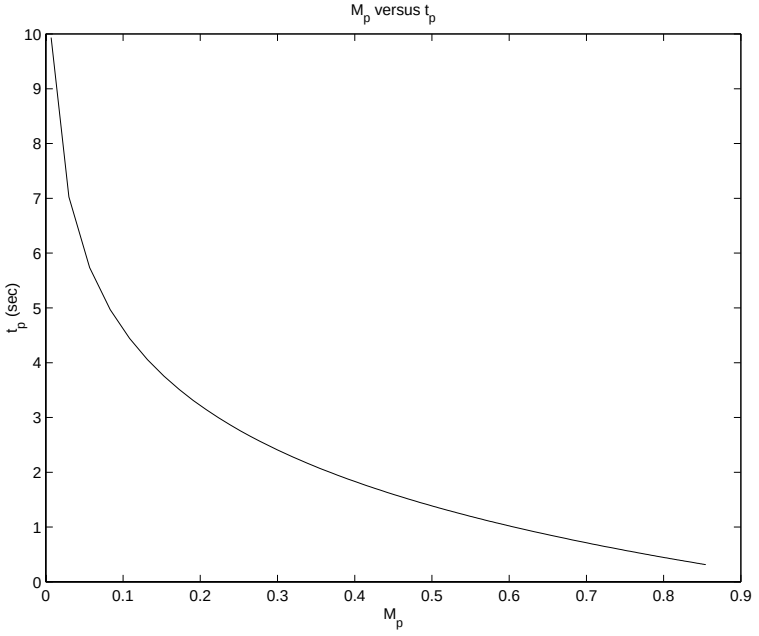

Figure 1.3: Trade-off showing M_p against t_p

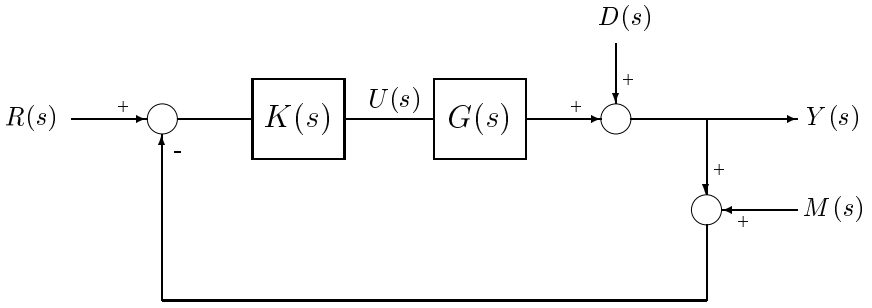

Figure 1.4: Standard Control System Configuration

Thus,

$$\begin{aligned} E(s) &= R(s) - S(s)D(s) - T(s)\left[R(s) - M(s)\right] \\ &= \left[1 - T(s)\right]R(s) - S(s)D(s) + T(s)M(s) \end{aligned} \tag{1.2.11}$$

but from (1.2.7) and (1.2.8)

$$T(s) + S(s) = 1 \tag{1.2.12}$$

so

$$E(s) = S(s)\,[R(s) - D(s)] + T(s)M(s) \tag{1.2.13}$$

Thus, in order to reduce the error, both $S(s)$ and $T(s)$ need to be made small in some sense. However, from (1.2.12) it can be seen that if $S(s)$ is made nearly zero, $T(s)$ becomes nearly unity; and conversely, if $T(s)$ is nearly zero, $S(s)$ is nearly unity. There is thus an unavoidable trade-off between attenuating plant disturbances, and filtering out measurement error. This also extends to a trade-off between system performance and stability robustness to multiplicative plant perturbation (Doyle *et al.*, 1991, pp 46-62).

Other unavoidable trade-offs can be shown to exist for control systems, such as the so-called water-bed effects resulting from right-half-plane poles and zeros, whereby sensitivity reduction at some frequencies is traded-off for sensitivity increases at other frequencies. See Skogestad and Postlethwaite (1986, pp 165-170) for further details.

1.2.2 Multiobjective Robust Control

In general, it is almost impossible to get an absolutely accurate model of a practical system plant. Some assumptions and simplifications are made about the model and few (if any) real systems are linear. During the operation of a control system, the plant dynamics change with operating conditions over time, for example, parts wear out, ambient temperature changes, and the load change. Therefore, to design control systems which can handle this uncertainty, robust control which can maintain stability and performance in the face of uncertainty is needed.

Traditionally, designers use design methods based on either analytical optimisation or parameter optimisation for robust control system design. Both methods have advantages and disadvantages. Briefly, analytical optimisation techniques (*e.g.* H_∞, LQG) generally are robustly stable, provide a global optimum, and can deal with relatively large multivariable problems; but they are single-objective and not very flexible, have non-explicit closed-loop performance, provide high-order controllers. On the other hand, parameter optimisation based methods (*e.g.* MoI) generally are often multiobjective and flexible, have explicit closed-loop performance and provide simple controllers; however, they are often non-convex resulting in local minima, not implicitly robustly stable, can deal with small problems only, and may have difficulty stabilising the system. A combination of analytical optimisation and parameter search methods, called multiobjective robust control by mixed optimisation, has been employed to overcome some of the limitations of using just one approach (Whidborne, *et al.*, 1995b).

The multiobjective robust control design by mixed optimisation has been applied to the loop-shaping design procedure (LSDP), which is based on H_∞ robust stabilisation combined with classical loop-shaping. The procedure is essentially a two-stage design process. First, the open-loop plant is augmented by pre- and post-plant weighting functions to give a desired shape to the singular values of the open-loop frequency response. Then the resulting shaped plant is robustly stabilised using H_∞ optimisation of a normalised left coprime factorisation description of the plant. The designer thus chooses the optimisation technique and the structure of the weighting functions. He/she defines suitable performance functions (*e.g.* rise time, settling time, bandwidth, system norms, etc.) along with the design goals, and the multiobjective optimisation methods can be used to search for suitable values of the weighting function parameters such that is satisfied. Details of the above procedure may be found in Whidborne *et al.* (1994a, 1995a) and Chipperfield *et al.* (1999).

Details of the mixed-optimisation procedure applied to a 2 degree-of-freedom LSDP (Hoyle *et al.*, 1991; Limebeer *et al.*, 1993) may be found in Murad *et al.* (1993) and Whidborne *et al.* (1995b) and to a mixed sensitivity H_∞ approach and an LQG optimisation in Whidborne *et al.* (1995a). An alternative 2 degree-of-freedom LSDP approach with a parameterised precompensator is proposed in Whidborne *et al.* (1994b). Other authors have suggested using a mixed optimisation approach. The first appears to be Baras *et al.* (1984), who suggested using nonlinear optimisation for designing the weights for LQG control design. A similar approach has been independently suggested by Paddison *et al.* (1994), Haessig (1995) and Tych (1994). Use of genetic algorithms to design the weighting functions for the LSDP has also been suggested by White *et al.* (1995). Some efficient methods for mixed optimisation with the LSDP have been proposed by Pantas (1998).

1.2.3 Multiobjective Critical Control

A problem that occurs frequently in control engineering is to control outputs (usually the errors) of a system subjected to random external inputs (reference inputs and/or disturbances) so that the absolute value of the outputs is within prescribed bounds. Any violation of the bounds results in unacceptable, perhaps catastrophic, operation. Such a system is said to be critical (Zakian, 1989; Whidborne and Liu, 1993). Usually, a control system is probably subjected to two kinds of uncertainties. One is the uncertainty in external inputs which impinge on the system, called the external uncertainty. The other is the uncertainty in the mathematical models used to represent the process, called the internal uncertainty. Though the external uncertainty in critical systems has been considered in many papers, the internal uncertainty in these systems has been paid little attention. Therefore, the robust control, which refers to the maintenance of design specification in the presence of uncertainties, of critical systems with external and internal uncertainties is a very interesting

and important subject.

The robust control design of multivariable critical systems with external and internal uncertainties has been considered in Liu *et al.* (1996). The design problem formulated as a set of inequalities includes the output performance criteria in the time domain and the robust performance criterion in the frequency domain. Some relationships between an input space, an unmodelling error space, a controller, output performance and robust performance are established for SISO and MIMO cases so that the design problem is largely simplified and can be solved in the frequency domain using multiobjective optimisation techniques.

1.2.4 Multiobjective Eigenstructure Assignment

Eigenstructure assignment is a design technique which may be used to assign the entire eigenstructure (eigenvalues, and right or left eigenvectors) of a closed-loop linear system via a feedback control law. It has been found that degrees of freedom are available over and above eigenvalue assignment using feedback control for linear time-invariant multi-input multi-output (MIMO) systems. Many methods and algorithms have been developed to exercise those degrees of freedom to give the systems some good performance characteristics (Kautsky *et al.*, 1985; Liu and Patton, 1998a).

Most eigenstructure assignment techniques have only paid attention to optimal solutions for one special performance index, *e.g.*, the eigenvalue sensitivity function or the linear quadratic index. However, a number of practical control systems are required to have the ability to fit simultaneously different and often conflicting performance objectives as best as possible, for instance, closed-loop stability, low feedback gains and insensitivity to model parameter variations. A multiobjective control system design using eigenstructure assignment has been presented in Liu and Patton (1996a). The multiobjective performance indices include the individual eigenvalue sensitivities, low feedback gains, and the sensitivity functions in the frequency domain.

In addition, the multi-criteria optimisation design for multivariable control systems using eigenstructure assignment and the method of inequalities (Patton *et al.*, 1994) has been studied. This uses a set of performance inequalities to describe the multiple objectives which include closed-loop stability, low feedback gains and insensitivity to model parameter variations. The controller designed using eigenstructure assignment and the method of inequalities has great potential for yielding the best eigenstructure and insensitivity to perturbations of the system parameters.

1.2.5 Multiobjective PID Control

It is well known that the PID (proportional integral derivative) controller is the most popular approach for industrial process control and many design techniques have been developed (see, for example, Ziegler and Nichols, 1942;

Astrom and Hagglund, 1984; Hang *et al.*, 1991; Zhuang and Atherton, 1993; McCormack and Godfrey, 1998; Daley and Liu, 1999). Although many PID design methods have been developed to give simple tuning rules for the controller parameters using either one or two measurement points of the system frequency-response, their control performance may not satisfy the desired requirements.

In the frequency domain, there are two quantities used to measure the stability margin of the system. One is the gain margin, which is the factor by which the gain is less than the neutral stability value. The other is the phase margin, which is the amount by which the phase of the system exceeds when the system gain is unity. The gain and phase margins are also related to the damping of a system. In addition to the stability of a design, the system is also expected to meet a speed-of-response specification like bandwidth. The crossover frequency, which is the frequency at which the gain is unity, would be a good measurement in the frequency domain for the system's speed of time response. Also, the larger the value of the magnitude on the low-frequency asymptote, the lower the steady-state errors will be for the closed-loop system. This relationship is very useful in the design of suitable compensation. After the above considerations, a multiobjective PID control scheme in the frequency domain has been proposed by Liu and Daley (1999). This scheme considers four major requirements in the frequency domain, which are: the gain-margin, the phase-margin, the crossover frequency and the steady-state error. It was successfully applied to a rotary hydraulic system.

A multi-input multi-output (MIMO) PI (proportional integral) controller design approach using multiobjective optimisation is proposed. It has been used for controller design of a gasifier (Liu *et al.*, 2000). The design aim is to satisfy a set of specifications on the system outputs, inputs and input rates when a step disturbance is applied to the system. The parameters of the MIMO PI controller are optimally designed to satisfy a set of multiobjective performance criteria which are formulated from the system specifications.

1.2.6 Multiobjective Optimisation of Controller Implementations

The process of modern digital controller design is generally presented in the literature as taking one of two paths. Either the design is performed in the continuous time domain, and the continuous time controller is subsequently discretised. Alternatively, the plant is discretised and the design performed completely in the discrete time domain. Both paths produce a design as some state-space or transfer function representation of the controller. However, it is generally not sufficient to implement the controller as a simple difference equation without a consideration of the relevant arithmetic that will be used (floating or fixed-point), the wordlength and the controller realisation (difference equation, lattice filter, state-space, δ-transform etc.). This final part of

the design, the implementation stage, is frequently ignored or considered in a purely ad hoc manner.

There are a number of critical issues that need to be considered in deciding on the digital controller implementation. Almost all digital controller implementations will require some rounding of the signals and parameters. Rounding errors cause noise to be introduced into the system, and finite precision representation of the controller parameters causes a reduction in the relative stability and closed loop performance of the system. These problems become more marked as (i) the sampling rate increases, and (ii) the controller complexity/order increases. The question then arises of how to arrange the controller structure (or parameterisation) so as to minimise the effects of the finite precision, whilst simultaneously minimising the required word-length, memory requirements and speed of computation as well as minimising quantisation noise and ensuring that the variables are suitably scaled so as to prevent overflow (and underflow for floating point arithmetic). Sampling time and inter-sample effect are also important. The implementation stage is of course extremely important for the successful application of advanced control designs which are often high order.

Digital control is now the main platform for control implementation in almost all application areas, and since almost all digital computing devices have finite precision due to the finite word-length, the issue of finite precision in the controller implementation is important. Clearly, the precision of the computing device can be increased by increasing the word-length or amended by use of logarithmic or exponent scaling schemes typical in floating point architectures. However, there is a cost involved in the use of more complex architectures, in terms of increased financial cost in hardware and memory, as well as in terms of chip design, software costs, lower speed, and lower reliability. This may be of little disadvantage in slow-dynamic, high capital systems, such as those typically found in process control applications, but for systems which are mass-produced, very high speed, safety critical or power critical, reducing the cost and complexity of the computing device is a critical consideration. Thus the question of how to satisfactorily implement the controller with minimal cost is a concern, and since there is an obvious cost/performance trade-off, this problem can be considered as one of multiobjective optimisation. For an overview of relevant digital controller implementation issues, see Istepanian and Whidborne (2001).

1.2.7 Multiobjective Nonlinear Identification

For nonlinear system identification using the approximation approach, two key questions are important. One is how to judge the accuracy for the nonlinear function being approximated and other is how to choose nonlinear function units to guarantee the accuracy. Many of nonlinear system identification approaches fix the number of nonlinear function units and use only a single

performance function, *e.g.*, L_2-norm of the difference between the real nonlinear system and the nonlinear model which results in the well-known least squares algorithm, to measure and judge the accuracy of the identification model and to optimise the approximation. The assumption behind choosing the L_2-norm is that the noise in the process and measurements has Gaussian (normal) distributions.

However, in nonlinear system identification there are often a number of objectives to be considered. The objectives are often conflicting and no identification which can be considered best with respect to all objectives exists. Hence, there is an inevitable trade-off between objectives, for example, the distance measurement and maximum difference measurement between the real nonlinear system and the nonlinear model. Model comparison methods, such as Information Criterion (Akaike, 1974), *Bayesian model selection* (MacKay, 1992) and *Minimum Description Length* (MDL) (Rissanen, 1989), consider two such objectives, namely, Euclidean distance (L_2-norm) and model complexity. These procedures allow the selection of the best amongst a small number of candidate models (MacKay, 1992). In addition to the above two objectives, the L_∞-norm of the difference between the real nonlinear system and the nonlinear model is considered (Liu and Kadirkamanathan, 1999). This is because this performance function represents the accuracy bound of the approximation achieved by the estimated model.

1.2.8 Multiobjective Fault Detection

Fault diagnosis has become an important subject in modern process automation as it provides the pre-requisites for fault tolerance, reliability or security, which constitute fundamental design features in any complex engineering system. The general procedure of fault diagnosis in dynamical systems with the aid of analytical redundancy consists of the generation of so-called residuals and the decision on the occurrence of a fault and isolation of the faulty element. In order to make the residual insensitive to modelling uncertainty and sensitive to sensor faults, a number of performance indices have been defined to achieve good fault diagnosis performance. Some of performance indices are defined in the frequency domain to account for the fact that the effects of modelling uncertainty and faults occupy different frequency ranges. Robust fault detection in the frequency domain has been attracting enormous attention.

In order to diagnose incipient faults, the design of optimal residuals based on multiobjective optimisation is discussed in Chen *et al.* (1996). The residual is generated via an observer. To reduce false and missed alarm rates in fault diagnosis, a number of performance indices are introduced into the observer design. These performance indices are expressed in the frequency domain to take account of the frequency distributions of faults, noise and modelling uncertainties. All objectives then are reformulated into a set of

inequality constraints on performance indices. A multiobjective fault diagnosis algorithm is thus used to search for an optimal solution to satisfy all the objectives expressed by a set of inequalities. This algorithm is applied to fault diagnosis of a flight control system example and shows that incipient sensor faults can be detected reliably in the presence of modelling uncertainty.

1.3 Outline of the Book

This monograph will show fundamental theory, a number of design methods and algorithms, and some applications in multiobjective optimisation and control, which cover main recent research work on this subject. Its contents are arranged as follows.

Chapter 1 gives an overview about multiobjective optimisation and control, and the outline of the monograph.

Chapter 2 introduces basic concepts and main methods for unconstrained nonlinear optimisation. Many nonlinear optimisation methods are based on one-dimensional search. Typical approaches in one-dimensional search will be introduced in this chapter, including the dichotomy method, the Fibonacci method and the golden section search method. The necessary and sufficient conditions for local optimality will be presented. Several unconstrained optimisation methods will be discussed, for example the steepest decent method, the Newton's method and the quasi-Newton's methods. The purposes of this chapter is to introduce the main ideas and concepts in nonlinear optimisation so that the reader is equipped for the following chapters.

Chapter 3 presents the optimality conditions and typical methods for constrained optimisation. Both necessary and sufficient conditions will be discussed, including Kuhn-Tucker necessary conditions and the saddle point (sufficient) condition. Two main classes of methods will be introduced, *e.g.*, primal methods and dual methods. Among the primal methods are the sequential linear programming and sequential quadratic programming. The dual methods include the exterior penalty method, the interior penalty method and the Lagrangean method. These methods cover the wide spectrum of nonlinear optimisation methods and provide a basis for multiobjective optimisation and control.

Chapter 4 discusses the concepts, the definitions and the method classification in multiobjective optimisation. Two main classes of multiobjective methods will be described: methods based on L_p - norm, where p is a positive integer, and interactive methods. The first class includes goal programming, minimax solution scheme, goal attainment method and the reference point method. Among the typical methods of the second class are STEM method, Geoffrion's method, the ISTM method and the gradient projection method. These different methods provide various ways of acquiring and utilising preference information from the decision makers (control system designers) for achieving the most preferred solutions. They provide a powerful means for

multiobjective optimisation and control system design. Examples will be provided to demonstrate the solution procedures of these methods.

Chapter 5 introduces genetic algorithms (GAs) for optimisation. The genetic algorithm is a stochastic global search method for optimisation that mimics the metaphor of natural biological evolution. Applying the principle of survival of the fittest to produce better and better approximations to a solution, GAs operate on a population of potential solutions. A new set of approximations at each generation is created by the process of selecting individuals, which actually are chromosomes in GAs, according to their fitness level in the problem domain and breeding them together using operators borrowed from natural genetics, for example, crossover and mutation. This process results in the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation. Application of GAs to multiobjective optimisation is discussed.

In Chapter 6, specifications for control system designs frequently include stringent closed-loop performance requirements, which are required to be met in the face of uncertainties. These requirements are usually conflicting, the problem is thus multiobjective as a number of conflicting objectives need to be simultaneously obtained. Such problems can be effectively solved by a combination of analytical optimisation techniques (*e.g.*, such as H^∞), from which robustness is obtained, and multiobjective optimisation techniques (*e.g.*, the method of inequalities), which are designed for explicit closed-loop performance. This chapter describes this approach and applies it to the design of an unknown plant.

Chapter 7 considers robust control design of critical systems with external and internal uncertainties using multiobjective optimisation methods. The formulation of the robust control design of these systems is expressed by a set of performance criteria which includes output performance criteria in the time domain and robust performance criterion in the frequency domain of the system. Some relationships between an input space, a modelling error space, a controller, output performance and robust performance are established for both single-input single-output and multi-input multi-output critical systems so that the robust control design problem of these systems is largely simplified.

Chapter 8 discusses multiobjective robust control design for multivariable systems using eigenstructure assignment. It covers various performance indices (or cost functions) in the objectives, which are individual eigenvalue sensitivity functions, and the sensitivity and the complementary sensitivity functions in the frequency domain. Based on these performance indices, the robustness criteria are expressed by a set of inequalities. It makes full use of the freedom provided by eigenstructure assignment to find a controller to satisfy the robustness criteria.

Chapter 9 is concerned with PI controller design using multiobjective optimisation which addresses the ALSTOM benchmark challenge on gasifier con-

trol. The nonlinear gasifier has been linearised about three operating points. The design aim is to satisfy a set of specifications on the system outputs, inputs and input rates when a step disturbance is applied to the system. The parameters of the multi-input multi-output (MIMO) PI controller are selected to satisfy a set of multiobjective performance criteria, which are formulated from the system specifications. Simulation results demonstrate the performance of the controller at the three operating points.

Chapter 10 applies a multiobjective genetic algorithm based method to determine optimal FWL structures for PID digital controllers. The method is illustrated by practical examples. The method exploits the fact that the implementation of FWL controllers is by means of binary numbers, as is the representation in genetic algorithms. The method requires the solution of a linear system similarity completion problem. A solution to the linear system similarity completion problem for 2 state SISO systems has been presented. This solution would need to be extended to general linear systems for the methodology to be extended to higher order controllers. Otherwise, the method is entirely generic, and any computable set of stability and performance measures could be included.

Chapter 11 presents an approach to model selection and identification of nonlinear systems via neural networks, based on multiobjective performance criteria. It considers three performance indices (or cost functions) as the objectives, which are the Euclidean distance (L_2-norm) and maximum difference (L_∞-norm) measurements between the real nonlinear system and the nonlinear model, and the complexity measurement of the nonlinear model. An algorithm based on the method of inequalities, least squares and genetic algorithms is developed for optimising over the multiobjective criteria. Genetic algorithms are also used for model selection in which the structure of the neural networks are determined. The Gaussian radial basis function network is applied to the identification of a liquid level nonlinear system.

Chapter 12 focuses on design of optimal residuals in order to diagnose incipient faults using multiobjective optimisation. The residual is generated via an observer. To reduce false and missed alarm rates in fault diagnosis, a number of performance indices are introduced into the observer design. Some performance indices are expressed in the frequency domain to take account of the frequency distributions of faults, noise and modelling uncertainties. All objectives are then reformulated into a set of constraints on performance indices. Multiobjective optimisation methods are thus used to search for an optimal solution to satisfy these constraints. The design approach is applied to a flight control system which shows that incipient sensor faults can be detected reliably in the presence of modelling uncertainty.

Finally, the contributions and responsibilities of the authors are as follows: Chapter 1 – G. P. Liu and J. F. Whidborne, Chapters 2, 3 and 4 – J. B. Yang, Chapters 6 and 10 – J. F. Whidborne, and Chapters 5, 7, 8, 9, 11 and 12 – G. P. Liu.

Chapter 2

Nonlinear Optimisation

In this chapter, basic concepts and main methods for unconstrained nonlinear optimisation are discussed. Many nonlinear optimisation methods are based on one-dimensional search. Typical one-dimensional search methods will be introduced first, including the dichotomy method, the Fibonacci method and the golden section search method. The necessary and sufficient conditions for local optimality will be presented. Several unconstrained optimisation methods will be discussed, including the steepest decent method, Newton's method and the quasi-Newton's methods. The purpose of this chapter is not to discuss a complete range of optimisation methods but to introduce the main ideas and concepts in unconstrained nonlinear optimisation so that the reader is well equipped for reading the following chapters.

2.1 One-Dimensional Optimisation

2.1.1 The Dichotomy Method with Derivatives

Suppose $f(x)$ is a continuously differentiable function and has a single minimum solution in an interval $[x_{min}, x_{max}]$, that is $f^{'}(x_{min}) \leq 0$ and $f^{'}(x_{max}) \geq 0$. Then, there is one point x^* at which the derivative of $f(x)$ vanishes and $f(x)$ is minimised. It is required to find the point x^*.

The dichotomy method consists of finding a first interval $[x_{min},\ x_{max}]$, where the minimum solution lies, and then of reducing progressively this interval by bisections until we find a final interval of width $< \varepsilon$, for a sufficiently small ε.

More precisely, at a given step we compute $f^{'}(x_1)$ at the middle point x_1 of the interval, or

$$x_1 = \frac{1}{2}[x_{min} + x_{max}] \tag{2.1.1}$$

At x_1, there are three possible cases: $f^{'}(x_1) > 0$, $f^{'}(x_1) < 0$ and $f^{'}(x_1) = 0$.

The first case is shown in Figure 2.1. In this case, the minimum solution must lie between x_{min} and x_1, and the interval $[x_1 \ x_{max}]$ can be eliminated.

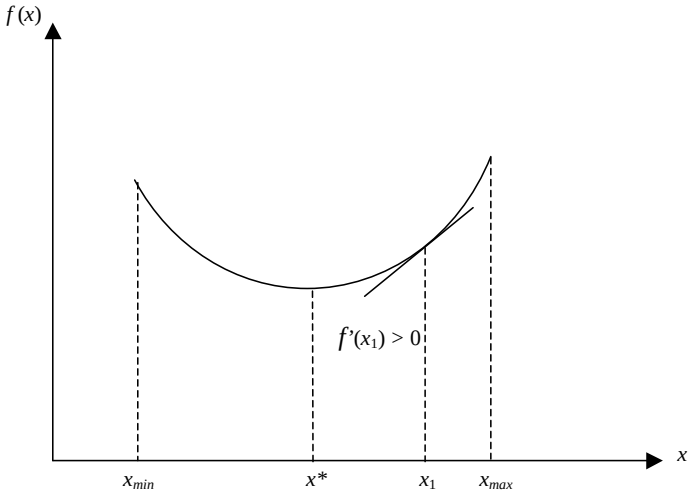

Figure 2.1: $f'(x_1) > 0$

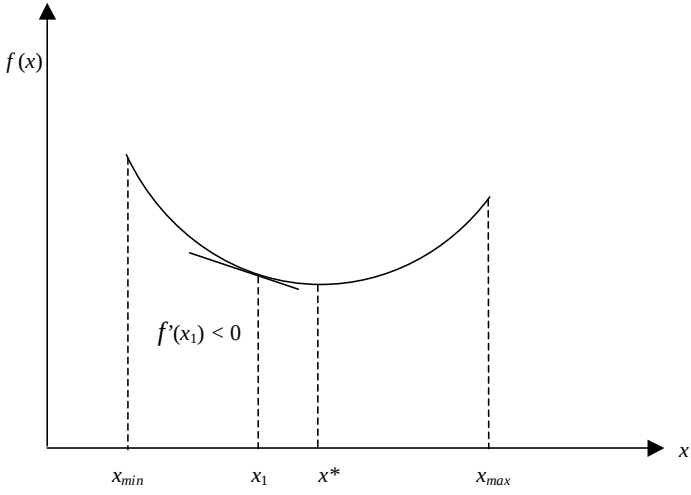

Figure 2.2: $f'(x_1) < 0$

The second case is as illustrated in Figure 2.2. In this case, the minimum solution must lie between x_1 and x_{max}, and the interval $[x_{min}\ x_1]$ can be eliminated.

In the third case where $f'(x_1) = 0$, x_1 is already the minimum solution. The above process can be summarised as follows.

If $f'(x_1) > 0$, then we replace x_{max} by x_1 and repeat the process.

If $f'(x_1) < 0$, then we replace x_{min} by x_1 and repeat the process.

Since the length of the interval is halved at each step, the dichotomy method converges linearly with a rate of convergence equal to $\frac{1}{2}$ (Minoux, 1986).

2.1.2 The Dichotomy Method without Derivatives

This method is very similar to the method discussed in section 2.1.1. However, it does not require information about derivatives.

At each step, this method computes the function $f(x)$ in two new points in order to halve the length of the interval that contains the optimal solution. Suppose the initial interval is $[a^o, b^o]$. We divide the interval into four equal sub-intervals by first computing the middle point of the interval $c^0 = (a^o + b^0)/2$, and then two points $d^o = (a^o + c^0)/2$ and $e^0 = (c^0 + b^0)/2$. The length of the sub-intervals is $\delta^0 = (b^0 - a^o)/4$.

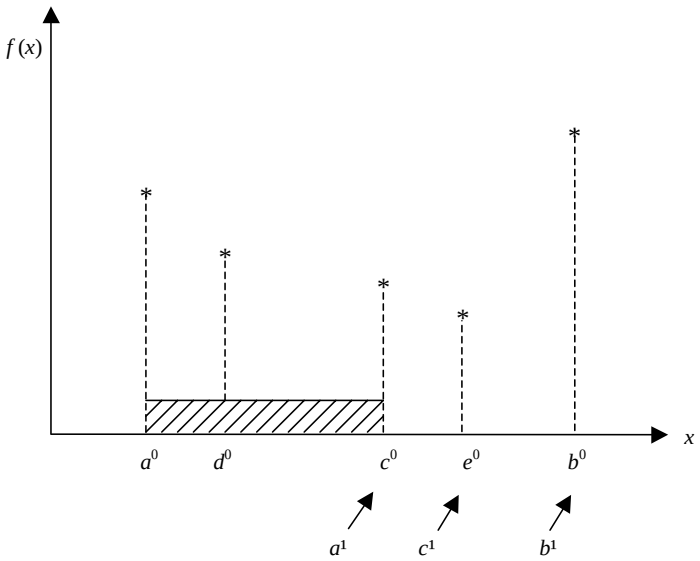

Figure 2.3: Elimination of sub-interval: Case 1

Suppose the function $f(x)$ only has one optimal solution. It can then be shown that two of the four sub-intervals can always be eliminated because

the optimum cannot be within them and that the optimum is within two adjacent sub-intervals.

In Figure 2.3, it is clear that the two sub-intervals [a^o, d^o] and [d^o, c^0] can be eliminated and the optimum is within the two adjacent sub-intervals [c^0, e^0] and [e^0, b^0] (Minoux, 1986). Let $a^1 = c^0$, $b^1 = b^0$ and $c^1 = e^0$. We then obtain the same problem on the interval [a^1, b^1] of half the length [a^o, b^0].

In Figure 2.4, one can see that the two sub-intervals [a^o, d^o] and [e^0, b^0] can be eliminated and the optimum is within the two adjacent sub-intervals [d^o, c^0] and [c^0, e^0] (Minoux, 1986). Let $a^1 = d^o$, $b^1 = e^0$ and $c^1 = c^0$. We then obtain the same problem on the interval [a^1, b^1] of half the length of [a^o, b^0].

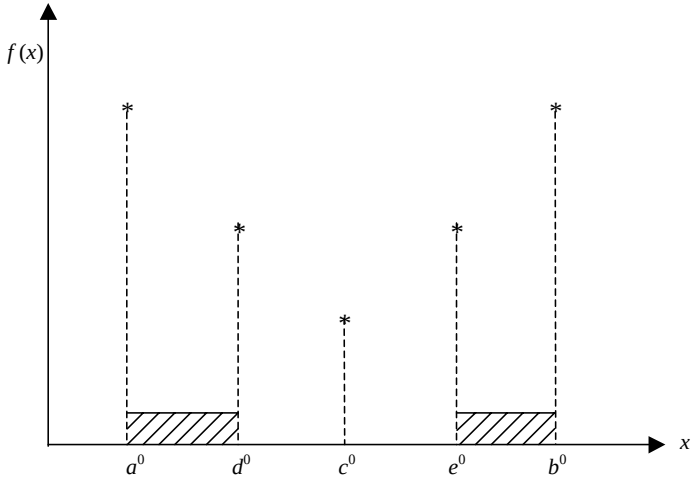

Figure 2.4: Elimination of sub-interval: Case 2

To further divide the interval [a^1, b^1], only two further calculations of the function $f(x)$ are required at $d^1 = (a^1 + c^1)/2$ and $e^1 = (c^1 + b^1)/2$. This will lead to the further reduction of the interval [e^0, b^0]. The same process is repeated until the interval is reduced to be small enough. Table 2.1 shows the rate of reduction of the interval length as a function of calculations of the function $f(x)$ (Minoux, 1986). Note that 5 calculations are required at the first stage and only 2 calculations are needed at other stages.

2.1.3 The Fibonacci Method

This method makes use of the Fibonacci sequence and is an optimal method in the sense that for a given number of calculations on a function $f(x)$ it leads to the smallest possible reduced interval (Minoux, 1986). In other words, the method can reduce a largest possible sub-interval of the current interval.

Table 2.1: Rate of convergence

n	$\frac{b^n - a^n}{b^0 - a^0} = \frac{1}{2^{(n-3)/2}}$
5	0.5
13	10^{-1}
17	10^{-2}
23	10^{-3}
29	10^{-4}
42	10^{-6}

Suppose the original interval is $[a^1, b^1]$ and the values of the function $f(x)$ are known. Suppose there are two intermediate points c^1 and d^1 in the interval with $b^1 > c^1 > d^1 > a^1$. If $f(x)$ only has one minimum in the interval, it is then possible to eliminate a sub-interval, either $[c^1, b^1]$ as shown in Figure 2.5 or $[a^1, d^1]$ as shown in Figure 2.6 (Minoux, 1986).

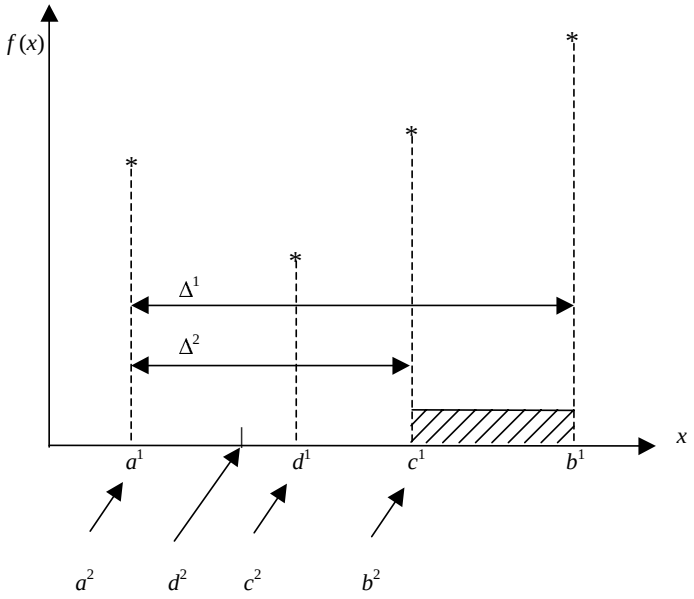

Figure 2.5: Elimination of sub-interval: Case 3

As shown in Figure 2.5, the length of the original interval is given by $\Delta^1 = b^1 - a^1$. Let Δ^2 be the length of a sub-interval that is not eliminated. If the sub-interval $[c^1, b^1]$ is eliminated, the length of the remaining sub-interval is $c^1 - a^1$. If the sub-interval $[a^1, d^1]$ is eliminated, then the length of the remaining sub-interval is $b^1 - d^1$. The length Δ^2 of the remaining sub-interval

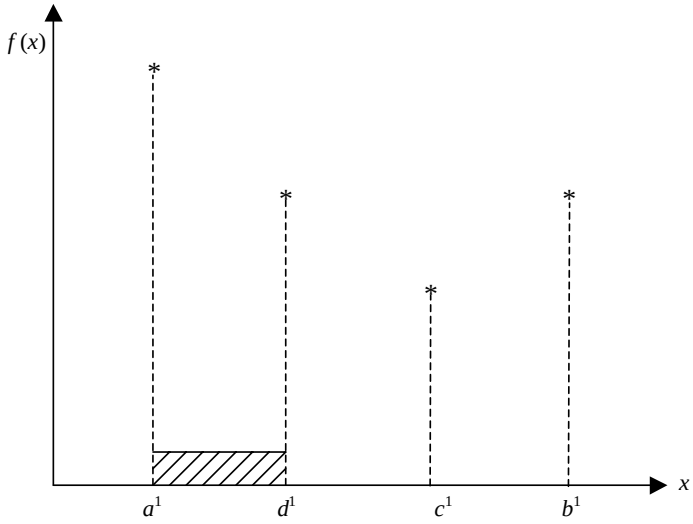

Figure 2.6: Elimination of sub-interval: Case 4

is required to be independent of the result of the trial, or independent of the function $f(x)$. Then there must be

$$c^1 - a^1 = \Delta^2 \tag{2.1.2}$$

$$b^1 - d^1 = \Delta^2 \tag{2.1.3}$$

which means $b^1 - c^1 = d^1 - a^1$. In other words, the two points c^1 and d^1 are symmetrical with regard to the centre of the interval $[a^1, b^1]$.

Let's consider the case where the sub-interval $[c^1, b^1]$ is eliminated, as shown in Figure 2.5. Suppose $\Delta^3 = d^1 - a^1$. From the above analysis and Figure 2.5, one can see that

$$b^1 - a^1 = (c^1 - a^1) + (b^1 - c^1) = (c^1 - a^1) + (d^1 - a^1) \tag{2.1.4}$$

Therefore

$$\Delta^1 = \Delta^2 + \Delta^3 \tag{2.1.5}$$

More generally, we can have

$$\Delta^k = \Delta^{k+1} + \Delta^{k+2} \tag{2.1.6}$$

Let N denote the number of calculations required and Δ^{N-1} be the length of the interval obtained after N calculations of the function $f(x)$. Define F_{N-k}

as a reduction factor by which the interval Δ^k is reduced to Δ^{N-1} after $N-k$ calculations, that is

$$\Delta^k = F_{N-k}\Delta^{N-1} \tag{2.1.7}$$

Note that since $\Delta^{N-1} = F_1\Delta^{N-1}$ we have $F_1 = 1$.

Dividing the both sides of (2.1.6) by Δ^{N-1}, we get

$$\frac{\Delta^k}{\Delta^{N-1}} = \frac{\Delta^{k+1}}{\Delta^{N-1}} + \frac{\Delta^{k+2}}{\Delta^{N-1}} \tag{2.1.8}$$

From definition (2.1.7), we then have

$$F_{N-k} = F_{N-k-1} + F_{N-k-2}, \quad F_1 = 1, \quad (k = N-3, N-4, ..., 2, 1) \tag{2.1.9}$$

Let $n = N - k$. Then (2.1.9) becomes

$$F_n = F_{n-1} + F_{n-2}, \quad F_1 = 1, \quad (k = 3, 4, ..., N-1) \tag{2.1.10}$$

It is clear from (2.1.10) that since F_1 is fixed the whole sequence will be determined if F_2 can be determined. Assuming $k = 1$ in (2.1.7) leads to the following

$$\Delta^{N-1} = \frac{\Delta^1}{F_{N-1}} \tag{2.1.11}$$

which shows that to have a smallest final interval after N calculations we should have a reduction factor F_{N-1} as large as possible. Therefore, F_2 should be made as large as possible. If we set $k = N-2$ in (2.1.7), we can estimate F_2 as follows

$$F_2 = \frac{\Delta^{N-2}}{\Delta^{N-1}} \tag{2.1.12}$$

From Figure 2.5, we must have $\Delta^n/\Delta^{n-1} \geq 0.5$ for all $n = 3, 4, ..., N-1$. It follows that there must be $\Delta^{N-2}/\Delta^{N-1} \leq 2$, which means that the largest possible F_2 is 2.

The sequence generated from (2.1.9) with $F_1 = 1$ and $F_2 = 2$ is called Fibonacci sequence. Table 2.2 shows the first 16 numbers of the sequence and its limit. The last column of Table 2.2 means that there is the following relationship (Cai, 1982).

$$\lim_{n\to\infty} \frac{F_{n-1}}{F_n} = \lim_{n\to\infty} \frac{F_{n-2}}{F_{n-1}} = 0.618 \tag{2.1.13}$$

In fact, from (2.1.10) we have

$$\frac{F_n}{F_{n-1}} = 1 + \frac{F_{n-2}}{F_{n-1}} \tag{2.1.14}$$

Table 2.2: Fibonacci sequence

n	F_1	$\frac{F_{n-1}}{F_n}$
1	1	1
2	2	0.5
3	3	0.66667
4	5	0.6
5	8	0.625
6	13	0.61538
7	21	0.61905
8	34	0.61765
9	55	0.61818
10	89	0.61798
11	144	0.61806
12	233	0.618026
13	377	0.618037
14	610	0.618033
15	987	0.618034
16	1597	0.6180338
⋮	⋮	⋮
∞		0.618034

Let

$$\lim_{n\to\infty} \frac{F_{n-1}}{F_n} = \lim_{n\to\infty} \frac{F_{n-2}}{F_{n-1}} = r \tag{2.1.15}$$

Then

$$\frac{1}{r} = 1 + r \quad \text{or} \quad r^2 + r - 1 = 0 \tag{2.1.16}$$

Solving the above equation leads to $r \approx 0.618$.

To use this method, the length of the final interval Δ^{N-1} must be given. For a given initial interval Δ^1, we can then use (2.1.7) to find the largest number N and the corresponding factor F_{N-1} from Table 2.2. For instance, if the initial interval has length $\Delta^1 = 1$ and the required precision is $\Delta^{N-1} = 10^{-3}$, then we can determine N from Table 2.2 so that

$$\frac{\Delta^{N-1}}{\Delta^1} = \frac{1}{F_{N-1}} = 10^{-3} \tag{2.1.17}$$

which gives $N = 16$.

2.1.4 The Golden Section Search Method

In the Fibonacci method, it was shown that when $k \to \infty$ the ratio of two successive intervals of the Fibonacci sequence approaches 0.618, or

$$\frac{F_{n-1}}{F_n} \to 0.618 \tag{2.1.18}$$

If this ratio remains the same at every iteration, then the resultant search method is called 0.618 method or the golden section search method. This method can be used when the number of calculations is not known in advance. The ratio can be generated following the same principle used to deduce (2.1.16). In fact, this method consists of taking the lengths of successive intervals to have a fixed ratio γ, or

$$\frac{\Delta_1}{\Delta^2} = \frac{\Delta_2}{\Delta^3} = \ldots = \gamma \tag{2.1.19}$$

so that at iteration $k+1$ the relative disposition of the points could be the same as that at iteration k (Cai, 1982; Minoux, 1986).

Since we have as in (2.1.6)

$$\Delta^k = \Delta^{k+1} + \Delta^{k+2} \tag{2.1.20}$$

and if we impose

$$\frac{\Delta^k}{\Delta^{k+1}} = \frac{\Delta^{k+1}}{\Delta^{k+2}} = \gamma \tag{2.1.21}$$

it follows that

$$\frac{\Delta^k}{\Delta^{k+1}} = 1 + \frac{\Delta^{k+2}}{\Delta^{k+1}} \tag{2.1.22}$$

Combining (2.1.22) with (2.1.21) leads to

$$\gamma = 1 + \frac{1}{\gamma} \tag{2.1.23}$$

or

$$\gamma^2 - \gamma - 1 = 0 \tag{2.1.24}$$

The positive root of (2.1.24) is the golden section number

$$\gamma = \frac{\sqrt{5}+1}{2} \approx 1.618 \tag{2.1.25}$$

Note that the reciprocal of 1.618 is exactly 0.618. It follows that the rate of convergence of the golden section search method is linear with rate 0.618.

Since the golden section search method reduces the lengths of intervals at the fixed ratio 0.628, it is not optimal and different from the Fibonacci method. Suppose the length of the original interval is L_0. After k iterations, the length of the interval generated using the Fibonacci method will be

$$L_k = \frac{L_0}{F_{k+1}} \tag{2.1.26}$$

and the length of the interval generated using the golden section search method will be

$$L_k = (0.618)^k L_0 \tag{2.1.27}$$

The ratios of the lengths of the intervals (L_k/L_0) generated using the two methods are shown in Table 2.3 (Cai, 1982).

Table 2.3: Reduction of the ratio L_k/L_0

k	Fibonacci	Golden Section
1	0.5	0.618
2	0.33	0.382
3	0.2	0.23
4	0.125	0.146
5	0.0777	0.083
6	0.048	0.054
7	0.029	0.034
8	0.018	0.0213
9	0.01	0.013
10	0.00694	0.00813

It is clear from Table 2.3 that for a sufficiently large number of calculations, the golden section search method leads asymptotically to the same disposition of points as in the Fibonacci method.

2.2 Optimisation Conditions

2.2.1 Necessary Conditions for Local Optimality

In the last section, we tried to find an optimum of a function of only one variable. The problem to be discussed in this section is to search for an optimum of a function of n variables $x_1, \ldots, x_n$, each of which can take any real numbers.

Let $x = (x_1, \ldots, x_n)^T$. Suppose $f(x)$ is a continuously differentiable function and has a minimum solution x^*. x^* is called a global minimum if for all $x \in \mathcal{R}^n$, $f(x^*)$ is a minimum, or

$$\forall x \in \mathcal{R}^n : \quad f(x^*) \leq f(x) \tag{2.2.1}$$

In Figure 2.7, the point x^* is a global minimum. If (2.2.1) holds only in a neighbourhood of x^*, then x^* is called a local minimum. In Figure 2.7, the point x^1 is a local minimum only.

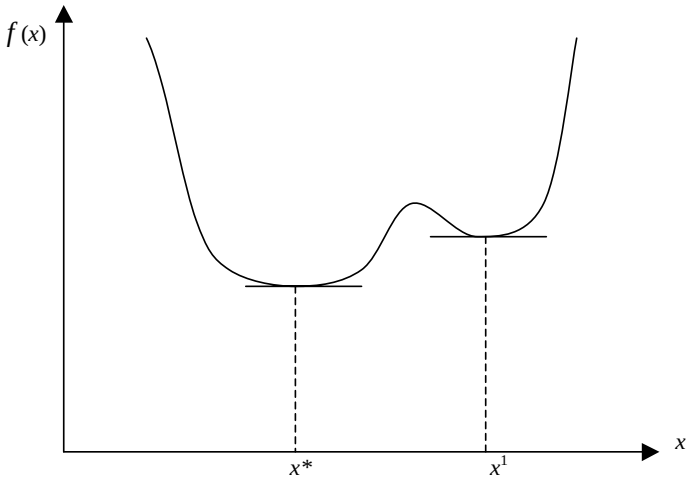

Figure 2.7: Local and global minima

Suppose $f(x)$ has continuous partial derivatives $(\partial f(x)/\partial x_i)$ and second derivatives $(\partial^2 f(x)\ /\partial x_i \partial x_j)$ for all $x \in \mathcal{R}^n$. Then, the necessary conditions for x^* to be a local or global minimum of f can be expressed as

a) $\nabla f(x^*) = \partial f(x^*)/\partial x = 0$ and

b) The Hessian matrix

$$\nabla^2 f(x^*) = \frac{\partial^2 f(x^*)}{\partial x_i \partial x_j} = \begin{bmatrix} \frac{\partial^2 f(x^*)}{\partial x_1 \partial x_1} & \frac{\partial^2 f(x^*)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x^*)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x^*)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x^*)}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f(x^*)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f(x^*)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x^*)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x^*)}{\partial x_n \partial x_n} \end{bmatrix} \tag{2.2.2}$$

must be a positive semi-definite matrix. Note that the condition of positive semi-definiteness is stated by $\forall y \in \mathcal{R}^n$, $y^T \nabla^2 f(x^*) y \geq 0$.

In Figure 2.7, the necessary condition is satisfied at both points x^* and x^1. If a point x^* satisfies condition (a), it is called a stationary point. A stationary point is not guaranteed to be a local minimum because it could be a point of inflexion. In Figure 2.8, the point x^* is an inflexion point.

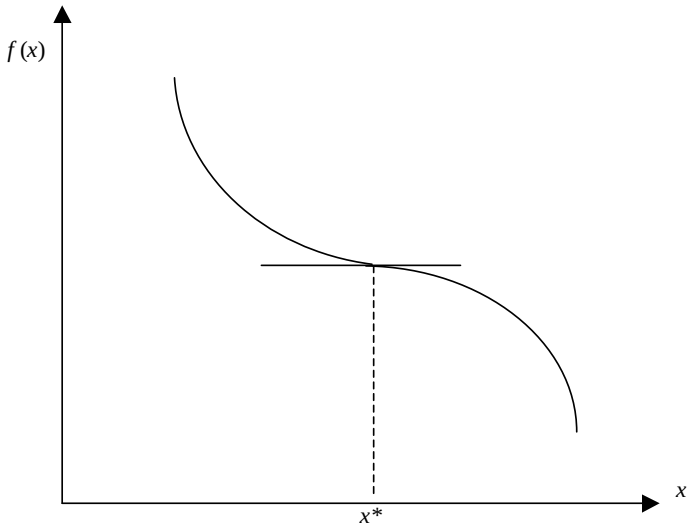

Figure 2.8: A point of inflexion

2.2.2 Sufficient Conditions for Local Optimality

To guarantee that x^* is a local minimum of f, x^* must satisfy the following sufficient conditions:

a) $\nabla f(x^*) = 0$ and

b) The Hessian matrix $\nabla^2 f(x^*)$ must be a positive definite matrix, which means that $\forall y \in \mathcal{R}^n$, $y^T \nabla^2 f(x^*) y > 0$.

The difference between sufficient condition b) and necessary condition b) is that the former means that f is strictly convex in a neighbourhood of x^*. In Figure 2.8, condition b) is not satisfied.

2.3 Unconstrained Optimisation Methods

2.3.1 Steepest Decent Method

The steepest decent method always uses negative gradient directions as search directions. At each iteration, it takes an optimal step size along a negative gradient direction to give the largest decrease of f and is therefore referred to as the optimal gradient method. One of its advantages is that the function f may decrease quickly in initial iterations. Since it only uses first-order derivatives to determine search directions, its overall convergence to an optimum may be very slow. This is often the case around an optimum point.

Before we introduce the method, the concept of gradient is discussed first.

The gradient of a nonlinear function f is defined as follows

$$\nabla f(x^*) = \frac{\partial f}{\partial x} = \left[\frac{\partial f}{x_1}, \quad \frac{\partial f}{x_2}, \quad \ldots, \quad \frac{\partial f}{x_n}\right]^T \tag{2.3.1}$$

which is a n-dimensional vector.

The steepest decent method uses a negative gradient direction as a search direction. Let $P = \nabla f(x)$ be a gradient vector. Then, the unit vector of a gradient direction is given by

$$S = \frac{\nabla f(x)}{\| \nabla f(x) \|} = \frac{P}{\| \nabla f(x) \|} \tag{2.3.2}$$

The gradient direction S is proportional to the normal vector of the function f. A negative gradient direction is given by $-S$. $\| \nabla f(x) \|$ is the 2-norm of the gradient $\nabla f(x)$, defined by

$$\| \nabla f(x) \| = \sqrt{\left(\frac{\partial f}{\partial x_1}\right)^2 + \left(\frac{\partial f}{\partial x_2}\right)^2 + \ldots + \left(\frac{\partial f}{\partial x_n}\right)^2} \tag{2.3.3}$$

Example 2.1 Calculate the gradient of the following function at the point $x^0 = (2, 1)$

$$f(x) = x_1^2 + 2x_1x_2 + 3x_2^2$$

Solution: The gradient of the above function and its 2-norm at $x^0 = (2, 1)$ are given by

$$\nabla f(x^0) = \begin{bmatrix} 2x_1 + 2x_2 \\ 2x_1 + 6x_2 \end{bmatrix}_{x=x^0} = \begin{bmatrix} 6 \\ 10 \end{bmatrix}$$

$$\| \nabla f(x^0) \| = \sqrt{136}$$

The unit vector of the gradient direction is thus given by

$$S = \frac{\nabla f(x^0)}{\| \nabla f(x^0) \|} = \frac{1}{\sqrt{136}} \begin{bmatrix} 6 \\ 10 \end{bmatrix}$$

The above example shows that for a nonlinear function gradient directions are different at different points.

The simplest gradient method is the one with predetermined steps. Start from an initial point x^0 and compute the gradient $\nabla f(x^0)$ at x^0. Since $\nabla f(x^0)$ gives the direction of the largest increase of f, make a step size λ_0 in the direction opposite to the gradient, and find the next point

$$x^1 = x^0 - \lambda_0 \frac{\nabla f(x^0)}{\| \nabla f(x^0) \|} \tag{2.3.4}$$

The procedure is repeated and a sequence of points $x^0, x^1, \ldots, x^k$ is generated using the following update equation

$$x^{k+1} = x^k - \lambda_k \frac{\nabla f(x^k)}{\| \nabla f(x^k) \|}, \quad \forall k, \quad \lambda_k > 0 \tag{2.3.5}$$

In the gradient method with predetermined steps, we choose the step sizes λ_k in advance. This makes the method very simple to implement. The main drawback of this method is that the convergence may be very slow.

The steepest decent method is a gradient method with optimal step sizes. In this frequently used method, λ_k is determined so as to minimise the function f in terms of λ

$$g(\lambda) = f[x^k - \lambda \nabla f(x^k)] \tag{2.3.6}$$

for all $\lambda \geq 0$ using one-dimensional optimisation methods.

The steepest decent method can be summarised as follows:

a) Choose a starting point x^0 and let $k = 0$.

b) At step k, determine the search direction by calculating the unit negative gradient $d_k = -\nabla f(x^k) / \| \nabla f(x^k) \|$.

c) Find the optimal step size λ_k such that

$$f(x^k + \lambda_k d_k) = \min_{\lambda \geq 0} (x^k + \lambda d_k) \tag{2.3.7}$$

d) Calculate a new point $x^{k+1} = x^k + \lambda_k d_k$.

e) Conduct a stopping test. If satisfied, stop the procedure. Otherwise let $k = k + 1$ and return to b).

Since the convergence is generally not finite, a stopping test must be conducted. Some of the most frequently used criteria are given as follows:

Criterion 1: $\max_{i=1,\ldots,n} |\frac{\partial f}{\partial x_i}| < \varepsilon \ (\varepsilon > 0)$

Criterion 2: $\| \nabla f(x) \|^2 = \sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i} \right)^2 \ (\varepsilon > 0)$

Criterion 3: $|f(x^{k+1}) - f(x^k)| < \varepsilon \ (\varepsilon > 0)$

Regarding the convergence of the steepest decent method, there is the following conclusion. If the function f is continuously differentiable with the property $f(x) \to \infty$ for $\| x \| \to \infty$, then with any starting point the steepest decent method with one-dimensional optimisation converges to a stationary point of f.

Example 2.2 Use the steepest decent method to find the minimum of the following function (Cai, 1982),

$$f(x) = x_1^2 + 25 x_2^2$$

Solution: Given a starting point $x^0 = [2, 2,]^T$, we can calculate the value of the function, the gradient and the 2-norm as follows

$$\begin{aligned} f(x^0) &= 104 \\ \nabla f(x^0) &= \begin{bmatrix} 2x_1 \\ 50x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 100 \end{bmatrix} \\ \| \nabla f(x^0) \| &= \sqrt{4^2 + 100^2} = \sqrt{10016} \end{aligned}$$

Then the search direction at x^0 is given by

$$d_0 = \frac{\nabla f(x^0)}{\| \nabla f(x^0) \|} = \frac{[4, 100]^T}{\sqrt{10016}} \approx [0.04, \quad 1]^T$$

The next point is calculated by

$$x^1 = x^0 + \lambda d_0 = [2 + 0.04\lambda, \quad 2 + \lambda]^T$$

where λ is the step size at x^0. The function f can be expressed as a function of λ as follows.

$$f(\lambda) = (2 + 0.04\lambda)^2 + 25(2 + \lambda)^2$$

This function only has a single variable. We can find the optimal value for λ using the optimality (stationarity) condition. Let

$$\frac{df(\lambda)}{d\lambda} = 2(2 + 0.04\lambda)0.04 + 50(2 + \lambda) = 0$$

We obtain the optimal step size $\lambda_0 = -2.003$. The new point is then given by

$$x^1 = [2 + 0.04\lambda, \quad 2 + \lambda]^T = [1.92, \quad -0.003]^T$$

The value of the function at x^1 is 3.69, or $f(x^1) = 3.69$. The decrease of the function along the negative gradient is quite large after just one step. Table 2.4 summarises the results for the first three iterations.

Table 2.4: Results for the steepest decent method

	x_1	x_2	$f(x)$
start	2	2	104
λ_0=2.003	1.92	-0.003	3.69
λ_1=1.85	0.07	0.07	0.13
λ_2=0.07	0.07	0	0.0049

Table 2.5 shows the results generated using the gradient method with the predetermined step size $\lambda = 1$.

Table 2.5: Results for the gradient

k	x_1	x_2	$\nabla f(x^k)^T$	$f(x^k)$
0	2	2	[4, 100]	104
1	1.96	1	[3.92, 50]	28.8416
2	1.88	0	[3.76, 0]	3.5344
3	0.88	0	[1.76, 0]	0.7744
4	-0.12	0	[-0.24, 0]	0.0144
5	0.88	0	[1.76, 0]	0.7744

It is clear from Table 2.5 that for a fixed step size the convergence is slow if the step size is chosen to be small. For a large fixed step size, oscillation happens around the optimum point. In comparison, the convergence of the steepest decent method is much faster.

For any function $f(x)$, an optimal step size can be determined using the second order Taylor series

$$f(x^{k+1}) = f(x^k) + \nabla f(x^k)\Delta x + \frac{1}{2}\Delta x^T A \Delta x \tag{2.3.8}$$

$$\Delta x = x^{k+1} - x^k \tag{2.3.9}$$

where $A = \nabla^2 f(x^k)$ is the Hessian matrix.

Let $x^{k+1} = x^k + \lambda d_k$ and $\Delta x = \lambda d_k$ with d_k being a unit vector. Then (2.3.8) becomes

$$f(x^k + \lambda d_k) = f(x^k) + \nabla f(x^k)\lambda d_k + \frac{1}{2}(\lambda d_k)^T A \lambda d_k \tag{2.3.10}$$

In the above function, taking the first-order derivative leads to

$$\frac{df(x^k + \lambda d_k)}{d\lambda} = 0 \tag{2.3.11}$$

or

$$\nabla f(x^k) d_k + (d_k)^T A d_k \lambda = 0 \tag{2.3.12}$$

An optimal step size is then given by

$$\lambda^* = -\frac{\nabla f(x^k) d_k}{(d_k)^T A d_k} \tag{2.3.13}$$

2.3.2 Newton's Method

Newton's method is one of several traditional methods used to solve a set of nonlinear equations. As we discussed in section 2.1, at a local minimum a

continuously differential function $f(x)$ satisfies the following necessary (stationary) condition

$$\nabla f(x^*) = 0 \tag{2.3.14}$$

This is a set of n nonlinear equations. Solving this set of equations results in x^*. Newton's method can be used for this purpose. Let's assume that the function f is twice continuously differentiable and that all second derivatives can be calculated.

Suppose the k-th iteration results in a solution x^k. The basic idea in Newton's method consists of replacing the function f by its quadratic approximation in the neighbourhood of the point x^k

$$q(x) = f(x^k) + \nabla f(x^k)(x - x^k) + \frac{1}{2}(x - x^k)\nabla f^2(x^k)(x - x^k) \tag{2.3.15}$$

Then a new point x^{k+1} is taken as the minimum of $q(x)$. This requires that $\nabla f^2(x^k)$ be a positive definite matrix or the function $q(x)$ strictly convex. If this is the case, then x^{k+1} is a unique minimum of $q(x)$, at which we have

$$\nabla q(x^{k+1}) = 0 \tag{2.3.16}$$

The first order derivative of $q(x)$ is given by

$$\nabla q(x) = \frac{dq(x)}{dx} = \nabla f(x^k) + \nabla f^2(x^k)(x - x^k) \tag{2.3.17}$$

Combining (2.3.16) with (2.3.17) leads to the linear equations

$$-\nabla f(x^k) = \nabla f^2(x^k)(x^{k+1} - x^k) \tag{2.3.18}$$

which can be rearranged as the following recurrence equation

$$x^{k+1} = x^k - [\nabla f^2(x^k)]^{-1}\nabla f(x^k) \tag{2.3.19}$$

which is nothing other than Newton's method (Minoux, 1986).

In (2.3.19), the search direction and the step size are generated as $-[\nabla f^2(x^k)]^{-1}\nabla f(x^k)$. If f is a strictly convex quadratic function, we have $f = q(x)$ and x^{k+1} as generated in (2.3.19) is the unique minimum solution of f. In this case, the method converges in a single step.

For an arbitrary function, x^{k+1} generated in (2.3.19) may not lead to the decrease of the function, to say nothing of reaching the minimum of f in one step. In these circumstances, (2.3.19) needs to be revised to guarantee that x^{k+1} always leads to the decrease of f compared with x^k. First of all, since the approximation of $f(x)$ by $q(x)$ is only valid in the neighbourhood of x^k, the step size needs to be controlled in the recurrence equation. For instance, (2.3.19) may be re-written as follows

$$x^{k+1} = x^k - \lambda_k[\nabla f^2(x^k)]^{-1}\nabla f(x^k) \tag{2.3.20}$$

Equation (2.3.20) will be identical to (2.3.19) if we set $\lambda_k = 1$.

λ_k should be properly chosen so that $f(x)$ can be minimised as far as possible using (2.3.20). For example, we can use a one-dimensional search method to find λ_k that minimises $f(x^k + \lambda_k d_k)$ where d_k is the search direction

$$d^k = -[\nabla f^2(x^k)]^{-1} \nabla f(x^k) \tag{2.3.21}$$

Another simple way of choosing the step size is to set $\lambda_k = 1$ initially. If it is true that $f(x^k + d_k) < f(x^k)$, then x^{k+1} is a desirable new solution. Otherwise, the optimal step size λ_k that minimise $f(x^k + \lambda_k d_k)$ must lie between 0 and 1. We could try to use another step size in this range, for example the middle point between 0 and 1, that is $\lambda_k = 0.5$. If $f(x^k + 0.5 d_k) < f(x^k)$, we set $x^{k+1} = x^k + 0.5 d_k$. Otherwise, we repeat the procedure until a sufficiently small step size is found so that $f(x^k + \lambda_k d_k) < f(x^k)$.

To guarantee that x^{k+1} obtained using (2.3.20) provides a better solution than x^k, the Hessian matrix $\nabla f^2(x^k)$ needs to be positive definite. However, this may not always be the case. In this situation, the search direction $-\nabla f^2(x^k) \nabla f(x^k)$ may not be a decent direction of f. Consequently, the convergence of the method to an optimal solution is not guaranteed.

If it happens that the Hessian matrix $\nabla f^2(x^k)$ is not positive definite, the matrix may be slightly perturbed so that a positive definite matrix M_k is generated. Equation (2.3.20) can then be modified as follows

$$x^{k+1} = x^k - \lambda_k [M_k]^{-1} \nabla f(x^k) \tag{2.3.22}$$

where λ_k can be determined using one of the techniques discussed above. Note that the positive definiteness of M_k ensures that the search direction $d_k = -[M_k]^{-1} \nabla f(x^k)$ is a decent direction of f. In fact, we have

$$\nabla f(x^k) d_k = -\nabla f^T(x^k) [M_k]^{-1} \nabla f^T(x^k) < 0 \tag{2.3.23}$$

M_k can be generated from $\nabla f^2(x^k)$ to improve the convergence (Minoux, 1986). For instance, the following perturbation can be used to construct M_k

$$M_k = \mu_k I + \nabla f^2(x^k) \tag{2.3.24}$$

where $\mu_k > 0$ is a real number that is chosen to be sufficiently small, subject to the constraint that all eigenvalues of M_k be larger than or equal to a given positive constant. The overall convergence of the method can be proved.

Example 2.3 Use Newton's method to find the minimum of the following function (Cai, 1982).

$$f(x) = \frac{3}{2} x_1^2 + \frac{1}{2} x_2^2 - x_1 x_2 - 2 x_1$$

Solution: It can be shown using the necessary optimality condition that the minimum point is

$$x^* = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Rewrite the above function as

$$f(x) = C^T x + \frac{1}{2} x^T Q x$$

where

$$C^T = [-2 \quad 0]$$
$$Q = \begin{bmatrix} 3 & -1 \\ -1 & 1 \end{bmatrix}$$

Select a starting point arbitrarily as follows

$$x^0 = \begin{bmatrix} -2 \\ 4 \end{bmatrix}$$

We then have

$$\nabla f(x^0) = \begin{bmatrix} -12 \\ 6 \end{bmatrix}, \quad Q^{-1} = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 3/2 \end{bmatrix}$$

Using (2.3.19), we get

$$x^1 = x^0 - Q^{-1}\nabla f(x^0) = \begin{bmatrix} -2 \\ 4 \end{bmatrix} - \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 3/2 \end{bmatrix} \begin{bmatrix} -12 \\ 6 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

which is the minimum point. For this quadratic function, Newton's method indeed converges to the minimum point in one step.

2.3.3 Quasi-Newton's Methods

These methods are based on a generalisation of the recurrence equation of Newton's method (2.3.20) (Minoux, 1986):

$$x^{k+1} = x^k - \lambda_k [\nabla f^2(x^k)]^{-1} \nabla f(x^k) \tag{2.3.25}$$

One of the drawbacks of Newton's method is the requirement that $\nabla f^2(x^k)$ be positive definite. It is also time consuming to calculate the inverse of $\nabla f^2(x^k)$ for determining the search direction. A natural extension of Newton's method is to replace $[\nabla f^2(x^k)]^{-1}$ by a positive definite matrix H_k that is used to calculate the search direction from the gradient $\nabla f(x^k)$. Equation (2.3.25) can then be modified as follows

$$x^{k+1} = x^k - \lambda_k H_k \nabla f(x^k) \tag{2.3.26}$$

λ_k could be determined so that $f(x^k + \lambda_k d_k)$ is minimised in the direction $d_k = -H_k \nabla f(x^k)$, or at least $f(x^k + \lambda_k d_k) < f(x^k)$.

The matrix H_k needs to be revised at every iteration. It is desirable that for a quadratic function

$$f(x) = \frac{1}{2}x^T Ax + b^T x + c \tag{2.3.27}$$

H_k converges to the inverse A^{-1} of the Hessian matrix A of f. At the end of the convergence, (2.3.26) is consistent with (2.3.25). For an arbitrary function f, H_k should be constructed as an approximation to the inverse of the Hessian matrix of f.

We know that the difference of a function at two successive points contains information about the first order derivative of the function between the two points, for example

$$\frac{\partial f(x)}{\partial x}\Big|_{x^k} \approx \frac{f(x^k + \Delta x^k) - f(x^k)}{\Delta x^k} \tag{2.3.28}$$

where

$$\Delta x^k = x^{k+1} - x^k \tag{2.3.29}$$

Similarly, the difference of the gradients of a function at two successive points contains information about the second order derivative of the function between the two points. On the basis of this recognition, the following condition can be imposed

$$H_{k+1}[\nabla f(x^{k+1}) - \nabla f(x^k)] = x^{k+1} - x^k \tag{2.3.30}$$

Given H_k, then H_{k+1} can be obtained using a correction formula and the new information generated at the kth step of the algorithm, including the gradient $\nabla f(x^{k+1})$. Various correction formulae have been developed, most of which are of the type

$$H_{k+1} = H_k + \Delta_k \tag{2.3.31}$$

Δ_k could be of rank 1 or rank 2.

A simple correction formula used to construct an approximation to the inverse of the Hessian matrix is to choose a matrix of rank 1 and the form

$$\Delta_k = \alpha_k u_k u_k^T \tag{2.3.32}$$

u_k and α_k are a vector and a scalar, respectively, and are chosen so that (2.3.30) is satisfied.

One of the features of the above correction is that it preserves the symmetry of the matrices H_k if the initial matrix H_0 is symmetric. Using this feature, we can obtain a correction formula of rank 1 as follows. First, let

$$\delta_k = x^{k+1} - x^k \tag{2.3.33}$$

$$\gamma_k = \nabla f(x^{k+1}) - \nabla f(x^k) \tag{2.3.34}$$

Then, u_k and α_k can be represented in terms of δ_k and γ_k.

From (2.3.30), (2.3.33) and (2.3.34), we have

$$H_{k+1}\gamma_k = \delta_k \tag{2.3.35}$$

Combining (2.3.35) with (2.3.31) and (2.3.32) leads to

$$[H_k + \alpha_k u_k u_k^T]\gamma_k = \delta_k \tag{2.3.36}$$

By multiplying both sides by γ_k^T, we have

$$\gamma_k^T H_k \gamma_k + \alpha_k(\gamma_k^T u_k)(u_k^T \gamma_k) = \gamma_k^T \delta_k \tag{2.3.37}$$

or

$$\alpha_k(u_k^T \gamma_k)^2 = \gamma_k^T(\delta_k - H_k\gamma_k) \tag{2.3.38}$$

Note that there is the identity

$$\alpha_k(u_k u_k^T) = \frac{(\alpha_k u_k u_k^T \gamma_k)(\alpha_k u_k u_k^T \gamma_k)^T}{\alpha_k(u_k^T \gamma_k)^2} \tag{2.3.39}$$

From (2.3.36), we have

$$\alpha_k u_k u_k^T \gamma_k = \delta_k - H_k\gamma_k \tag{2.3.40}$$

Substituting (2.3.38) and (2.3.40) into (2.3.39) results in the following correction formula of rank 1

$$H_{k+1} - H_k = \alpha_k(u_k u_k^T) = \frac{(\delta_k - H_k\gamma_k)(\delta_k - H_k\gamma_k)^T}{\gamma_k^T(\delta_k - H_k\gamma_k)} \tag{2.3.41}$$

or

$$H_{k+1} = H_k + \frac{(\delta_k - H_k\gamma_k)(\delta_k - H_k\gamma_k)^T}{\gamma_k^T(\delta_k - H_k\gamma_k)} \tag{2.3.42}$$

It can be proven that for a quadratic function f as shown in (2.3.27), the sequence of matrices H_k generated using (2.3.42) converges to the inverse of the Hessian matrix of f.

To calculate H_{k+1} using (2.3.42), one does not have to choose the point x^{k+1} as the minimum of f in the direction $d_k = -H_k\nabla f(x^k)$ starting from the point x^k. This is one of the advantages of the correction formula (2.3.42) and the reason why this formula has been widely used for the development of quasi-Newton's methods that do not require one-dimensional search.

The correction formula ((2.3.42)) has the disadvantage that even for a quadratic function f with H_0 and the Hessian matrix of f positive definite the matrices H_{k+1} obtained are not necessarily positive definite. Furthermore, the correction formula may not be used at all if the denominator $\gamma_k^T(\delta_k - H_k\gamma_k)$ is zero or very small. In such circumstances, other correction formulae of rank 2 need to be used, which do not have this drawback. However, they require the use of one-dimensional search to find x^{k+1}. For more detail about correction formulae of rank 2, the reader can refer to Minoux (1986).

2.4 Summary

In this chapter, we discussed three one-dimensional search methods and three unconstrained optimisation methods. The dichotomy method is easy to understand and simple to implement. However, it is not as efficient as the Fibonacci method that leads to the smallest possible reduced interval for a given number of calculations. The drawback of the Fibonacci method is that the number of calculations needs to be identified in advance. The golden section search method does not suffer from this drawback and provides an efficient search procedure, though it is not an optimal method as the Fibonacci method.

Three typical unconstrained optimisation methods were introduced. They are: the steepest decent method, Newton's method and quasi-Newton methods. The steepest decent method converges quite fast in initial iterations but may be very slow approaching an optimal solution. Newton's method can find the optimal solution of a quadratic problem at one step. For a general nonlinear problem whose Hessian matrix is not positive definite, perturbation methods could be used to improve the convergence. There is a family of quasi-Newton methods. In this chapter, a method with a correction formula of rank 1 was discussed, which does not guarantee to produce a positive definite correction matrix. More methods can be found in the references.

Chapter 3

Constrained Optimisation

3.1 Introduction

In the previous chapter, we discussed typical methods for optimisation without any constraints. In most real life optimisation problems, variables can only take certain values that are restricted by constrains. Constrained optimisation is also called mathematical programming and in general can be represented as follows

$$\begin{aligned} \text{Min} \quad & f(x) && (3.1.1)\\ s.t. \quad & h_i(x) = 0 \quad i = 1, 2, ..., m && (3.1.2)\\ & g_j(x) \geq 0 \quad j = 1, 2, ..., l && (3.1.3) \end{aligned}$$

The above problem is called linear programming if all functions in (3.1.1), (3.1.2) and (3.1.3) are linear functions of x. Otherwise, it is called nonlinear programming. If a mathematical programming problem has equality constraints only ((3.1.2), it can be solved by transforming it to an unconstrained problem.

It is more complicated to solve a mathematical programming problem with both equality and inequality constraints. This is due to the fact that a solution procedure must guarantee not only the reduction of the objective function but also the feasibility of solutions generated. Such requirements pose challenges for finding optimal solutions. To simplify an optimisation procedure, it is common to transform an inequality constrained problem to an equality constrained problem, a constrained problem to an unconstrained problem, and a nonlinear programming problem to a linear programming problem. In other words, a nonlinear programming problem could be solved using a sequence of linear approximations. In this chapter, we will describe several nonlinear programming methods that employ the above ideas.

3.2 Optimality Conditions

3.2.1 Basic Concepts

In (3.1.1), (3.1.2) and (3.1.3), suppose $f(x)$, $h_i(x)$ and $g_j(x)$ $(i = 1, 2, ..., m, j = 1, 2, ..., l)$ are all continuously differentiable. We use R to denote the feasible space enclosed by (3.1.2) and (3.1.3).

We first introduce the concepts of feasible directions and saturated constraints. Suppose x^0 is a feasible solution, or $x^0 \in R$. For a search direction d, if there exists $\lambda_0 > 0$ so that for any λ $(0 \leq \lambda \leq \lambda_0)$ the following is true

$$x^0 + \lambda d \in R \tag{3.2.1}$$

then d is called a feasible direction at the point x^0.

If x^0 is not an optimal solution, the next step is to look for a direction along which the objective function $f(x)$ will decrease. Such a direction is called the feasible decent direction of $f(x)$ at x^0. Using the first order Taylor series to expand $f(x)$ at x^0, we know that d must be a feasible decent direction if it satisfies the following condition

$$\nabla f(x^0)^T d < 0 \tag{3.2.2}$$

For a nonlinear programming problem (3.1.1), (3.1.2) and (3.1.3), (3.1.2) is satisfied at a feasible point x^0. For inequality constrain (3.1.3), however, there are two possible cases at x^0: (a) $g_j(x^0) > 0$ and (b) $g_j(x^0) = 0$. In the first case, x^0 is inside the feasible space formed by the constraint $g_j(x^0) \geq 0$ This means that $g_j(x^0) \geq 0$ does not pose any restriction on the solution x^0. In other words, $g_j(x^0) \geq 0$ is not saturated (or unbinding) at x^0. In the second case, x^0 is on the boundary of the feasible space formed by $g_j(x^0) \geq 0$. So this constraint poses a restriction on the solution x^0 and it is therefore called saturated (binding) constraint. Obviously, all equality constraints are saturated constraints.

Any direction from an interior point of the feasible space is a feasible one. If a point is on the boundary of the feasible space, however, a feasible direction from the point is restricted by the constraint that is saturated at this point.

Suppose the constraint $g_j(x) \geq 0$ is saturated at x^0. Its gradient $\nabla g_j(x^0)$ points to the normal direction of $g_j(x) = 0$ at x^0, which is orthogonal to the tangent plane of $g_j(x) = 0$ at x^0 and along which the function $g_j(x)$ increases the fastest. If the solution x^0 is not an optimum, then the search process should continue. Suppose d is a search direction. If the angle between d and $\nabla g_j(x^0)$ is less than 90^0, or

$$\nabla g_j(x^0)^T d > 0 \tag{3.2.3}$$

then d is a feasible direction. On the other hand, a search direction should also be a direction along which the objective function decreases, or (3.2.2) should be met.

In summary, if x^0 is a feasible solution of a nonlinear programming problem (3.1.1), (3.1.2) and (3.1.3) but not an optimum, then further search is required and a search direction d should satisfy both (3.2.2) and (3.2.3). In other words, the angle between a search direction and the negative gradient direction of the objective function or the gradient direction of a saturated constraint should both be an acute angle.

3.2.2 Kuhn-Tucker Necessary Condition

The Kuhn-Tucker condition provides a necessary condition that an optimal solution must satisfy. It can be used for mathematical programming problems with both equality and inequality constraints. However, it is generally not a sufficient condition, which means that a solution stratifying the Kuhn-Tucker condition is not guaranteed to be an optimal solution. For a convex mathematical programming problem, however, it is not only a necessary but also a sufficient condition for local optimality.

To describe the Kuhn-Tucker condition, we first define a regular point as follows. Suppose x^* is a point that satisfies (3.1.2) and (3.1.3). Let J be the set of the subscripts of inequality constraints such that $g_j(x^*) = 0$. If gradient vectors $\nabla h_i(x^*)$ and $\nabla g_j(x^*)$ $(1 \leq i \leq m;\ j \in J)$ are linearly independent, then x^* is called a regular point of the constraints (3.1.2) and (3.1.3). In other words, if the gradients of the saturated constraints at x^* are linearly independent, then x^* is a regular point, or x^* is regular to the set of the saturated constraints.

The Kuhn-Tucker condition is stated as follows.

Suppose $f(x)$, $h_i(x)$ and $g_j(x)$ are continuously differentiable, x^* is a local minimum of a nonlinear programming problem (3.1.1), (3.1.2) and (3.1.3), and x^* is a regular point of the problem. Then, there exists vectors $\lambda = [\lambda_1^*, \lambda_2^*, ..., \lambda_m^*]^T$ and $\mu = [\mu_1^*, \mu_2^*, ..., \mu_l^*]^T$ so that

$$\nabla f(x^*) - \sum_{i=1}^{m} \lambda_i^* \nabla h_i(x^*) - \sum_{j=1}^{l} \mu_j^* \nabla g_j(x^*) = 0 \tag{3.2.4}$$

$$\mu_j^* g_j(x^*) = 0, \quad j = 1, 2, ..., l \tag{3.2.5}$$

$$\mu_j^* \geq 0, \quad j = 1, 2, ..., l \tag{3.2.6}$$

It is clear from (3.2.5) that μ_j^* is positive only when the corresponding constraint is saturated, or $g_j(x^*) = 0$. Otherwise, there must be $\mu_j^* = 0$.

The following example can be used to interpret the geographic meaning of the Kuhn-Tucker condition (Li and Qian, 1982).

$$\text{Min} \qquad f(x) = (x_1 - 2)^2 + (x_2 - 1)^2 \tag{3.2.7}$$

$$s.t. \qquad g_1(x) = x_2 - x_1^2 \geq 0 \tag{3.2.8}$$

$$g_2(x) = 2 - x_1 - x_2 \geq 0 \tag{3.2.9}$$

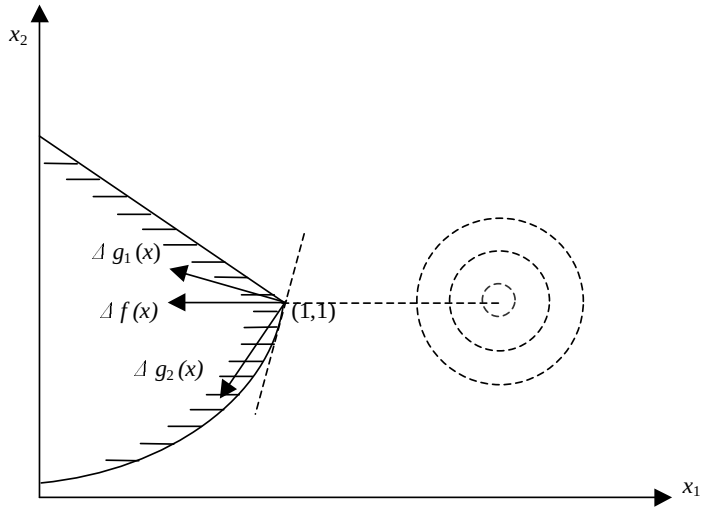

Figure 3.1: Feasible space of Example 3.1

Figure 3.1 shows the feasible region of the above problem and the contour of the objective function. Since the objective requires that an optimal solution be the one that is the closest to the point $(2,1)$, it is clear that the point $(1,1)$ is the minimum. As afore-stated, the negative gradient direction $-\nabla f(x)$ of the objective function points to the steepest decent direction of the function. If the angle between a search direction and $-\nabla f(x)$ is less than 90^o, then $f(x)$ can be reduced further.

The gradient direction $\nabla g_j(x)$ of a constraint function $g_j(x)$ is the fastest ascent direction of the function. At the point $(1,1)$, the vector $-\nabla f(x)$ is between $-\nabla g_1(x)$ and $-\nabla g_2(x)$, which means that one cannot find a feasible direction along which $f(x)$ can be reduced further. Therefore, the point $(1,1)$ is the optimal solution. On the other hand, if $-\nabla f(x)$ is outside the angle between $-\nabla g_1(x)$ and $-\nabla g_2(x)$, then there exist other solutions in the feasible space, at which the objective function can be improved.

The above analysis shows that if x^* is a minimum, $-\nabla f(x^*)$ can be represented using a non-negative linear combination of $-\nabla g_1(x^*)$ and $-\nabla g_2(x^*)$. This is actually what the Kuhn-Tucker condition means.

3.2.3 Second Order Sufficient Conditions

The second order sufficient condition for an optimal solution of the nonlinear programming problem (3.1.1), (3.1.2) and (3.1.3) can be stated as follows.

Suppose x^* is feasible for the constraints (3.1.1) and (3.1.2). If there exist vectors $\lambda = [\lambda_1^*, \lambda_2^*, ..., \lambda_m^*]^T$ and $\mu = [\mu_1^*, \mu_2^*, ..., \mu_l^*]^T$ so that the Kuhn-Tucker

condition (3.2.4), (3.2.5) and (3.2.6) are met and for any non-zero vector z satisfying the following condition

$$z^T \nabla g_j(x^*) = 0 \quad j \in J \quad \text{and} \quad \mu_j^* > 0 \tag{3.2.10}$$

$$z^T \nabla g_j(x^*) \geq 0 \quad j \notin J \quad \text{and} \quad \mu_j^* = 0 \tag{3.2.11}$$

$$z^T \nabla h_i(x^*) = 0 \quad j \notin J \text{ for all equality constraint } h_i(x) = 0 \tag{3.2.12}$$

the following is true

$$z^T [\nabla^2 f(x^*) - \sum_{i=1}^{m} \lambda_i^* \nabla^2 h_i(x^*) - \sum_{j=1}^{l} \mu_j^* \nabla^2 g_j(x^*)] z > 0 \tag{3.2.13}$$

then x^* is a local minimum of the nonlinear programming (3.1.1), (3.1.2) and (3.1.3).

In (3.2.10), (3.2.11) and (3.2.12), $\nabla^2 f(x^*)$, $\nabla^2 h_i(x^*)$ and $\nabla^2 g_j(x^*)$ are the Hessian matrices of $f(x)$, $h_j(x)$ and $g_j(x)$ respectively, or

$$\nabla^2 f(x^*) = \begin{bmatrix} \frac{\partial^2 f(x^*)}{\partial x_1^2} & \frac{\partial^2 f(x^*)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x^*)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x^*)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x^*)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x^*)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x^*)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x^*)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x^*)}{\partial x_n^2} \end{bmatrix} \tag{3.2.14}$$

$$\nabla^2 h_i(x^*) = \begin{bmatrix} \frac{\partial^2 h_i(x^*)}{\partial x_1^2} & \frac{\partial^2 h_i(x^*)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 h_i(x^*)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 h_i(x^*)}{\partial x_2 \partial x_1} & \frac{\partial^2 h_i(x^*)}{\partial x_2^2} & \cdots & \frac{\partial^2 h_i(x^*)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 h_i(x^*)}{\partial x_n \partial x_1} & \frac{\partial^2 h_i(x^*)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 h_i(x^*)}{\partial x_n^2} \end{bmatrix} \tag{3.2.15}$$

$$\nabla^2 g_j(x^*) = \begin{bmatrix} \frac{\partial^2 g_j(x^*)}{\partial x_1^2} & \frac{\partial^2 g_j(x^*)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 g_j(x^*)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 g_j(x^*)}{\partial x_2 \partial x_1} & \frac{\partial^2 g_j(x^*)}{\partial x_2^2} & \cdots & \frac{\partial^2 g_j(x^*)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 g_j(x^*)}{\partial x_n \partial x_1} & \frac{\partial^2 g_j(x^*)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 g_j(x^*)}{\partial x_n^2} \end{bmatrix} \tag{3.2.16}$$

In the above sufficient condition, it is required that $f(x)$, $h_i(x)$ and $g_j(x)$ be second order continuously differentiable. A more general sufficient condition

for local optimality is based on saddle points and Lagrange function. For simplicity of description, let's consider nonlinear programming problems of the following type

$$\begin{aligned} \text{Min} \quad & f(x) && (3.2.17)\\ s.t. \quad & g_j(x) \leq 0, \quad j = 1, 2, ..., l && (3.2.18)\\ & x \in S \subset \mathcal{R}^n && (3.2.19) \end{aligned}$$

where S could be a set of real or integer numbers.

Suppose a constraint $g_j(x) \leq 0$ is associated with a real number $\lambda_j \geq 0$, or Lagrange multiplier. The Lagrange function associated with the problem defined by (3.2.17), (3.2.18) and (3.2.19) is given as follows

$$L(x, \lambda) = f(x) + \sum_{j=1}^{l} \lambda_j g_j(x) \tag{3.2.20}$$

Define a saddle point as follows. Let $\bar{x} \in S$ and $\bar{\lambda} \geq 0$. Then, the point $(\bar{x}, \bar{\lambda})$ is said to be a saddle point of $L(x, \lambda)$ if the following is true

$$L(\bar{x}, \bar{\lambda}) \leq L(x, \bar{\lambda}) \quad \forall x \in S \tag{3.2.21}$$

$$L(\bar{x}, \lambda) \leq L(\bar{x}, \bar{\lambda}) \quad \forall \lambda \geq 0 \tag{3.2.22}$$

A saddle point $(\bar{x}, \bar{\lambda})$ can be generated using the following conditions (Minoux, 1986):

a) $L(\bar{x}, \bar{\lambda}) = \min_{x \in S} L(x, \bar{\lambda})$

b) $g_j(\bar{x}) \leq 0 \quad j = 1, 2, ..., l$

c) $\bar{\lambda}_j g_j(\bar{x}) = 0 \quad j = 1, 2, ..., l$

Furthermore, if $(\bar{x}, \bar{\lambda})$ is a saddle point of $L(x, \lambda)$, then the solution $\bar{x}$ is a global minimum of the problem given by (3.2.17), (3.2.18) and (3.2.19). This conclusion is general and applies to any mathematical programme, whether it is convex or non-convex, or $f(x)$ and $g_j(x)$ are differentiable or not, or S is a continuous, discrete or finite set (Minoux, 1986). However, a saddle point may not exist for some problems and this is in general the case for non-convex problems.

Example 3.1 Check if the point $x^* = (-2.37, -1.84)$ is the minimum of the following problem (Li and Qian, 1982):

$$\begin{aligned} \text{Min} \quad & f(x) = x_1^2 + x_2 \\ s.t. \quad & h(x) = x_1^2 + x_2^2 - 9 \\ & g_1(x) = -(x_1 + x_2^2) + 1 \geq 0 \\ & g_2(x) = -(x_1 + x_2) + 1 \geq 0 \end{aligned}$$

Solution: It is easy to show that at $x = x^*$, $h(x) = 0$ and $g_1(x) = 0$. According to the Kuhn-Tucker condition, we first examine if the gradients of the

saturated constraints are linearly independent. Let

$$c_1 \nabla h(x^*) + c_2 \nabla g_1(x^*) = 0$$

or

$$c_1 \begin{pmatrix} 2x_1^* \\ 2x_2^* \end{pmatrix} + c_2 \begin{pmatrix} -1 \\ -2x_2^* \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

The above will be true only if c_1 and c_2 are both zero. Therefore, the gradients are indeed linearly independent.

The Kuhn-Tucker condition is shown as follows. First, we check if all the constraints are satisfied.

$$h(x^*) = (x_1^*)^2 + (x_2^*)^2 - 9 = (-2.37)^2 + (-1.84)^2 - 9 = 0$$
$$g_1(x^*) = -[x_1^* + (x_2^*)^2] + 1 = -[(-2.37) + (-1.84)^2] + 1 = 0$$
$$g_2(x^*) = -[x_1^* + x_2^*] + 1 = -[(-2.37) - 1.84] + 1 = 5.21 > 0$$

Now, we check the other part of the Kuhn-Tucker condition.

$$\nabla f(x^*) - \lambda^* \nabla h(x^*) - \sum_{j=1}^{2} \mu_j^* \nabla g_j(x^*) = 0$$
$$\begin{pmatrix} 2x_1^* \\ 1 \end{pmatrix} - \lambda^* \begin{pmatrix} 2x_1^* \\ 2x_2^* \end{pmatrix} - \mu_1^* \begin{pmatrix} -1 \\ -2x_2^* \end{pmatrix} - \mu_2^* \begin{pmatrix} -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$
$$\mu_1^*(1 - x_1^* - (x_2^*)^2) = 0$$
$$\mu_2^*(1 - x_1^* - x_2^*) = 0$$
$$\mu_1^* \geq 0 \quad \text{and} \quad \mu_2^* \geq 0$$

Solving the above equations results in

$$\mu_1^* = 1.05, \quad \mu_2^* = 0, \quad \lambda^* = 0.778$$

which shows that the Kuhn-Tucker condition is met.

Finally, the second order sufficient condition is given as follows. Since

$$\nabla^2 f(x^*) = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}, \quad \nabla^2 h(x^*) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$
$$\nabla^2 g_1(x^*) = \begin{pmatrix} 0 & 0 \\ 0 & -2 \end{pmatrix}, \quad \nabla^2 g_2(x^*) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

we have

$$\begin{aligned}
&(z_1 \ z_2) \left[\begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix} - \lambda^* \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} - \mu_1^* \begin{pmatrix} 0 & 0 \\ 0 & -2 \end{pmatrix} \right] \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \\
&= (z_1 \ z_2) \left[\begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix} - 0.778 \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} - 1.05 \begin{pmatrix} 0 & 0 \\ 0 & -2 \end{pmatrix} \right] \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \\
&= (z_1 \ z_2) \begin{pmatrix} 0.444 & 0 \\ 0 & 0.544 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \\
&= 0.444 z_1^2 + 0.544 z_2^2
\end{aligned}$$

It is clear that for any $z = (z_1, z_2) \neq 0$ the value of the above equation is always larger than zero. Therefore, $x^* = (-2.37, -1.84)$ is a minimum.

Most of the constrained nonlinear programming methods can be divided into two families: primal methods and dual methods. The former methods operate directly with the given problem (or primal problem) and generate a sequence of solutions that satisfy the constraints and ensure the decrease of the objective function. Their advantages include that if the iterative process is not interrupted, then they can generate a feasible solution. However, they are generally difficult to implement and their convergence is not guaranteed. In the next section, we introduce two primal methods: sequential linear programming and sequential quadratic programming, which are implemented in some computer software packages.

The dual methods are easier to implement and more robust. Their convergence can be more easily obtained. However, they only produce a feasible solution at the end of the iterative process. In the last section of this chapter, we discuss three widely used dual methods: exterior penalty method, interior penalty method and Lagrangean method.

3.3 Primal Methods

3.3.1 Sequential Linear Programming

Sequential linear programming (SLP) is developed due to the success of the Simplex method for solving linear programming. Its general principle consists of approximating a nonlinear programme by a sequence of linear programmes using the first order Taylor expansion of each nonlinear constraint (or objective) function (Sen and Yang, 1998).

To solve a nonlinear optimisation problem, SLP consists of linearising its nonlinear objective function and nonlinear constraint functions at a given solution. Around this solution, a search space is established using the given step sizes of variables. If the intersection of the linearised feasible space and the established search space is empty, the search space will be expanded by increasing the step sizes. If the intersection is not empty, SLP will search for a solution that optimises the linearised objective function within a new feasible space defined by the intersection. The obtained optimal solution is then used to re-linearise the original problem. The process is repeated until the optimum (or its approximation) of the original nonlinear problem is found.

Generally, this method does not converge without online regulation of the step sizes of variables. Figure 3.2 (Minoux, 1986) shows why this may happen. The feasible space of the example shown in Figure 3.2 is a convex polyhedron and the lines of constant objective values are concentric circles. $\bar{f}(x)$ is the tangent line of $f(x)$ at the point x. Suppose x^0 is the starting solution and we obtain x^1 and x^2 successively. Then, the solution process indefinitely oscillates

at these last two points (x^1 and x^2). The process does not converge due to the oscillation.

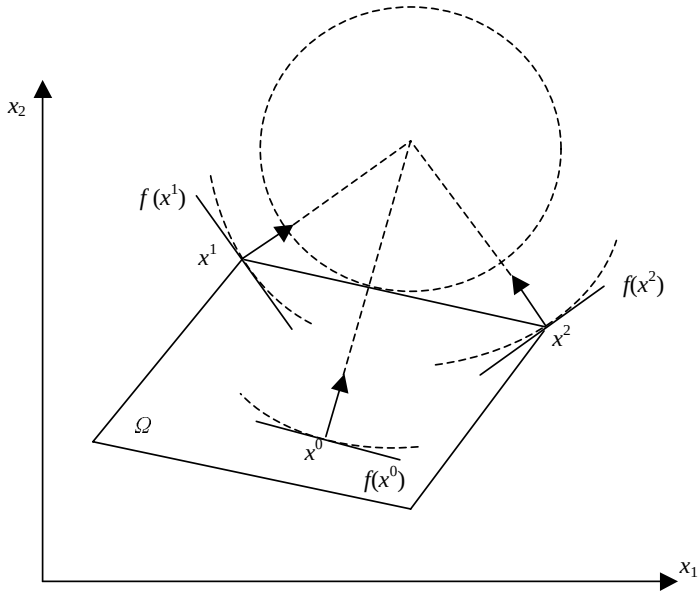

Figure 3.2: Oscillation in SLP

One of the techniques for solving the oscillation problem is to reduce the step sizes of variables once oscillation is detected. In Figure 3.3, for instance, a nonlinear function y is linearised at an initial solution x^0. A search space (a line segment in this case) around x^0 is established using the initial step size t^0. The maximum of the linearised function (the tangent line at x^0) is sought within the established line segment, which is x^1. Repeating the same process with the fixed step size t^0, we can generate x^2, x^3 and x^4 successively. Since $x^4 = x^2$, the first oscillation occurs. If the step size is reduced so that $t^1 = t^0/2$, for example, then we can find x^5 and x^6 successively. As $x^6 = x^4$, the step size need to be further reduced with for example $t^2 = t^1/2$. We then get x^7 which is quite close to the maximum of $y(x)$.

It should be noted that for multivariable problems oscillation may occur between the current solution and a solution of a previously linearised problem (not the most recent one). It is therefore necessary to record previously generated solutions for checking such oscillation. How many previous solutions should be recorded depends on the complexity of a problem in question.

The computational steps of SLP may be summarised below.

S1 Define a nonlinear single-objective optimisation problem as follows

$$\text{Min} \qquad f(x) \tag{3.3.1}$$

$$s.t. \qquad x \in \Omega \tag{3.3.2}$$

where

$$\Omega = \left\{ x \;\middle|\; \begin{array}{l} h_i(x) = 0, i = 1, 2, ..., m \\ g_j(x) \geq 0, j = 1, 2, ..., l \end{array} \right\} \tag{3.3.3}$$

S2 Select an initial solution x^0 that may be feasible or unfeasible. Initialise the step sizes of variables denoted as Δx_i^0 $(i = 1, 2, ..., n)$. Let $t = 0$.

S3 Calculate the first order Taylor expansion of the objective function at the current solution x^t

$$\bar{f}(x) = f(x^t) + \nabla f(x^t)(x - x^t) \tag{3.3.4}$$

Linearise the nonlinear constraint functions $h_i(x)$ $(i = 1, 2, ..., m)$ and $g_j(x)$ $(j = 1, 2, ..., l)$ at x^t using their first order Taylor expansions

$$\bar{h}_i(x) = h_i(x^t) + \nabla h_i(x^t)(x - x^t) \tag{3.3.5}$$

$$\bar{g}_j(x) = g_j(x^t) + \nabla g_j(x^t)(x - x^t) \tag{3.3.6}$$

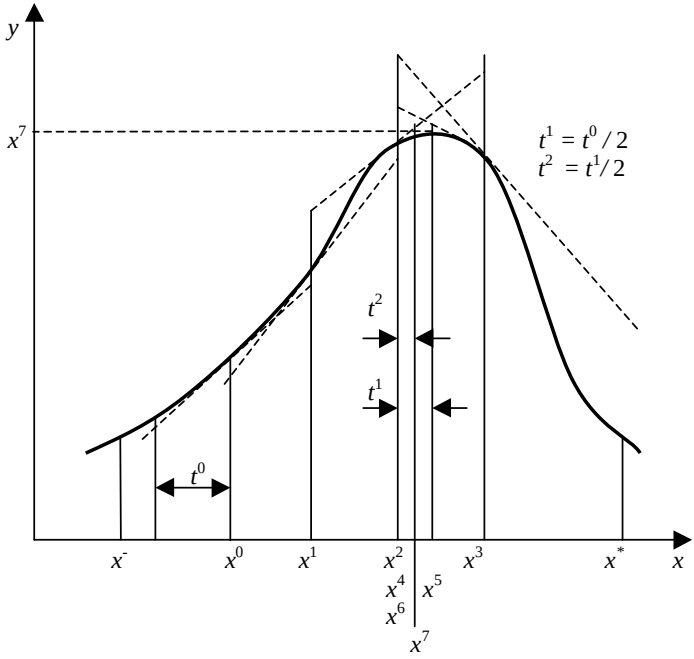

Figure 3.3: SLP searching process

S4 In a neighbourhood of x^t, define a search space S^t using the step sizes of all variables as follows

$$S^t = \left\{ x \;\middle|\; \begin{array}{l} x_i^t - \Delta x_i^t \le x_i \le x_i^t + \Delta x_i^t, i = 1, 2, ..., n \\ x = [x_1, x_2, ..., x_n]^T \end{array} \right\} \tag{3.3.7}$$

S5 Formulate the following linear programming problem

$$\text{Min} \quad \bar{f}(x) \tag{3.3.8}$$

$$s.t. \quad S^t \cap \Omega^t \tag{3.3.9}$$

where

$$\Omega^t = \left\{ x \;\middle|\; \begin{array}{l} \bar{h}_i(x) = 0, i = 1, 2, ..., m \\ \bar{g}_j(x) \ge 0, j = 1, 2, ..., l \end{array} \right\} \tag{3.3.10}$$

S6 If Ω^t is empty, we must select a new solution and go to S3 to repeat the process. It is easy to show that Ω^t will not be empty if x^t is feasible.

S7 If $S^t \cap \Omega^t$ is empty, we need to increase the step sizes Δx_i^t to expand the search space S^t until $S^t \cap \Omega^t$ becomes not empty.

S8 Find the optimal solution x^{t+1} of the constructed linear programme (3.3.9) using, for example, the Simplex method.

S9 If x^{t+1} is unfeasible with regard to the original nonlinear problem, let $t = t+1$ and go to S3 to construct a new linear programming problem in a neighbourhood of x^{t+1}. If x^{t+1} is the same as one that was generated before, then, reduce the step sizes of variables and go to S3.

S10 If x^{t+1} is also feasible with regard to the original nonlinear problem, then x^{t+1} will be taken as the approximate optimal solution of the original problem and the iteration process will be ended if:
a) the step sizes of all variables have been reduced to be significantly small, and
b) the values of both the variables and the objective function are not significantly different in two successive iterations.
Otherwise, let $t = t + 1$ and go to S3 to construct a new linear problem in a neighbourhood of x^{t+1}.

Note that the above algorithm does not guarantee to generate a feasible or an optimal solution for a strongly nonlinear problem. It can provide an approximate solution for a nonlinear programming problem with a nonlinear objective function and linear (or mainly linear) constraints.

3.3.2 Sequential Quadratic Programming

This method is based on quadratic programming. A nonlinear programme is called a quadratic programme if its objective function is a quadratic function of variables and all the constraint functions are linear functions of variables.

Sequential quadratic programming consists of replacing the solution of a nonlinear programme by the solution of a sequence of quadratic programmes that approximate the given problem in a sense that the objective function is approximated by its second order Taylor expansion and each nonlinear constraint function is approximated by its first order Taylor expansion.

Before introducing sequential quadratic programming, we first discuss quadratic programming. In general, the mathematical model of a quadratic programme can be represented as follows.

$$\text{Min} \qquad f(x) = \sum_{j=1}^{n} c_j x_j + \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{jk} x_j x_k$$

$$c_{jk} = c_{kj}, \quad k = 1, 2, ..., n \tag{3.3.11}$$

$$s.t. \qquad \sum_{j=1}^{n} a_{ij} x_j + b_i \geq 0, \quad i = 12, ..., m \tag{3.3.12}$$

$$x_j \geq 0, \quad j = 1, 2, ..., n \tag{3.3.13}$$

where the second term of the objective function is quadratic.

If the quadratic objective function is positive definite (or semi-positive definite), then the function is strictly convex. Since the feasible solution space of a quadratic problem is a convex set, a quadratic programme is a convex programme. For this type of problems, a local optimum is also a global optimum and the Kuhn-Tucker condition is not only necessary but also sufficient.

Applying the Kuhn-Tucker conditions (3.2.4) to (3.2.6) to the quadratic programme (3.3.11) to (3.3.13) and replacing μ by y, we have

$$-\sum_{k=1}^{n} c_{jk} x_k + \sum_{i=1}^{m} a_{ij} y_{n+i} + y_j = c_j, \quad j = 1, 2, ..., n \tag{3.3.14}$$

$$\left(\sum_{j=1}^{n} a_{ij} x_j + b_i \right) y_{n+i} = 0, \quad i = 1, 2, ..., m \tag{3.3.15}$$

$$x_j y_j = 0, \quad j = 1, 2, ..., n \tag{3.3.16}$$

Adding surplus variables in (3.3.12) leads to

$$\sum_{j=1}^{n} a_{ij} x_j - x_{n+i} + b_i = 0, \quad i = 1, 2, ..., m \tag{3.3.17}$$

Multiplying (3.3.17) by y_{n+i} and considering (3.3.15) and (3.3.16) results in

$$x_j y_j = 0, \quad j = 1, 2, ..., n + m \tag{3.3.18}$$

Note that

$$x_j \geq 0, \quad y_j \geq 0, \quad j = 1, 2, ..., n + m \tag{3.3.19}$$

If a solution of (3.3.14) and (3.3.17) also satisfies (3.3.18) and (3.3.19), it is the solution of the original quadratic programme. In (3.3.14), the parameter c_j could be positive or negative. Introduce an artificial variable $z_j (z_j \geq 0)$ so that (3.3.14) can be re-written as follows.

$$-\sum_{k=1}^{n} c_{jk} x_k + \sum_{i=1}^{m} a_{ij} y_{n+i} + y_j + \text{sgn}(c_j) z_j = c_j, \quad j = 1, 2, ..., n \quad (3.3.20)$$

where $\text{sgn}(c_j) = 1$ for $c_j \geq 0$ and $\text{sgn}(c_j) = -1$ for $c_j < 0$.

From the above discussion, we can find the following basic solution

$$z_j = \text{sgn}(c_j) c_j, \quad j = 1, 2, ..., n \quad (3.3.21)$$

$$x_{n+i} = b_i, \quad i = 1, 2, ..., m \quad (3.3.22)$$

$$x_j = 0, \quad j = 1, 2, ..., n \quad (3.3.23)$$

$$y_j = 0, \quad j = 1, 2, ..., n + m \quad (3.3.24)$$

The above solution is not an optimum of the original quadratic unless $z_j = 0$. We therefore formulate the following linear programming problem

$$\text{Min} \quad \varphi(z) = \sum_{j=1}^{n} z_j \quad (3.3.25)$$

$$s.t. \quad -\sum_{k=1}^{n} c_{jk} x_k + \sum_{i=1}^{m} a_{ij} y_{n+i} + y_j + \text{sgn}(c_j) z_j = c_j,$$

$$j = 1, 2, ..., n \quad (3.3.26)$$

$$\sum_{j=1}^{n} a_{ij} x_j - x_{n+i} + b_i = 0, \quad i = 1, 2, ..., m \quad (3.3.27)$$

$$x_j \geq 0, \quad j = 1, 2, ..., n + m \quad (3.3.28)$$

$$y_j \geq 0, \quad j = 1, 2, ..., n + m \quad (3.3.29)$$

$$z_j \geq 0, \quad j = 1, 2, ..., n \quad (3.3.30)$$

The solution of the above linear programme should also satisfy (3.3.18). This means that for each j, x_j and y_j must not be basic variables simultaneously. If the optimal solution of problem (3.3.25) is $(x_1^*, x_2^*, ..., x_{n+m}^*, y_1^*, y_2^*, ..., y_{n+m}^*, z_1^* = 0, z_2^* = 0, ..., z_n^* = 0)$, then $(x_1^*, x_2^*, ..., x_n^*)$ is the optimal solution of the original quadratic problem.

Example 3.2 Solve the following quadratic programme (Li and Qian, 1982)

$$\begin{aligned} \text{Min} \quad & f(x) = 8x_1 + 10x_2 - x_1^2 - x_2^2 \\ s.t. \quad & 3x_1 + 2x_2 \leq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Solution: The above quadratic programme is equivalent to the following problem

$$\begin{array}{ll} \text{Min} & \bar{f}(x) = -8x_1 - 10x_2 + \frac{1}{2}(2x_1^2 + 2x_2^2) \\ s.t. & 6 - 3x_1 - 2x_2 \geq 0 \\ & x_1, x_2 \geq 0 \end{array}$$

In comparison with (3.3.11) to (3.3.13), we have

$$\begin{aligned} &c_1 = -8, \quad c_2 = -10, \quad c_{11} = 2, \quad c_{22} = 2 \\ &c_{12} = C_{21} = 0, \quad b_1 = 6, \quad a_{11} = -3, \quad a_{12} = -2 \end{aligned}$$

Since c_1 and c_2 are negative, we set $\text{sgn}(c_1) = -1$ and $\text{sgn}(c_2) = -1$. Following (3.3.25), we get the following linear programme

$$\begin{array}{ll} \text{Min} & \varphi(z) = z_1 + z_2 \\ s.t. & -3y_3 + y_1 - 2x_1 - z_1 = -8 \\ & -2y_3 + y_2 - 2x_2 - z_2 = -10 \\ & -3x_1 - 2x_2 - x_3 + 6 = 0 \\ & x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2 \geq 0 \end{array}$$

or

$$\begin{array}{ll} \text{Min} & \varphi(z) = z_1 + z_2 \\ s.t. & 2x_1 + 3y_3 - y_1 + z_1 = 8 \\ & 2x_2 + 2y_3 - y_2 + z_2 = 10 \\ & 3x_1 + 2x_2 + x_3 = 6 \\ & x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2 \geq 0 \end{array}$$

In addition, the following equation should be satisfied

$$x_i y_i = 0, \quad j = 1, 2, 3$$

Using the simplex method to solve the above problem results in the following solution

$$\begin{aligned} &x_1 = \frac{4}{13}, \quad x_2 = \frac{33}{13}, \quad x_3 = 0, \quad y_1 = 0 \\ &y_2 = 0, \quad y_3 = \frac{32}{13}, \quad z_1 = 0, \quad z_2 = 0 \end{aligned}$$

This means that the optimal solution of the original problem is given by

$$x_1^* = \frac{4}{13}, \quad x_2^* = \frac{33}{13}, \quad f(x) = 21.3$$

It can be shown that the Kuhn-Tucker condition is satisfied at the above solution.

To solve a general nonlinear programme as defined in (3.1.1) to (3.1.3), sequential quadratic programming consists of approximating the original problem in a neighbourhood of a given solution using the second order Taylor series of its objective function and the first order Taylor series of its nonlinear constraint functions. This will generate a quadratic programming problem, which can be solved using the technique as discussed above. If the optimal solution of the quadratic problem is not a minimum of the original problem, we can construct a new quadratic problem by approximating the original problem in a neighbourhood of the current solution. This will result in a new solution. The process is repeated until the minimum of the original problem is found.

The computational steps of sequential quadratic programming are summarised as follows:

S1 Define a nonlinear single-objective optimisation problem using the following format

$$\text{Min} \quad f(x) \tag{3.3.31}$$

$$s.t. \quad x \in \Omega \tag{3.3.32}$$

where

$$\Omega = \left\{ x \;\middle|\; \begin{array}{l} h_i(x) = 0, i = 1, 2, ..., m \\ g_j(x) \geq 0, j = 1, 2, ..., l \end{array} \right\} \tag{3.3.33}$$

S2 Select an initial solution x^0 that may be feasible or unfeasible. Initialise the step sizes of variables denoted as Δx_i^0 $(i = 1, ..., n)$. Let $t = 0$.

S3 Calculate the second order Taylor expansion of the objective function at the solution

$$\bar{f}(x) = f(x^t) + \nabla f(x^t)(x - x^t) + (x - x^t)^T \nabla^2 f(x^t)(x - x^t) \tag{3.3.34}$$

Linearise the nonlinear constraint functions $h_i(x)$ $(i = 1, 2, ..., m)$ and $g_j(x)$ $(j = 1, 2, ..., l)$ at the solution x^t using their first order Taylor expansions

$$\bar{h}_i(x) = h_i(x^t) + \nabla h_i(x^t)(x - x^t) \tag{3.3.35}$$

$$\bar{g}_j(x) = g_j(x^t) + \nabla g_j(x^t)(x - x^t) \tag{3.3.36}$$

S4 In a neighbourhood of the current solution x^t, use the given step sizes to define a search space S^t

$$S^t = \left\{ x \;\middle|\; \begin{array}{l} x_i^t - \Delta x_i^t \leq x_i \leq x_i^t + \Delta x_i^t, i = 1, 2, ..., n \\ x = [x_1, x_2, ..., x_n]^T \end{array} \right\} \tag{3.3.37}$$

S5 Formulate the following quadratic problem

$$\text{Min} \quad \bar{f}(x) \tag{3.3.38}$$
$$s.t. \quad S^t \cap \Omega^t \tag{3.3.39}$$

where

$$\Omega^t = \left\{ x \;\middle|\; \begin{array}{l} \bar{h}_i(x) = 0, i = 1, 2, ..., m \\ \bar{g}_j(x) \geq 0, j = 1, 2, ..., l \end{array} \right\} \tag{3.3.40}$$

S6 If Ω^t is empty, we must select a new solution and go to S3 to repeat the process. One can see that Ω^t will not be empty if x^t is feasible.

S7 If $S^t \cap \Omega^t$ is empty, we need to increase the step sizes Δx_i^t to expand the search space S^t until $S^t \cap \Omega^t$ becomes not empty.

S8 Find the optimal solution x^{t+1} of quadratic programme (3.3.39) using, for example, the technique described earlier in this section.

S9 If x^{t+1} is unfeasible with regard to the original nonlinear problem, let $t = t + 1$ and go to S3 to construct a new quadratic problem in a neighbourhood of x^{t+1}.

S10 If x^{t+1} is also feasible with regard to the original nonlinear problem, then x^{t+1} will be taken as the approximate optimal solution of the original problem and the iteration process will be ended if either:
a) the step sizes of all variables have been reduced to be significantly small, or
b) both the values of the variables and the objective function are not significantly different in two successive iterations.
Otherwise, let $t = t + 1$ and go to S3 to construct a new quadratic problem in a neighbourhood of x^{t+1}.

The algorithm provides an alternative way of solving nonlinear programming problems, which have nonlinear objective functions and linear (or mostly linear) constraint functions. However, one should note that it does not guarantee to generate a feasible or an optimal solution for a strongly nonlinear problem.

3.4 Dual Methods

Sequential linear or quadratic programming is based on the approximation of a nonlinear programming problem by a sequence of linear or quadratic programming problems, which can be solved using, for example, the Simplex method. In this section, we discuss three dual methods that differ significantly from primal methods. The common principle of these dual methods is to reduce the original problem to the solution of a sequence of unconstrained optimisation problems.

3.4.1 Lagrangean Methods

Lagrangean methods are developed on the basis of duality theory. Consider a mathematical programming problem as shown by (3.2.17) to (3.2.19). The Lagrange function of the problem is given by

$$L(x,\lambda) = f(x) + \sum_{j=1}^{l} \lambda_j g_j(x) \tag{3.4.1}$$

In section 3.2.3 of this chapter, we mentioned that the problem ((3.2.17) to (3.2.19)) can be solved if we can find a saddle point of the Lagrange function, which is a pair $(\bar{x}, \bar{\lambda})$ satisfying the following conditions

$$L(\bar{x},\bar{\lambda}) = \min_{x\in S} L(x,\bar{\lambda}) \tag{3.4.2}$$

$$g(\bar{x}) \leq 0 \tag{3.4.3}$$

$$\bar{\lambda}_i g_i(\bar{x}) = 0 \tag{3.4.4}$$

A saddle point can be found by solving a dual problem. First, we define a dual function for $\lambda \geq 0$, as follows

$$w(\lambda) = L(\bar{x},\lambda) = \min_{x\in S} L(x,\lambda) \tag{3.4.5}$$

In other words, at any $\lambda \geq 0$ the dual function $w(\lambda)$ takes the minimum value of the Lagrange function in the feasible decision space. The dual problem of the original (or primal) problem as defined in (3.2.17) to (3.2.19) can be defined as follows

$$\max_{\lambda} w(\lambda) = \max_{\lambda}\{\min_{x\in S} L(x,\lambda)\}, \qquad \text{for} \qquad \lambda \geq 0 \tag{3.4.6}$$

The dual problem has several important properties. First, for all $\lambda \geq 0$ the value of the dual function $w(\lambda)$ is a lower bound of the optimal objective value $f(\lambda^*)$ of the primal problem. In other words, if $w(\lambda^*)$ is the optimal value of the dual problem, then we have

$$w(\lambda) \leq w(\lambda^*) \leq f(\lambda^*), \ \forall \lambda \geq 0 \tag{3.4.7}$$

Another property of the dual problem is that the dual function $w(\lambda)$ is a concave function of λ. This is a very general property and does not require any assumption about convexity of the objective function f or the constraint functions $g_i(x)$, or about convexity of the decision space S. Even in situations where the dual function $w(\lambda)$ is not differentiable at every point (for example S is a discrete set), the concavity of $w(\lambda)$ ensures that a local optimum λ^0 is also a global optimum. Therefore, in general the dual problem can be more easily solved than the primal problem.

The duality theory states that the optimal value of the primal problem is equal to the optimal value of the dual problem if the primal problem has a saddle point $(\bar{x}, \bar{\lambda})$, or

$$w(\bar{\lambda}) = f(\bar{x}) \tag{3.4.8}$$

In other words, if a saddle point exists, the solution of the dual problem will lead to the generation of an optimal solution of the primal problem.

Example 3.3 Find the saddle point of the following problem.

$$\begin{array}{ll} \text{Min} & x_1^2 + 2x_2^2 \\ s.t. & x_1 + 3x_2 + 11 \leq 0 \end{array}$$

Solution: The objective function $f(x)$ and the constrained function $g(x)$ are written by

$$\begin{aligned} f(x) &= x_1^2 + 2x_2^2 \\ g(x) &= x_1 + 3x_2 + 11 \end{aligned}$$

So, $f(x)$ and $g(x)$ are both convex, which means that the problem has a saddle point. The Lagrange function of the problem is given by

$$L(x, \lambda) = x_1^2 + 2x_2^2 + \lambda x_1 + 3\lambda x_2 + 11\lambda$$

The minimum of $L(x, \lambda)$ in x can be found by formulating the following necessary conditions

$$\begin{aligned} \frac{\partial L(x, \lambda)}{\partial x_1} &= 2x_1 + \lambda = 0 \\ \frac{\partial L(x, \lambda)}{\partial x_2} &= 4x_2 + 3\lambda = 0 \end{aligned}$$

From the above conditions, we have

$$x_1 = -\frac{\lambda}{2} \quad \text{and} \quad x_2 = -\frac{3\lambda}{4}$$

The dual function is therefore given by

$$w(\lambda) = L(\bar{x}, \lambda) = -\frac{11}{8}\lambda^2 + 11\lambda$$

The application of the necessary condition for maximising $w(\lambda)$ leads to the following equation

$$\frac{\partial w(\lambda)}{\partial \lambda} = -\frac{11}{4}\lambda + 11 = 0$$

So

$$\bar{\lambda} = 4, \bar{x}_1 = -2 \quad \text{and} \quad \bar{x}_2 = -3$$

Note that

$$f(\bar{x}) = \bar{x}_1^2 + 2\bar{x}_2^2 = (-2)^2 + 2(-3)^2 = 22$$
$$w(\bar{\lambda}) = -\frac{11}{8} \times 4^2 + 11 \times 4 = 22$$

So $f(\bar{x}) = w(\bar{\lambda})$, as expected.

In the above example, the minimum of the Lagrange function in x was obtained as an explicit function of λ. This is in general difficult to achieve. In the dual problem, if $\bar{\lambda}$ is known, which maximises the dual function, then we can obtain $\bar{x}$ by minimising $L(x, \lambda)$ by means of unconstrained optimisation. Since $\bar{\lambda}$ is in general not known in advance, it is often generated using iterative methods. The principle of such methods is to produce a sequence of λ^k with $\lambda^k \to \bar{\lambda}$. For each λ^k generated, a corresponding solution x^k will be generated by minimising $L(x, \lambda^k)$ using an unconstrained optimisation method. The iterative methods operate in such a way that the sequence of the corresponding solutions x^k will converge to $\bar{x}$. This way the solution of the original constrained problem is replaced by the solution of a sequence of unconstrained optimisation problems.

One of the main iterative methods, called the method of Uzawa (Minoux, 1986), is to use a classical gradient method to solve the dual problem. This method consists of a two-stage process. At the first stage, the Lagrange multiplier vector λ^k is updated using the gradient of the Lagrange function. At the second stage, the Lagrange function is minimised for an updated λ^k. The method can be summarised by the following steps:

S1 Select a starting Lagrange multiplier $\lambda^0 \geq 0$ and let $k = 0$.

S2 At iteration k, the Lagrange multiplier λ^k is given. Compute the value of the dual function at λ^k by minimising the Lagrange function over the decision space, or

$$\begin{aligned} w(\lambda^k) &= \min_{x \in S} \{f(x) + \lambda^k g(x)\} \\ &= f(x^k) + \lambda^k g(x^k) \end{aligned} \tag{3.4.9}$$

S3 $g(x^k)$ is the gradient of $w(\lambda^k)$ at λ^k. Update λ as follows

$$\lambda^{k+1} = \max\{0, \lambda^k + \tau_k g(x^k)\} \tag{3.4.10}$$

where $\tau_k \geq 0$ is the step size at iteration k, which needs to be adjusted to ensure that $w(\lambda^{k+1}) \geq w(\lambda^k)$.

S4 Check if the stopping criterion is satisfied, which could be

$$\left|w(\lambda^{k+1}) - w(\lambda^k)\right| \leq \delta \tag{3.4.11}$$

where δ is a very small positive real number. If this is the case, terminate the iteration. Otherwise, let $k = k + 1$ and go to S2.

In the above method, the step sizes can be chosen in advance, leading to a gradient method with fixed steps. The steps can also be determined using one-dimensional search by minimising $w(\lambda^k)$ along the direction λ^k, leading to the steepest descent method.

It is clear from (3.4.10) that if a constraint $g_j(x)$ is not satisfied at x^k, or $g_j(x^k) > 0$, then the corresponding Lagrange multiplier λ_j will increase. This will increase the penalty on the violated constraint as shown in the second term in (3.4.9). On the other hand, if $g_j(x)$ is satisfied at x^k, or $g_j(x^k) \leq 0$, then the corresponding Lagrange multiplier λ_j will decrease towards the lower bound of zero.

3.4.2 Method of Exterior Penalties

Similar to Lagrangean methods, the principle of penalty methods is to replace an original constrained minimisation problem by a sequence of unconstrained problems. They are therefore referred to as sequential unconstrained minimisation techniques (or SUMT). The exterior penalty method consists of constructing a penalty function and searching for a sequence of solutions that approach an optimum outside of the feasible space.

Define an optimisation problem as shown in (3.2.17) to (3.2.19). Construct the following penalty function

$$P(x, \tau_k) = f(x) + \tau_k \sum_{j=1}^{l} [g_j(x)]^2 u_j(g_j) \tag{3.4.12}$$

where τ_k is called the penalty coefficient at the kth iteration and $\tau_{k+1} > \tau_k$. $\mu_j(g_j)$ is the following function.

$$u_j(g_j) = \begin{cases} 0 & \text{if} \quad g_j(x) \leq 0 \\ 1 & \text{if} \quad g_j(x) > 0 \end{cases} \tag{3.4.13}$$

Equation (3.4.12) can also be represented using the following equivalent form

$$P(x, \tau_k) = f(x) + \tau_k \sum_{j=1}^{l} \{\max[g_j(x), 0]\}^2 \tag{3.4.14}$$

where

$$\max[g_j(x),\ 0] = \begin{cases} 0 & \text{if} \quad g_j(x) \leq 0 \\ g_j(x) & \text{if} \quad g_j(x) > 0 \end{cases} \tag{3.4.15}$$

For any $\tau_k > 0$, a minimum of the penalty function $P(x, \tau_k)$ is denoted by $\bar{x}(\tau_k)$. The penalty coefficient τ_k should be properly chosen. On the one hand, it has to be large enough for $\bar{x}(\tau_k)$ to be feasible or close to the feasible solution space. On the other hand, if τ_k is too large, the penalty function could be ill-conditioned, leading to numerical difficulties in the search for the optimum of the penalty function.

The above analysis explains why the exterior penalty method is normally implemented in an iterative manner. Initially, choose a relatively small penalty coefficient τ_1, and then solve the unconstrained problem

$$\min_x P(x, \tau_1) = f(x) + \tau_1 \sum_{j=1}^{l} \{\max[g_j(x), 0]\}^2 \tag{3.4.16}$$

Suppose $\bar{x}(\tau_1)$ is the optimum of problem (3.4.16). If the second term of the problem is sufficiently small, then $\bar{x}(\tau_1)$ is a good approximation of the optimum of the original problem. Otherwise, choose a new penalty coefficient $\tau_2 > \tau_1$ and solve the following new unconstrained problem

$$\min_x P(x, \tau_2) = f(x) + \tau_2 \sum_{j=1}^{l} \{\max[g_j(x), 0]\}^2 \tag{3.4.17}$$

We then obtain a new solution $\bar{x}(\tau_1)$. The exterior penalty method converges to an optimal solution of the original problem when the penalty coefficient τ_k tends to ∞. The computational steps of the exterior penalty method can be summarised as follows:

S1 Select an initial solution $\bar{x}^0$ and set the first penalty coefficient τ_1. Let $k = 1$ and $\tau_{k+1}/\tau_k = c$ with $c > 1$.

S2 Use $\bar{x}^{k-1}$ as the starting point to solve the unconstrained problem (3.4.14), resulting in a new solution $\bar{x}^k$.

S3 If all constraints are satisfied at $\bar{x}^k$, then $\bar{x}^k$ is a feasible solution and it is also optimal for the original problem. The iteration process terminates. Otherwise, go to the next step.

S4 Update the penalty coefficient, for example $\tau_{k+1} = 10\tau_k$, and go to S2.

Example 3.4 Solve the following programming problem (Cai, 1982)

$$\begin{aligned} \text{Min} \quad & f(x) = x_1^2 + x_2^2 \\ s.t. \quad & 3x_1 + 2x_2 - 6 \geq 0 \\ & x_1 \geq 0, \quad x_2 \geq 0 \end{aligned}$$

Solution: Re-write the problem as follows

$$\text{Min} \qquad f(x) = x_1^2 + x_2^2$$

$$\begin{aligned} s.t. \quad & g_1(x) = -3x_1 - 2x_2 + 6 \leq 0 \\ & g_2(x) = -x_1 \leq 0 \\ & g_3(x) = -x_2 \leq 0 \end{aligned}$$

First, we use the Kuhn-Tucker conditions to solve the problem. The Lagrange function of the above problem is given as follows.

$$L(x, \lambda) = x_1^2 + x_2^2 + \lambda_1(-3x_1 - 2x_2 + 6) + \lambda_2(-x_1) + \lambda_3(-x_2)$$

The Kuhn-Tucker conditions of the Lagrange function read

$$\begin{aligned} \frac{\partial L(x,\lambda)}{\partial x_1} &= 2x_1 - 3\lambda_1 - \lambda_2 = 0 \\ \frac{\partial L(x,\lambda)}{\partial x_2} &= 2x_2 - 2\lambda_1 - \lambda_3 = 0 \\ \frac{\partial L(x,\lambda)}{\partial \lambda_1} &= -3x_1 - 2x_2 + 6 \geq 0 \\ \frac{\partial L(x,\lambda)}{\partial \lambda_2} &= x_1 \geq 0 \\ \frac{\partial L(x,\lambda)}{\partial \lambda_3} &= x_2 \geq 0 \\ \lambda_1(-3x_1 - 2x_2 + 6) &= 0 \\ \lambda_2(-x_1) &= 0 \\ \lambda_3(-x_2) &= 0 \end{aligned}$$

Solving the above equations, we get

$$\begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} = \begin{bmatrix} {}^{18}/_{13} \\ {}^{12}/_{13} \end{bmatrix}, \quad \begin{bmatrix} \bar{\lambda}_1 \\ \bar{\lambda}_2 \\ \bar{\lambda}_3 \end{bmatrix} = \begin{bmatrix} {}^{12}/_{13} \\ 0 \\ 0 \end{bmatrix}$$

Now, we use the exterior penalty method to solve the problem. First, construct the following penalty function

$$\begin{aligned} P(x, \tau_k) \quad = \quad & x_1^2 + x_2^2 + \tau_{k1}(-3x_1 - 2x_2 + 6)^2 u_1(g_1) \\ & + \tau_{k2}(-x_1)^2 u_2(g_2) + \tau_{kk3}(-x_2)^2 u_3(g_3) \end{aligned}$$

Suppose x is within the first quadrant. Then, the constraints $x_1 \geq 0$ and $x_2 \geq 0$ can be satisfied, resulting in $u_2(g_2) = 0$ and $u_3(g_3) = 0$. Since the constraint $g_1 \leq 0$ is not satisfied, we set $u_1(g_1) = 1$. Let $\tau_k = \tau_{k1}$.

Use the necessary conditions to find the optimal solution of the penalty function.

$$\begin{aligned} \frac{\partial P}{\partial x_1} &= 2x_1 - 6\tau_k(-3x_1 - 2x_2 + 6) = 0 \\ \frac{\partial P}{\partial x_2} &= 2x_2 - 4\tau_k(-3x_1 - 2x_2 + 6) = 0 \end{aligned}$$

Solving the above equations leads to

$$\bar{x}_1(\tau_k) = \frac{36\tau_k}{2 + 26\tau_k}, \quad \bar{x}_2(\tau_k) = \frac{24\tau_k}{2 + 26\tau_k}$$

For $\tau_k \to \infty$, we have

$$\bar{x}_1 = \frac{18}{13}, \quad \bar{x}_2 = \frac{12}{13}$$

This is the same solution as we obtained using the Kuhn-Tucker conditions.

Example 3.5 Use the exterior penalty method to solve the following problem (Cai, 1982).

$$\begin{aligned} \text{Min} \quad & f(x) = \frac{1}{3}(x_1 + 1)^3 + x_2 \\ s.t. \quad & g_1(x) = 1 - x_1 \leq 0 \\ & g_2(x) = -x_2 \leq 0 \end{aligned}$$

Solution: The penalty function of the above problem is given by

$$\begin{aligned} P(x, \tau_k) \quad = \quad & \frac{1}{3}(x_1 + 1)^3 + x_2 + \tau_k \{\max[0, 1 - x_1]\}^2 \\ & + \tau_k \{\max[0, \ -x_2]\}^2 \end{aligned}$$

To explain the principle of the exterior penalty method, we use a traditional method to solve the unconstrained optimisation problem. The necessary conditions for minimising the above penalty function are given by

$$\begin{aligned} \frac{\partial P}{\partial x_1} &= (x_1 + 1)^2 - 2\tau_k \{\max[0, 1 - x_1]\} = 0 \\ \frac{\partial P}{\partial x_2} &= 1 - 2\tau_k \{\max[0, \ -x_2]\} = 0 \end{aligned}$$

Assume that the constraint $g_1(x) = 1 - x_1 \leq 0$ is satisfied, or $\max[0, 1 - x_1] = 0$, which means that $x_1 \geq 1$. From the above first order necessary condition, we must have $(x_1 + 1)^2 = 0$, or $x_1 = -1$, which is in contradiction with the assumption. We therefore have to assume that $1 - x_1 > 0$, or the first constraint is not met, leading to $\max[0, 1 - x_1] = 1 - x_1$. From the first order necessary condition, we then have

$$(x_1 + 1)^2 - 2\tau_k (1 - x_1) = 0$$

or

$$x_1 = -1 - \tau_k + \sqrt{\tau_k^2 + 4\tau_k}$$

Similarly, only when $1 + 2\tau_k x_2 = 0$, the second necessary condition can be satisfied. We then have

$$x_2 = -\frac{1}{2\tau_k}$$

For $\tau_k \to \infty$, we get the following optimal solution

$$x_1 = 1, \quad x_2 = 0$$

Using the iterative method, we could set the initial value of the penalty coefficient to be 0.001, for example, and $\tau_{k+1} = 10\tau_k$. The sequence of solutions generated is as shown in Table 3.1.

Table 3.1: Iterative optimisation process of the exterior method

τ_k	x_1^k	x_2^k	$P^k(\tau_k)$	$f^k(\tau_k)$
0.001	-0.93775	-500.00000	-249.9962	-500.0000
0.010	-0.80975	-50.00000	-24.965	-49.9977
0.100	-0.45969	-5.00000	-2.2344	-4.9474
1.000	0.23607	-0.50000	0.9631	0.1295
10.000	0.83216	-0.05000	2.3068	2.0001
100.000	0.98039	-0.00500	2.6249	2.5840
1000.000	0.99800	-0.00050	2.6624	2.6582
10000.000	0.99963	-0.00005	2.6655	2.6652
∞	1	0	8/3	8/3

3.4.3 Method of Interior Penalties

The interior penalty method consists of constructing a penalty function and searching for a sequence of solutions that approach an optimum inside the feasible space. The penalty function of the interior method is therefore different from that of the exterior method.

For the optimisation problem defined by (3.2.17) to (3.2.19), construct the following penalty function

$$P(x, \tau_k) = f(x) + \tau_k \sum_{j=1}^{l} -\frac{1}{g_j(x)} \tag{3.4.18}$$

where τ_k is the penalty coefficient at the k-th iteration with $\tau_k > 0$ and $\tau_{k+1} < \tau_k$.

For a given τ_k, we use the unconstrained optimisation method to solve problem (3.4.18), resulting in a solution $\bar{x}(\tau_k)$. When $\tau_k \to 0$, $\bar{x}(\tau_k)$ approaches the optimal solution $\bar{x}$ of the original problem, or $\bar{x}(\tau_k) \to \bar{x}$.

If x is within the feasible space and far away from the boundary, then $g_j(x) < 0$, the penalty term is small and $P(x, \tau_k) \approx f(x)$. When x moves towards the boundary of the feasible space and some constraint tends to be binding, or $g_j(x) \to 0$, then the penalty function will become very large, preventing x from moving out of the feasible space.

The computational steps of the interior penalty method can be summarised as follows:

S1 Select a feasible solution $\bar{x}^0$ that satisfies all constraints, or $g_j(\bar{x}^0) \leq 0$ for all $j = 1, 2, ..., l$. Choose an initial value τ_1 of the penalty coefficient and let $k = 1$.

S2 Taking $\bar{x}^{k-1}$ as a starting point, use an unconstrained optimisation method to find the optimum of the penalty function, denoted by $\bar{x}^k$.

S3 Check if the solution $\bar{x}^k$ is the optimum solution of the original problem. If yes, stop the process. Otherwise, go to the next step. The termination criterion can be one of the following forms:
a) The relative difference between the objective values of two successive iterations is sufficiently small, or
$\left|\frac{f(x^{k+1})-f(x^k)}{f(x^{k+1})}\right| \leq \varepsilon_1$ with ε_1 being a very small real number,
b) The variables do not have significant changes in two successive iterations
$\left|x^{k+1} - x^k\right| \leq \varepsilon_2$ with ε_2 being a very small real number.

S4 Set $\tau_{k+1} = c\tau_k$ with $c < 1$, for example $c = 0.1$. Let $k = k + 1$ and go to S2.

Example 3.6 Use the interior penalty method to solve the optimisation problem as given in Example 3.5.

Solution: The interior penalty function is given by

$$P(x, \tau_k) = \frac{1}{3}(x_1 + 1)^3 + x_2 - \frac{\tau_k}{1 - x_1} - \frac{\tau_k}{- x_2}$$

Applying the necessary condition to the penalty function results in

$$\frac{\partial P}{\partial x_1} = (x_1 + 1)^2 - \frac{\tau_k}{(1 - x_1)^2} = 0$$
$$\frac{\partial P}{\partial x_2} = 1 - \frac{\tau_k}{x_2^2} = 0$$

Solving the above equations, we have

$$\bar{x}_1^k(\tau_k) = \sqrt{1 + \sqrt{\tau_k}}$$
$$\bar{x}_2^k(\tau_k) = \sqrt{\tau_k}$$

The optimum solution is obtained for $\tau_k \to 0$, or

$$\bar{x} = \lim_{\tau_k \to 0} \begin{bmatrix} \bar{x}_1^k(\tau_k) \\ \bar{x}_2^k(\tau_k) \end{bmatrix} = \lim_{\tau_k \to 0} \begin{bmatrix} \sqrt{1+\sqrt{\tau_k}} \\ \sqrt{\tau_k} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

which is the same as we obtained using the exterior penalty method.

Using the iterative method, we could set the initial value of the penalty coefficient to be 1000, for example, and $\tau_{k+1} = 0.1\tau_k$. The sequence of solutions generated is as shown in Table 3.2.

Table 3.2: Iterative optimisation process of the interior method

τ_k	x_1^k	x_2^k	$P^k(\tau_k)$	$f^k(\tau_k)$
1000.000000	5.71164	31.62278	376.2636	132.4003
100.000000	3.31662	10.00000	89.9772	36.8109
10.000000	2.04017	3.16228	25.3048	12.5286
1.000000	1.41421	1.00000	9.1046	5.6904
0.100000	1.14727	0.31623	4.6117	3.6164
0.010000	1.04881	0.10000	3.2716	2.9667
0.001000	1.01569	0.03162	2.8569	2.7615
0.000100	1.00499	0.01000	2.7267	2.6967
0.000010	1.00158	0.00316	2.6856	2.6762
0.000001	1.00050	0.00100	2.6727	2.6697
0	1	0	8/3	8/3

In the interior penalty method, the penalty term can also be taken as the logarithm of the constraint functions. Such a method is called the logarithmic interior penalty method. If the original optimisation problem is defined by (3.2.17) to (3.2.19), then the logarithmic interior penalty function is given by

$$P(x, \tau_k) = f(x) - \tau_k \sum_{j=1}^{l} \ln(-g_j(x)) \tag{3.4.19}$$

Example 3.7 Use the logarithmic interior penalty method to solve the following problem (Cai, 1982).

$$\begin{aligned} \text{Min} \quad & f(x) = x_1 + 2x_2 \\ s.t. \quad & g_1(x) = x_1^2 - x_2 \le 0 \\ & g_2(x) = -x_1 \le 0 \end{aligned}$$

Solution: The penalty function of the problem is given by

$$P(x, \tau_k) = x_1 + 2x_2 - \tau_k \ln(-x_1^2 + x_2) - \tau_k \ln x_1$$

The necessary optimisation conditions read

$$\frac{\partial P}{\partial x_1} = 1 + \frac{2\tau_k x_1}{-x_1^2 + x_2} - \frac{\tau_k}{x_1} = 0$$
$$\frac{\partial P}{\partial x_2} = 2 - \frac{\tau_k}{-x_1^2 + x_2} = 0$$

Solving the above equations, we have

$$x_1^k(\tau_k) = \frac{1}{8}\left(-1 + \sqrt{1 + 16\tau_k}\right)$$
$$x_2^k(\tau_k) = \frac{1}{64}\left(-1 + \sqrt{1 + 16\tau_k}\right)^2 + \frac{1}{2}\tau_k$$

For $\tau \to 0$, we get the following optimal solution

$$x_1^k(\tau_k = 0) = 0, \quad x_2^k(\tau_k = 0) = 0$$

The above solution can also be obtained using the iterative method. Set the initial value of the penalty coefficient $\tau_k = 4$, for example, and let $\tau_{k+1}/\tau_k = c = 0.5$. Then, we can generate the results as shown in Table 3.3.

Table 3.3: Iterative optimisation process of the interior method

k	τ_k	$x_1^k(\tau_k)$	$x_2^k(\tau_k)$	$f^k[x^k(\tau_k)]$
1	4	0.885	2.781	6.447
2	2	0.593	1.350	3.293
3	1	0.391	0.653	1.697
4	0.5	0.375	0.390	1.155
5	0.25	0.155	0.149	0.453
6	0.125	0.090	0.071	0.232
7	0.630	0.057	0.035	0.127
8	0.032	0.029	0.017	0.063
9	0.016	0.016	0.008	0.032
10	0.008	0.008	0.004	0.016
11	0.004	0.003	0.002	0.007
∞	0	0	0	0

3.5 Summary

In this chapter, we discussed basic concepts and typical methods in constrained optimisation. Both the Kuhn-Tucker necessary conditions and the sufficient conditions for optimisation were introduced. It should be noted that the Kuhn-Tucker conditions are both necessary and sufficient for convex optimisation problems. Two types of sufficient conditions are discussed,

one based on second order derivatives (Hessian matrices) and the other on saddle points. Note that the saddle point based sufficient condition is very general and does not require assumptions about convexity, differentiability or continuity of mathematical problems in question.

Two types of constrained optimisation methods, namely primal and dual methods, were introduced. Sequential linear programming and sequential quadratic programming are two primal methods that have been implemented in some optimisation software packages. Lagrangean methods, exterior penalty method and interior penalty method are widely used dual methods. They have been implemented in many standard optimisation tools. They have also been used in genetic algorithm based optimisation procedures to solve complex optimisation problems that have many local optima and are non-differentiable at points or strongly nonlinear.

Chapter 4

Multiple Objective Optimisation

4.1 Introduction

In the previous chapters, we discussed typical methods for optimisation with one objective function. In many real-life optimisation problems, multiple objectives have to be taken into account, which may be related to the economical, technical, social and environmental aspects of optimisation problems. In this chapter, we first discuss the basic concepts and principles of multiple objective optimisation and then introduce several typical multiple objective methods.

In general, a multiple objective optimisation problem can be represented as the following vector mathematical programme

$$\text{optimise} \quad F(x) = \{f_1(x) \cdots f_l(x) \cdots f_k(x)\} \tag{4.1.1}$$

$$s.t. \quad x \in \Omega \tag{4.1.2}$$

$$\Omega = \left\{ x \;\middle|\; \begin{array}{l} g_i(x) \leq 0 \quad i = 1, ..., m_1 \\ h_j(x) = 0 \quad j = 1, ..., m_2 \\ x = [x_1 \cdots x_n]^T \end{array} \right\} \tag{4.1.3}$$

where x_i is a decision variable, x denotes a solution, $f_l(x)$ is a nonlinear objective function, and $g_i(x)$ and $h_j(x)$ are nonlinear inequality and equality constraint functions respectively. These multiple objectives are usually incommensurate and in conflict with one another. This means that, in general, a multiple objective optimisation problem does not have a single solution that could optimise all objectives simultaneously. Otherwise, there is no need to consider multiple objectives. Because of this, multiple objective optimisation is not to search for optimal solutions but for efficient (non-inferior, non-dominated or Pareto-optimal) solutions that can best attain the prioritised multiple objectives as greatly as possible. Such solutions are referred to

as the best compromise solutions.

4.2 Basic Concepts and Methods

In this section, we first introduce the concepts of non-dominance (efficiency), non-dominated (efficient) solutions and trade-offs; then, the Kuhn-Tucker conditions for Pareto optimality will be stated; finally, one of the simplest multiple objective optimisation method, the simple additive weighting method, will be discussed in some detail.

4.2.1 Concepts and Definitions

In a single objective optimisation problem defined in (3.1.1) to (3.1.3), any two solutions x^1 and x^2 can be compared completely, that is:

either $x^1 \succ x^2$ if and only if $f(x^1) < f(x^2)$

or $x^1 \prec x^2$ if and only if $f(x^1) > f(x^2)$

or $x^1 \text{ - } x^2$ if and only if $f(x^1) = f(x^2)$

where $\succ$ reads 'is preferred to' and - 'is indifferent to'. For instance, $x^1 \succ x^2$ means that the solution x^1 is preferred to x^2 and x^1-x^2 reads x^1 is indifferent to x^2. Therefore, for a single objective problem we can find an optimal solution x^*, at which the objective function is minimised.

In a multiple objective optimisation problem as defined in (4.1.1) to (4.1.3), however, not all solutions could be compared completely, that is:

either $x^1 \succ x^2$ if and only if $F(x^1) \leq F(x^2)$ (Strict inequality for at least one objective)

or $x^1 \prec x^2$ if and only if $F(x^1) \geq F(x^2)$ (Strict inequality for at least one objective)

or $x^1 \text{ - } x^2$ if and only if $F(x^1) = F(x^2)$

or $x^1 \prec\succ x^2$ if and only if $F(x^1) <> F(x^2)$.

Here $x^1 \succ x^2$ means that x^1 dominates x^2 as the former is better than the latter on all objectives. $x^1 \prec\succ x^2$ means that the two solutions x^1 and x^2 are not dominating each other if x^1 performs better than x^2 on some objectives but worse on other objectives.

In general, we are not interested in solutions dominated by other solutions. It is also the case that no feasible solution could dominate all other feasible solutions; otherwise there would be no need to conduct multiple objective optimisation. Therefore, we need to find solutions that are not dominated by any other solutions. Such solutions are called non-dominated, non-inferior, efficient or Pareto-optimal solutions. Non-dominance results from the conflicts among objectives, which are inherent in multiple objective optimisation. Non-dominance means that the improvement of some objective could only be achieved at the expense of other objectives. We use the following example to illustrate the concept of non-dominance.

Example 4.1 In Table 4.1, the two objectives $f_1(x)$ and $f_2(x)$ are for minimisation. Identify whether the four solutions are dominated or non-dominated solutions.

Table 4.1: Non-dominance

Solution	Objective function	
	$f_1(x)$	$f_2(x)$
x^1	-10	-21
x^2	-14	-18
x^3	-12	-16
x^4	-8	-20

Solution: Since $\begin{bmatrix} f_1(x^1) \\ f_2(x^1) \end{bmatrix} = \begin{bmatrix} -10 \\ -21 \end{bmatrix} < \begin{bmatrix} -8 \\ -20 \end{bmatrix} = \begin{bmatrix} f_1(x^4) \\ f_2(x^4) \end{bmatrix}$, from the definition of dominance we can see that x^1 dominates x^4, or $x^1 \succ x^4$. Similarly, we can find that $x^2 \succ x^3$, $x^1 \prec\succ x^2$, $x^1 \prec\succ x^3$, and $x^2 \prec\succ x^4$. Therefore, among the four solutions x^1 and x^2 are non-dominated (efficient) solutions and x^3 and x^4 are dominated solutions.

Non-dominance or efficiency can be formally defined as follows:

Definition 4.1 *x^t is called an efficient (non-inferior, non-dominated or Pareto-optimal) solution of problem (4.1.1) if there does not exist any $x \in \Omega$ ($x \neq x^t$), so that $F(x) \leq F(x^t)$ and $F(x) \neq F(x^t)$, where f_i ($i = 1, 2, ..., k$) are assumed for minimisation.*

The condition of efficiency is rather strict and many multiple optimisation algorithms cannot guarantee to generate efficient solutions but only weakly efficient solutions. Weak efficiency is defined as follows:

Definition 4.2 *x^t is called a weakly efficient (non-inferior, non-dominated or Pareto-optimal) solution of problem (4.1.1) if there does not exist any $x \in \Omega$ ($x \neq x^t$), so that $F(x) < F(x^t)$, where f_i ($i = 1, 2, ..., k$) are assumed for minimisation.*

The condition of weak efficiency is easier to satisfy than dominance. In other words, an efficient solution must be a weakly efficient solution but the opposite may not be true. For a two objective problem, the geometric interpretation of efficient or weakly efficient solutions can be shown as in Figure 4.1 for a discrete decision space, where there are 9 solutions. The weakly efficient solutions include x^1, x^2, x^3, x^5, x^7, x^8 and x^9, among which x^2 and x^8 are not efficient solutions and they are weakly dominated by x^1 and x^9, respectively. Solutions x^4 and x^6 are inefficient as they are dominated by x^5 and x^7, respectively.

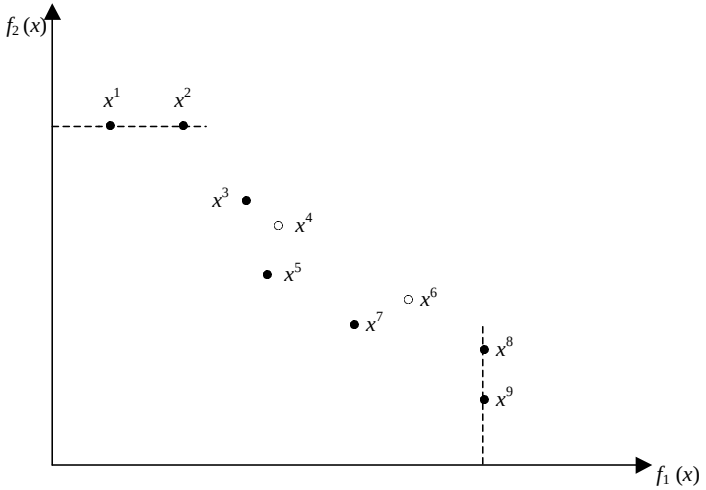

Figure 4.1: Efficient and weakly efficient solutions

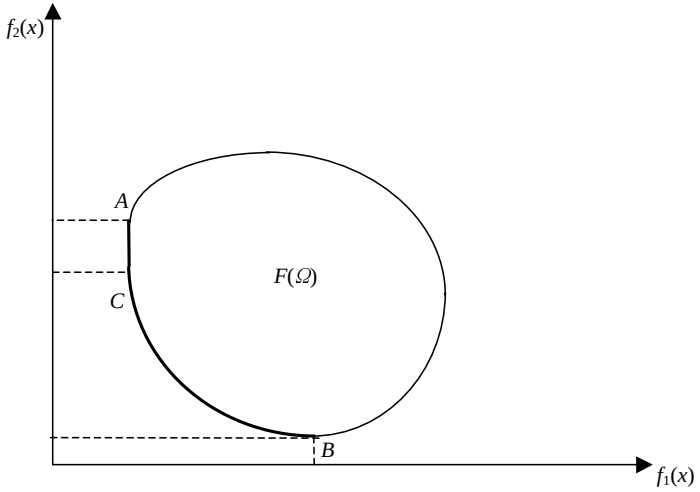

Figure 4.2: Efficient solution frontier

For a continuous decision space, Figure 4.2 illustrates efficient solutions, weakly efficient solutions and an efficient solution frontier, where $f_1(x)$ and $f_2(x)$ are assumed for minimisation. Note that all points on the line segment between points A and C are weakly efficient solutions with only point C being strictly efficient. All points on the curve between point C and point B are efficient solutions. In Figure 4.2, the line segment $\bar{A}\bar{C}$ and the curve $\tilde{C}\tilde{B}$ constitute

the efficient solution frontier of the example. In general, an efficient solution frontier can be defined as follows:

Definition 4.3 *The collection of all efficient solutions of problem (4.1.1) is called the efficient set. The image of the efficient set by F is referred to as the efficient solution frontier or trade-off surface.*

From Figure 4.2, one can find that in a multiobjective optimisation problem there is normally infinite number of efficient solutions due to the conflicts between objectives. Therefore, multiple objective decision making usually comprises two main steps: generation of efficient solutions and identification of the best compromise solution which should be an efficient solution.

4.2.2 Method Classification

The best compromise solution should be a solution that can best satisfy the DM's preferences. If the DM's preferences can be modelled by a utility function aggregating all objective functions into one criterion, denoted by

$$u(F(x)) = u(f_1(x) \cdots f_l(x) \cdots f_k(x)) \tag{4.2.1}$$

then the best compromise solution may be defined as a solution that maximises the utility function $u(F(x))$ for all $x \in \Omega$.

If an explicit utility function can be constructed, then a multiple objective optimisation problem reduces to a single objective problem and can be solved using various methods such as those discussed in the previous chapter. In many decision situations, however, it is very difficult or impossible to extract global preference information from the decision maker to construct such an explicit utility function. In such circumstances, other methods must be employed that only use local preference information.

In the past two and a half decades or so, a large number of multiple optimisation methods have been developed. Most methods are based on the principle of solving one or a sequence of single objective optimisation problems to generate efficient solutions or the best compromise solutions. Based on the ways of extracting the decision maker's preference information and using it in decision analysis processes, multiple objective optimisation methods can be divided into three main classes: 1) efficient solution generation methods with preferences provided after optimisation; 2) methods for generating the best compromise solutions based on preferences provided *a priori*; and 3) interactive methods with preferences extracted progressively in decision analysis processes (Yang, 1996).

In the first class of methods, a set of desirable efficient solutions is generated. The values of these solutions and the objective functions are presented to the decision maker, who is expected to choose the best compromise solutions from these solutions on the basis of his preferences. This type of methods

is sometimes referred to as *posterior* methods. One of the advantages of this type of methods is that there is no need to construct an explicit utility function or to involve the decision maker in the generation of efficient solutions. However, the algorithms of these methods are usually complicated and require a large number of calculations. Another drawback is that if many efficient solutions are generated, then it will be difficult for the decision maker to decide which one is the best compromise solution. In this chapter, we only discuss the simple weighting method to introduce the ideas of generating efficient solutions.

The second type of methods requires the decision maker to provide global preference information in advance. Using the preferences, a multiple objective problem is then transformed to a single objective problem, the solution of which leads to the best compromise solution of the original problem. This type of methods is sometimes referred to as *a priori* methods. One of the advantages of these methods is that optimisation only needs to be conducted once and therefore the number of calculations is relatively small. However, it is difficult for the decision maker to provide global preference information. This is particularly the case for a new multiple objective problem where little or no *a priori* knowledge about the trade-offs among objectives is available.

The third type of methods requires the decision maker to provide appropriate local preference information progressively in an interactive optimisation and decision analysis process. Using the preference information, a sequence of single objective optimisation problems are constructed, which are related to the original problem in a certain way and the solutions of which will then approach the best compromise solution of the original problem. As the above feature stands, this type of methods is referred to as *interactive* methods. Many interactive methods have been developed and their main differences result from the ways of eliciting local preferences and constructing single objective optimisation problems. Since local preference information is relatively easy to provide, interactive methods are of wide application.

Following the brief introduction of the simple weighting method, the rest of the chapter is devoted to discussing the second and the third types of methods.

4.2.3 Simple Weighting Method

The weighting method is the simplest multiple objective optimisation method and has been widely applied, though not always adequately. In this section, we use a simple example and several diagrams to discuss the process and features of using the weighting method to generate efficient solutions.

Two objection optimisation problems are widespread; for example, maximising profit with the desire of minimising energy consumption, minimising cost with the requirement of maximising safety, etc. The main features of these problems include different units used to measure two objectives and

conflicting interests between two objectives, as common to most multiple objective optimisation problems. In general, a two objective optimisation problem can be represented as follows:

$$\text{Max} \quad f_1(x) \tag{4.2.2}$$
$$\text{Mim} \quad f_2(x) \tag{4.2.3}$$
$$s.t. \quad x \in \Omega \tag{4.2.4}$$

where Ω is as defined in (4.1.3). Note that maximisation of $f_1(x)$ is equivalent to minimisation of $-f_1(x)$.

To simplify the discussion, we assume that the two objective functions of the above problem are measured using the same scale; otherwise, they need to be normalised to the same scale. Suppose the projection of the feasible decision space to the objective space is denoted by $f(\Omega)$. If the objective space $f(\Omega)$ is convex and compensation between the two objectives is allowed, then we can use the weighting method to generate efficient solutions. The weighing method operates based on the following single objective optimisation problem

$$\text{Min} \quad f(x) = \omega_1 f_1(x) + \omega_2 f_2(x) \tag{4.2.5}$$
$$s.t. \quad x \in \Omega \tag{4.2.6}$$

where $\omega_1 \geq 0$ and $\omega_2 \geq 0$ are weighting factors.

In a single objective mathematical programme, dividing the objective by a positive real number does not change its optimum. Suppose $\omega_1 > 0$. Then, dividing the objective of problem (4.2.5) by ω_1 and setting $\omega = \omega_1/\omega_2$ lead to the following equivalent problem

$$\text{Min} \quad f(x, \omega) = f_1(x) + \omega f_2(x) \tag{4.2.7}$$
$$s.t. \quad x \in \Omega \tag{4.2.8}$$

Since Ω is assumed to be convex. For a given ω, the optimal solution of problem (4.2.7) is an efficient solution of problem (4.2.2) - (4.2.4). Given different values for ω, many efficient solutions can be generated. Here, we treat ω as a weighting parameter or a variable but not as the representation of the decision maker's preferences. We use a diagram (Figure 4.3) to explain how the weighting method works to generate efficient solutions.

The contour of the objective function is a line in the objective space, defined as follows

$$f_1 + \omega f_2 = a \tag{4.2.9}$$

where a is a constant. Equation (4.2.9) can be written as the following equivalent form

$$f_2 = -\frac{1}{\omega} f_1 + \frac{a}{\omega} \tag{4.2.10}$$

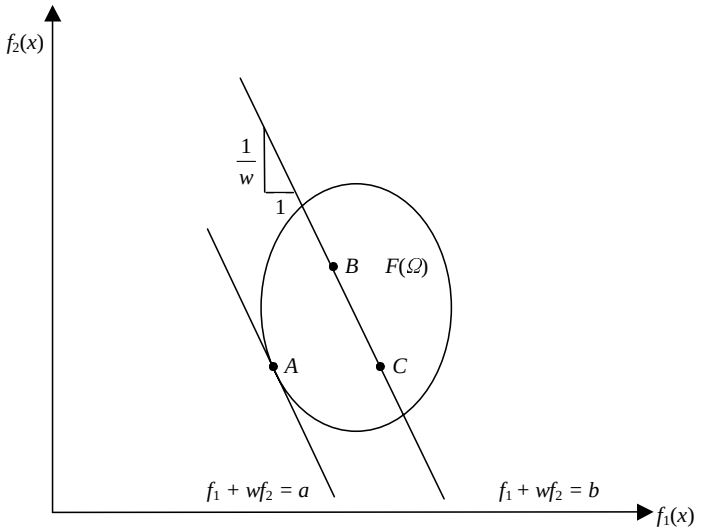

Figure 4.3: Simple weighting method

So the slope of the line is $-1/\omega$ and its y-intercept is a/ω.

On the line represented by (4.2.9) or (4.2.10), all points within the feasible objective space have the same weighted objective value and therefore the line is referred to as the linear indifference curve. For example, point B and point C in Figure 4.3 represent two solutions, at which the weighted objective function has the same value, or $f_1(B)+\omega f_2(B) = f_1(C)+\omega f_2(C) = b$. In other words, the two solutions represented by point B and point C are indifferent to the decision maker.

The solution of problem (4.2.7) is to move the objective line southwestwards in parallel as far as possible until it becomes tangent to the feasible objective space, as at point A in Figure 4.3. One can see that point A is an efficient solution. If ω represents the decision maker's preferences and the assumption of a linear utility function is acceptable, then point A would be his best compromise solution.

Changing ω is equivalent to rotating the objective line or changing the decision maker's preferences. In summary, there are the following cases.

a) Suppose the new weight is ω^1 with $\infty > \omega^1 > \omega$. Then the best compromise solution will be changed to point D as shown in Figure 4.4. Increasing ω to ω^1 means that the weight of f_2 is increased. In other words, the decision maker prefers to further improve (reduce) f_2. However, this can only be done at the expense of f_1.

b) Suppose the new weight is ω^2 with $\omega > \omega^2 > 0$. Then the best compromise solution will be changed to point E as shown in Figure 4.4.

Decreasing ω to ω^2 means that the weight of f_2 is reduced. In other words, the decision maker prefers to further improve (reduce) f_1. However, this can only be done at the expense of f_2.

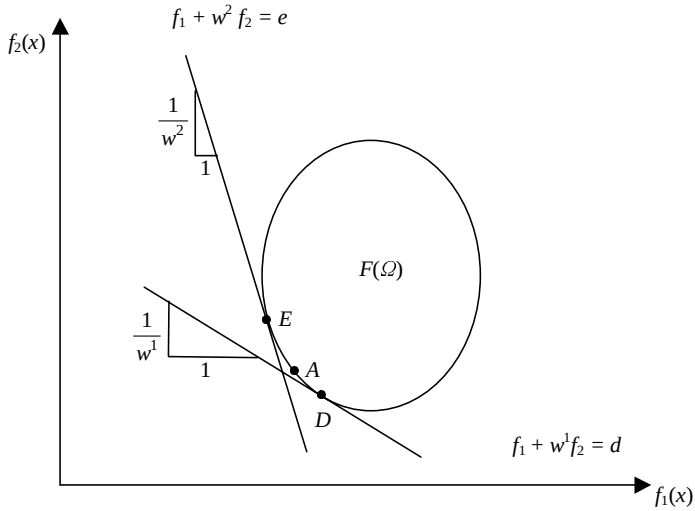

Figure 4.4: Change weight

c) If $\omega = 0$, then the best compromise solution will be changed to point G as shown in Figure 4.5. This is equivalent to say that f_2 is no longer considered and the decision maker only wishes to minimise f_1. Note that although point G in Figure 4.5 is an efficient solution, in general cases it may only be weakly efficient.

d) If $\omega = \infty$, then the best compromise solution will be changed to point H as shown in Figure 4.5. This is equivalent to say that f_1 is no longer considered and the decision maker only wishes to minimise f_2. Note that although point H in Figure 4.5 is an efficient solution, in general cases it may only be weakly efficient as well.

For three and more objective optimisation problems, the weighting method can be applied in the same way as for two objective problems. If the objective space $f(\Omega)$ of problem (4.1.1) - (4.1.3) is convex, then we could construct the following problem to generate efficient solutions.

$$\text{Min} \qquad F(x) = \sum_{l=1}^{k} \omega_l f_l(x) \tag{4.2.11}$$

$$s.t. \qquad x \in \Omega \tag{4.2.12}$$

It must be made clear, however, that if the objective space of the original problem is non-convex, then the weighting method may not be capable of

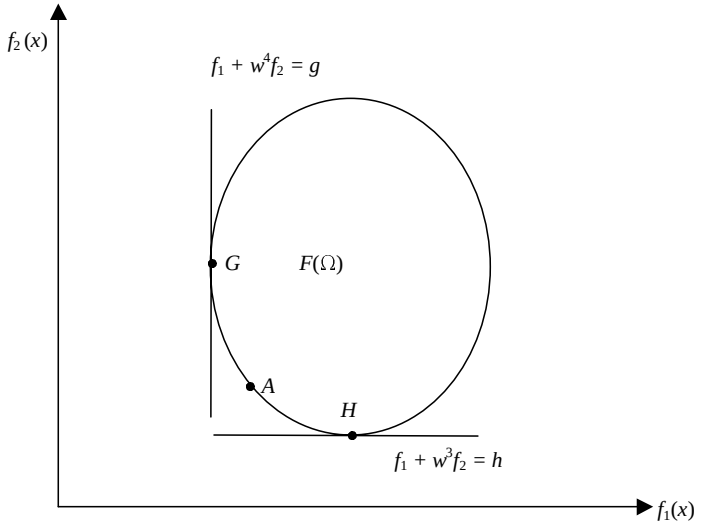

Figure 4.5: Change weight

generating the efficient solutions on the non-convex part of the efficient frontier. It must also be noted that the optimal solution of a weighting problem should not be used as the best compromise solution if the weights do not reflect the decision maker's preferences or if the decision maker does not accept the assumption of a linear utility function.

4.3 p-Norm Methods

In this section, we discuss a family of multiple objective optimisation methods, which are based on p-norm to search for the best compromise solutions. Most of these methods require global preference information. Some of them could be used in an interactive fashion. Four methods will be discussed in this section: the minimax (ideal point) method, the goal attainment method, goal programming, and the preference point method.

4.3.1 Minimax (Ideal Point) Method

In problem (4.1.1)-(4.1.3), we can optimise each of the objectives by solving the following problems

$$\text{Min} \quad f_l(x), \quad l = 1, 2, ..., k \tag{4.3.1}$$

$$s.t. \quad x \in \Omega \tag{4.3.2}$$

Suppose the optimal solution of the above problem is $\bar{x}^l$. Then, the optimal value of objective l is given by $f_l^* = f_l(\bar{x}^l)$.

Define an ideal point in the objective space as follows

$$F^* = [f_1^* \cdots f_l^* \cdots f_k^*] \tag{4.3.3}$$

In general, an ideal point is not a feasible solution. Otherwise, the objective would not be in conflict with one another. For a two objective problem, an ideal point can be illustrated as in Figure 4.6, where the two objectives are for minimisation.

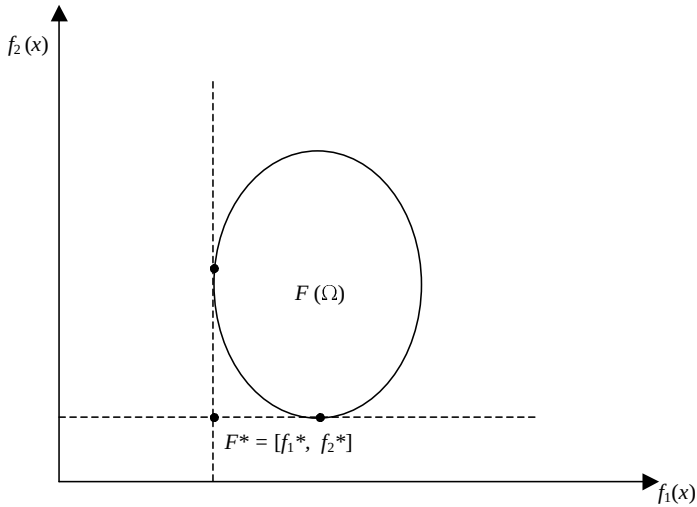

Figure 4.6: Ideal point definition

The decision rule of the ideal point method is that given a set of weights for objectives one should select a feasible solution such that the combined deviation between the feasible solution and the ideal solution is minimised. This rule means that the best compromise solution is the one that is the closest to the ideal point in the objective space.

Based on the above decision rule of the ideal point method, we can construct the following weighted p-norm problem (Yang, 1996)

$$\text{Min} \qquad d_p = \left\{ \sum_{j=1}^{k} \left[\omega_j \left| f_j^* - f_j(x) \right| \right]^p \right\}^{1/p} \tag{4.3.4}$$

$$s.t. \qquad x \in \Omega \tag{4.3.5}$$

It can be proven that for $1 \leq p \leq \infty$ if the optimal solution of the above problem is unique then it is also an efficient solution of problem (4.1.1)-(4.1.3).

Note that the weighted norm d_p is not unique. The value of d_p depends on the selection of p. It is of interest to investigate the features of d_p in relation to p. Suppose there are two points in the objective space: $F^1 = [f_1^1 \cdots f_k^1]$ and $F^2 = [f_1^2 \cdots f_k^2]$. Suppose the objectives are of equal importance. Then, the p-norm between the two points is given by

$$d_p = \left\{ \sum_{j=1}^{k} \left| f_j^1 - f_j^2 \right|^p \right\}^{1/p} \tag{4.3.6}$$

For $p = 2$, d_2 is the Euclidian (or geometric) distance between the two point F^1 and F^2. For $1 \leq p \leq \infty$, the norms d_1 and d_∞ provide the upper and lower bounds of the norm d_p, or

$$d_\infty \leq d_p \leq d_1, \quad \text{for} \quad \text{any} \quad 1 \leq p \leq \infty \tag{4.3.7}$$

Consider two cases: $p = 1$ and $p = \infty$. For $p = 1$, problem (4.3.4) becomes

$$\text{Min} \qquad d_p = \sum_{j=1}^{k} \omega_j \left| f_j^* - f_j(x) \right| \tag{4.3.8}$$

$$s.t. \qquad x \in \Omega \tag{4.3.9}$$

Problem (4.3.8) is a non-smooth optimisation problem. Since $f_j(x) \geq f_j^*$, for all $j = 1, 2, ..., k$, however, problem (4.3.8) can be re-written as follows

$$\text{Min} \qquad d_p = \sum_{j=1}^{k} \omega_j (f_j(x) - f_j^*) \tag{4.3.10}$$

$$s.t. \qquad x \in \Omega \tag{4.3.11}$$

Note that ω_j and f_j^* are both constant. Their product can be eliminated from the objective function of problem (4.3.10) without changing its optimal solution. Therefore, problem (4.3.10) is equivalent to the following problem

$$\text{Min} \qquad d_p = \sum_{j=1}^{k} \omega_j f_j(x) \tag{4.3.12}$$

$$s.t. \qquad x \in \Omega \tag{4.3.13}$$

Problem (4.3.12) is the weighting problem. This shows that the 1-norm ideal point method is equivalent to the weighting method.

For $p = \infty$, let

$$d_{\max} = \max_{1 \leq j \leq k} \left\{ \omega_j \left| f_j^* - f_j(x) \right| \right\} \tag{4.3.14}$$

We then have

$$\begin{aligned} d_\infty &= \left\{ \sum_{j=1}^{k} \left[\omega_j \left| f_j^* - f_j(x) \right| \right]^\infty \right\}^{1/\infty} \\ &= d_{\max} \left\{ \sum_{j=1}^{k} \left[\frac{\omega_j \left| f_j^* - f_j(x) \right|}{d_{\max}} \right]^\infty \right\}^{1/\infty} \end{aligned} \tag{4.3.15}$$

Since

$$0 \leq \frac{\omega_j \left| f_j^* - f_j(x) \right|}{d_{\max}} \leq 1 \tag{4.3.16}$$

we have

$$d_\infty = d_{\max} = \max_{1 \leq j \leq k} \left\{ \omega_j \left| f_j^* - f_j(x) \right| \right\} \tag{4.3.17}$$

Therefore, problem (4.3.4) becomes

$$\text{Min} \quad d_\infty = \max_{1 \leq j \leq k} \left\{ \omega_j \left| f_j^* - f_j(x) \right| \right\} \tag{4.3.18}$$

$$s.t. \quad x \in \Omega \tag{4.3.19}$$

Problem (4.3.18) is a non-smooth optimisation problem. Since $f_j(x) \geq f_j^*$ for all $j = 1, 2, ..., k$, problem (4.3.18) can be re-written as follows

$$\text{Min} \quad d_\infty = \max_{1 \leq j \leq k} \left\{ \omega_j \left(f_j(x) - f_j^* \right) \right\} \tag{4.3.20}$$

$$s.t. \quad x \in \Omega \tag{4.3.21}$$

which is still non-smooth. However, problem (4.3.20) is equivalent to the following problem

$$\text{Min} \quad d_\infty \tag{4.3.22}$$

$$s.t. \quad \omega_j \left(f_j(x) - f_j^* \right) \leq d_\infty, \quad j = 1, 2, ..., k \tag{4.3.23}$$

$$x \in \Omega \tag{4.3.24}$$

where d_∞ is treated as an auxiliary variable and (4.3.23) denotes objective constraints. It can be proven that if $\omega_j > 0$ for all $j = 1, ..., k$ or the optimal solution of problem (4.3.22)-(4.3.24) is unique, then the optimal solution is an efficient solution of the original multiple objective problem (4.1.1)-(4.1.3).

The geometrical significance of the minimax method is as illustrated by Figures 4.7. The contour of the ∞-norm in the objective space is a set of hyper-rectangles with F^* as the geometrical centre. Solving problem (4.3.22)-(4.3.24) is to find the smallest hyper-rectangle that just touches the feasible

objective space $F(\Omega)$. If $F(\Omega)$ is convex, it is always a vertex of the smallest hyper-rectangle that just touches $F(\Omega)$ as shown by point $F^1 = [f_1^1, f_2^1]$ in Figure 4.7. In this case, the following equations hold in general

$$\omega_1(f_1^1 - f_1^*) = \omega_2(f_2^1 - f_2^*) = \cdots = \omega_k(f_k^1 - f_k^*) \tag{4.3.25}$$

In Figure 4.7, equation (4.3.25) is a line denoted by $\omega_1(f_1^1 - f_1^*) = \omega_2(f_2^1 - f_2^*)$, which passes the two points F^1 and F^*. If $F(\Omega)$ is non-convex, it is possible that an edge instead of a vertex of the smallest hyper-rectangle may just touch $F(\Omega)$. In such cases, not all of equations (4.3.25) hold.

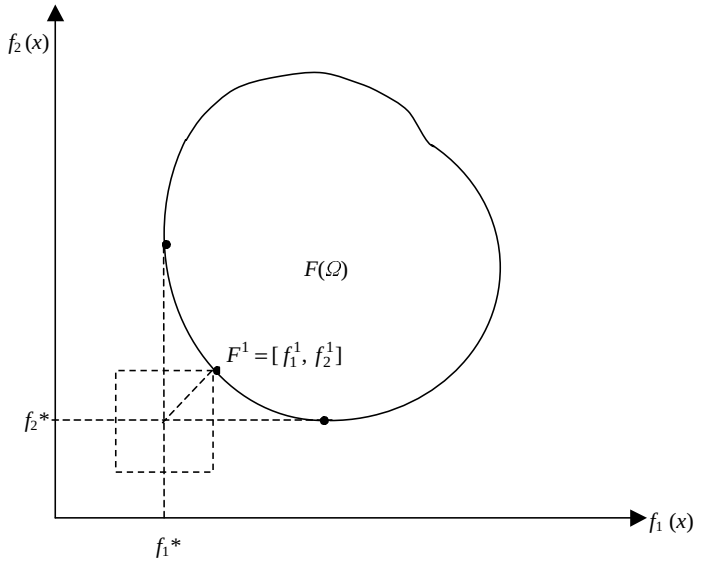

Figure 4.7: The minimax method

The computational steps of the minimax (ideal point) method are summarised as follows.

S1 Define the following multiple objective optimisation problem

$$\text{Min} \quad F(x) = \{f_1(x) \cdots f_l(x) \cdots f_k(x)\} \tag{4.3.26}$$
$$s.t. \quad x \in \Omega \tag{4.3.27}$$

Note that maximisation of $f_j(x)$ is equivalent to minimisation of $-f_j(x)$.

S2 Solve the following single objective problems

$$\text{Min} \quad f_1(x), \quad l = 1, 2, ..., k \tag{4.3.28}$$
$$s.t. \quad x \in \Omega \tag{4.3.29}$$

Suppose the optimal solution of (4.3.28) is $\bar{x}^l$ and the value of the objective $f_j(x)$ at $\bar{x}^l$ is $f_j^* = f_j(\bar{x}^l)$ $(j = 1, ..., k)$. Then construct the pay-off table as follows.

S3 Define the ideal point in the objective space and the weighting factors as follows

$$F^* = [f_1(\bar{x}^1), f_2(\bar{x}^2), \cdots, f_k(\bar{x}^k)] \tag{4.3.30}$$

$$\bar{\omega}_j = \frac{\omega_j}{f_j^- - f_j(\bar{x}^j)} \tag{4.3.31}$$

where ω is the relative weight of $f_j(x)$ and f_j^- is the worst value of $f_j(x)$ in the pay-off table, or

$$f_j^- = \max_{1 \le l \le k} \{f_j(\bar{x}^l)\} \tag{4.3.32}$$

S4 Formulate and solve the following problem to find an efficient solution, where λ is an auxiliary variable

$$\text{Min} \quad \lambda \tag{4.3.33}$$

$$s.t. \quad \bar{\omega}_j \left(f_j(x) - f_j(\bar{x}^j)\right) \le \lambda, \quad l = 1, 2, ..., k \tag{4.3.34}$$

$$x \in \Omega \tag{4.3.35}$$

S5 Suppose the optimal solution of problem (4.3.33)-(4.3.34) is x^t. If ω_j $(j = 1, 2, ..., k)$ represent the decision maker's overall preferences, x^t could be regarded as the best compromise solution. Otherwise, a new set of weights need be provided. This leads to different interactive minimax procedures.

Table 4.2: Pay-off table

	$f_1(x)$	$f_2(x)$	...	$f_k(x)$
$\bar{x}^1$	$f_1(\bar{x}^1)$	$f_2(\bar{x}^1)$	...	$f_k(\bar{x}^1)$
$\bar{x}^2$	$f_1(\bar{x}^2)$	$f_2(\bar{x}^2)$	...	$f_k(\bar{x}^2)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\bar{x}^k$	$f_1(\bar{x}^k)$	$f_2(\bar{x}^k)$	...	$f_k(\bar{x}^k)$

The minimax method is capable of discovering all efficient solutions of a multiple objective problem whether the problem is convex or non-convex (Yang, 1996; Lin and Yang, 1999). Given the relative weights of objectives, the minimax method provides a compromise solution that is nearest to the ideal solution in the sense of ∞-norm. Obviously, the quality of the compromise solution obtained in this way depends on the accuracy of the weights provided. Based on such a solution, however, interactive procedures would be designed

to conduct sensitivity analysis. This idea is adopted in the STEM method and the ISTM method as described in the next section.

Example 4.2 A two objective linear optimisation problem is given as follows

$$\begin{array}{ll} \text{Max} & F = [f_1(x), f_2(x)] = [5x_1 - 2x_2, \ -x_1 + 4x_2] \\ s.t. & x \in \Omega \end{array}$$

where

$$\Omega = \left\{ x \left| \begin{array}{l} -x_1 + x_2 \leq 3; x_1 + x_2 \leq 8 \\ x_1 \leq 6; x_2 \leq 4; x_1, x_2 \geq 0 \end{array} \right. \right\}$$

Use the weighting method and the minimax method to find the best compromise solution. Suppose the two objectives are of equal importance.

Solution: First, we optimise the two objectives individually. Maximising the first objective in Ω leads to $\bar{x}^1 = [6, 0]$ and $f_1(\bar{x}^1) = 30$; maximising the second objective in Ω leads to $\bar{x}^2 = [1, 4]$ and $f_2(\bar{x}^2) = 15$.

Using the weighting method, we can formulate the following problem

$$\begin{array}{ll} \text{Max} & d_1 = f_1(x) + f_2(x) = 4x_1 + 2x_2 \\ s.t. & x \in \Omega \end{array}$$

The optimal solution of the above problem is given by

$$\begin{array}{l} \bar{x}^3 = [\bar{x}_1^3, \bar{x}_2^3] = [6, 2] \\ f_1(\bar{x}^3) = 26 \quad \text{and} \quad f_2(\bar{x}^3) = 2 \end{array}$$

which is an efficient solution of the original problem.

Using the minimax method, we construct the following problem

$$\begin{array}{ll} \text{Min} & d_\infty \\ s.t. & 30 - 5x_1 + 2x_2 \leq d_\infty \\ & 15 + x_1 - 4x_2 \leq d_\infty \\ & x \in \Omega \end{array}$$

where d_∞, x_1 and x_2 are variables. The optimal solution of the above problem is given by

$$\begin{array}{l} \bar{x}^4 = [\bar{x}_1^4, \bar{x}_2^4] = [2.25, 2.75] \\ f_1(\bar{x}^4) = 20.75 \quad \text{and} \quad f_2(\bar{x}^4) = 5.75 \end{array}$$

which is also an efficient solution of the original problem.

Using the 1-norm (the weighting) method and the ∞-norm (the minimax) method, we generated two different solutions $\bar{x}^3$ and $\bar{x}^4$. It can be shown that

for $1 < p < \infty$ and the same set of weights any efficient solution $\bar{x}^p$ generated using a p-norm method is between the two solutions $\bar{x}^3$ and $\bar{x}^4$, or

$$f_1(\bar{x}^4) \leq f_1(x^p) \leq f_1(\bar{x}^3)$$
$$f_2(\bar{x}^3) \leq f_2(x^p) \leq f_2(\bar{x}^4)$$

For example, for $p = 2$ we can have $f_1(x^2) = 22.5$ and $f_2(x^2) = 5.5$.

4.3.2 Goal Attainment Method

In problem (4.3.22)-(4.3.23), we assume that the best compromise solution is the one that is the closest to the ideal point. In many decision situations, the decision maker may wish to specify a desired solution (goal) and find a feasible solution that is as close to the goal as possible. Such preferences could be accommodated using different ways. First of all, so-called canonical weights can be used for this purpose (Lightner and Director, 1981; Yang, 1999).

Suppose x^t is an efficient solution and $F^* = [f_1^* \cdots f_j^* \cdots f_k^*]$ is the ideal point. Define canonical weights as follows

$$\omega_j^t = \frac{1}{f_j^t - f_j^*}, \quad j = 1, 2, ..., k \tag{4.3.36}$$

where $f_j^t = f_j(x^t)$ $(j = 1, ..., k)$ and $F^t = [f_1^t \cdots f_j^t \cdots f_k^t]$. It can then be proven (Yang, 1999) that x^t is the optimal solution of the following minimax problem

$$\text{Min} \quad d_\infty \tag{4.3.37}$$
$$s.t. \quad \omega_j^t \left(f_j(x) - f_j^*\right) \leq d_\infty, \quad j = 1, 2, ..., k \tag{4.3.38}$$
$$x \in \Omega \tag{4.3.39}$$

Note that $\omega_1^t \left(f_1(x) - f_1^*\right) = \cdots = \omega_j^t \left(f_j(x) - f_j^*\right) = \cdots = \omega_k^t \left(f_k(x) - f_k^*\right)$ is a line in the objective space, which passes the two points F^* and F^t.

Following the above principle, we could ask the decision maker to provide a desired solution, say x^d. Let $F^d = [f_1^d \cdots f_j^d \cdots f_k^d]$ and $f_j^d = f_j(x^d)$. We could then define the following canonical weights

$$\omega_j^t = \frac{1}{f_j^d - f_j^*}, \quad j = 1, 2, ..., k \tag{4.3.40}$$

and solve problem (4.3.37)-(4.3.38), leading to an efficient solution x^{t+1}. If all objective constraints (4.3.38) are binding, then $F(x^{t+1})$ must be on the line passing the two points F^* and F^d.

The goal attainment method (Gembicki, 1974; Haimes *et al.*, 1975) can also be related to the minimax method. This method requires the decision

maker to provide a desired solution (goal), say x^d, and a set of weights $\omega = [\omega_1 \cdots \omega_j \cdots \omega_k]$. The goal attainment problem can be formulated as follows:

$$\begin{aligned} \text{Min} \quad & \lambda && (4.3.41)\\ s.t. \quad & f_j(x) - \omega_j \lambda \leq f_j^d, \quad j = 1, 2, ..., k && (4.3.42)\\ & x \in \Omega && (4.3.43) \end{aligned}$$

where λ is an auxiliary variable unrestricted in sign and ω is normalised so that $\sum_{j=1}^{k} \omega_j = 1$. If an objective is expected to be under-attained against the desired value, a smaller weight is assigned to it. If it is required to be over-attained, then a larger weight should be assigned to it.

The goal attainment method for two objectives is illustrated in Figure 4.8. The direction of the preferred solution vector, $F^d + \omega\lambda$, is fixed by the goal vector F^d and the weight vector ω. The minimum value of λ occurs where $F^d + \omega\lambda$ vector intersects the lower boundary of the objective space $F(\Omega)$. The best compromise solution generated using the goal attainment method is sensitive to the goal and the weight. However, the meaning of λ in problem (4.3.37)-(4.3.38) is different from d_∞ of problem (4.3.22)-(4.3.23). λ is merely an auxiliary variable while d_∞ is ∞-norm.

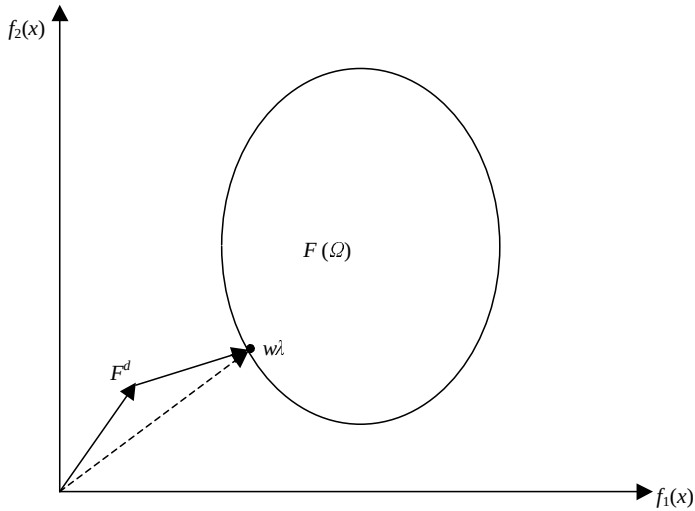

Figure 4.8: The goal attainment method

The goal attainment method may be extended. For example, a pay-off table may be constructed and the information is provided to the decision maker to help him choose the goal. The main steps of the goal attainment method can be summarised as follows:

S1 Define a multiple objective optimisation problem as in problem (4.3.26).

S2 Conduct a single objective optimisation problem and construct the pay-off table as shown in Table 4.2.

S3 Provide a goal value and a weight for each objective. Then construct either a minimax problem (problem (4.3.37)-(4.3.38)) with the canonical weights (4.3.40) or a goal attainment problem (problem (4.3.41)-(4.3.42)).

S4 Solve the constructed minimax or goal attainment problem, resulting in the best compromise solution.

Note that the goal and weights in the goal attainment formulation can be changed to develop interactive decision making processes. This will be discussed in the next section.

Example 4.3 Consider the following production scheduling problem

$$\begin{array}{ll} \text{Max} & f_1(x) = 0.4x_1 + 0.3x_2 \\ \text{Max} & f_2(x) = x_1 \\ s.t. & g_1(x) = x_1 + x_2 \leq 400 \\ & g_2(x) = 2x_1 + x_2 \leq 500 \\ & x_1, x_2 \geq 0 \end{array}$$

Solution: the goal attainment method requires that the decision maker provides a goal and weights. Suppose $F^d = [180, 200]$, and the weights of the two objective are $\omega = [\omega 1, \omega 2] = [0.67, 0.33]$. Note that maximisation of $f_j(x)$ is equivalent to minimisation of $-f_j(x)$. The goal attainment formulation of the problem is then given by

$$\begin{array}{ll} \text{Min} & \lambda \\ s.t. & g_1(x) = x_1 + x_2 \leq 400 \\ & g_2(x) = 2x_1 + x_2 \leq 500 \\ & -(0.4x_1 + 0.3x_2) - 0.67\lambda \leq -180 \\ & -x_1 - 0.33\lambda \leq -200 \\ & x_1, x_2 \geq 0 \end{array}$$

The solution of the above problem is given by

$$\begin{aligned} &\lambda = 95.4 \\ &[\bar{x}_1, \bar{x}_2] = [168.2, 163.6] \\ &[f_1(\bar{x}), f_2(\bar{x})] = [116.4, 168.2] \end{aligned}$$

4.3.3 Goal Programming

Goal programming (Charnes and Cooper, 1977) requires that preference information be provided before any efficient solutions are generated. In fact,

the method requires the decision maker to set goals for all objectives that he wishes to achieve. It adopts the decision rule that the best compromise design should be the one that minimises the deviations from the set goals. It allows the decision maker to assign pre-emptive weights to objectives and to define different achievement levels of the goals.

Goal programming is based on p-norm formulations. Suppose a goal vector is provided as $\hat{F} = [\hat{f}_1 \cdots \hat{f}_j \cdots \hat{f}_k]$. If in problem (4.3.4) the ideal point is replaced by the goal, we then have

$$\text{Min} \qquad d_p = \left\{ \sum_{j=1}^{k} \left[\omega_j \left| \hat{f}_j - f_j(x) \right| \right]^p \right\}^{1/p} \tag{4.3.44}$$

$$s.t. \qquad x \in \Omega \tag{4.3.45}$$

For linear goal programming, we set $p = 1$. Problem (4.3.44) then becomes

$$\text{Min} \qquad d_1 = \sum_{j=1}^{k} \omega_j \left| \hat{f}_j - f_j(x) \right| \tag{4.3.46}$$

$$s.t. \qquad x \in \Omega \tag{4.3.47}$$

which is a non-smooth optimisation problem.

To solve problem (4.3.46) using conventional optimisation algorithms, we introduce deviation variables d_j^+ and d_j^-

$$d_j^+ = \frac{1}{2} \left\{ \left| \hat{f}_j - f_j(x) \right| - \left[\hat{f}_j - f_j(x) \right] \right\} \tag{4.3.48}$$

$$d_j^- = \frac{1}{2} \left\{ \left| \hat{f}_j - f_j(x) \right| + \left[\hat{f}_j - f_j(x) \right] \right\} \tag{4.3.49}$$

Note that

$$d_j^+ + d_j^- = \left| \hat{f}_j - f_j(x) \right| \tag{4.3.50}$$

$$d_j^+ - d_j^- = f_j(x) - \hat{f}_j \tag{4.3.51}$$

$$d_j^+ \times d_j^- = 0, \quad d_j^+, d_j^- \geq 0 \tag{4.3.52}$$

We then have following goal programming formulation

$$\text{Min} \qquad \sum_{j=1}^{k} \omega_j (d_j^+ + d_j^-) \tag{4.3.53}$$

$$s.t. \qquad f_j(x) - d_j^+ + d_j^- = \hat{f}_j, \quad j = 1, 2, ..., k \tag{4.3.54}$$

$$d_j^+ \times d_j^- = 0, \quad d_j^+, d_j^- \geq 0, \quad j = 1, 2, ..., k \tag{4.3.55}$$

$$x \in \Omega \tag{4.3.56}$$

If the decision maker can provide goals for all objectives and accepts its decision rule, then goal programming may be one of the best methods that can be used to search for the best compromise solution. The computational steps of goal programming are listed below.

S1 Define a multiple objective optimisation problem as in (4.3.26).

S2 Set goal values $\hat{f}_j$ for all objectives $f_j(x)$, $j = 1, ..., k$.

S3 Assign a pre-emptive weight p_l to an objective where $p_l >> p_{l+1}$. This means that no number ω, however large it may be, could make $\omega p_{l+1} > p_l$. In other words, $f_j(x)$ will be regarded to be absolutely more important than $f_k(x)$ if $f_j(x)$ and $f_k(x)$ have pre-emptive weights p_l and p_{l+1}, respectively.

S4 Assign relative weights to objectives that have the same pre-emptive weight.

S5 Indicate whether each objective should be attained as closely to or as above or as below its goal value as possible.

S6 Use the above preference information to construct the goal programming problem as follows

$$\text{Min} \quad \sum_{l=1}^{L} p_l \left[\sum_{j=j_1}^{j_l} \left(\omega_j^+ d_j^+ + \omega_j^- d_j^- \right) \right] \tag{4.3.57}$$

$$s.t. \quad X_b \in \Omega_b \tag{4.3.58}$$

$$\Omega_b = \left\{ X_b \;\middle|\; \begin{array}{l} f_j(x) - d_j^+ + d_j^- = \hat{f}_j, \quad j = 1, \cdots, k \\ d_j^+ \times d_j^- = 0, d_j^+, d_j^- \geq 0 \\ X_b = [x^T d_1^+ d_1^- \cdots d_k^+ d_k^-]^T, \quad x \in \Omega \end{array} \right\} \tag{4.3.59}$$

where d_j^+ and d_j^- represent over-attainment and under-attainment of the goal $f_j(x)$. L is the total number of priority levels and j_l is the number of the objectives at a single priority level with the pre-emptive weight p_l.

S7 Problem (4.3.57) may be solved using the following sequential goal programming approach. Suppose $a_l(D^+, D^-)$ is the sum of deviations of the objectives at the lth priority level, defined by

$$a_l(D^+, D^-) = \sum_{j=j_1}^{j_l} \left(\omega_j^+ d_j^+ + \omega_j^- d_j^- \right) \tag{4.3.60}$$

where $D^+ = [d_1^+ \cdots d_k^+]^T$ and $D^- = [d_1^- \cdots d_k^-]^T$. Let a_1^* be the optimal value of $a_1(D^+, D^-)$ obtained by solving the following problem

$$\text{Min} \quad a_1(D^+, D^-) \tag{4.3.61}$$

$$s.t. \quad X_b \in \Omega_b \tag{4.3.62}$$

Then, solve the following (L-1) problems sequentially

$$\begin{aligned} \text{Min} \quad & a_l(D^+, D^-), \quad l = 2, 3, ..., L && (4.3.63)\\ s.t. \quad & X_b \in \Omega_l && (4.3.64)\\ & \Omega_l = \left\{ X_b \;\middle|\; \begin{array}{l} X_b \in \Omega_b \\ a_i(D^+, D^-) \le a_i^*, i = 1, \cdots, l-1 \end{array} \right\} && (4.3.65) \end{aligned}$$

If $\bar{X}_b = \left[\bar{x}^T \bar{d}_1^+ \bar{d}_1^- \cdots \bar{d}_k^+ \bar{d}_k^-\right]^T$ is the optimal solution of problem (4.3.63) for $l = L$, then $\bar{x}$ is the best compromise solution of the original problem.

Example 4.4 A multiple objective optimisation problem has the following four objectives: $f_1(x) = x_1$, $f_2(x) = x_2$, $f_3(x) = 8x_1 + 12x_2$, $f_4(x) = x_1 + 2x_2$. The constraints on the decision variables are $x_1 \ge 0$ and $x_2 \ge 0$. Suppose the goal values of the four objectives are $[\hat{f}_1, \hat{f}_2, \hat{f}_3, \hat{f}_4] = [30, 15, 1000, 40]$. It is hoped that all objectives are attained as closely to their goal values as possible but objectives 1, 2, and 4 are not above their goal levels and objective 3 is not below its goal level, or $f_1(x) \le \hat{f}_1$, $f_2(x) \le \hat{f}_2$, $f_3(x) \ge \hat{f}_3$ and $f_4(x) \le \hat{f}_4$. Suppose objectives 1 and 2 have the first priority, objective 3 has the second priority and objective 4 has the third priority. Suppose objective 1 and objective 2 are equally important. Use goal programming to find the best compromise solution of the problem.

Solution: The standard goal programming formulation of the problem is

$$\begin{aligned} \text{Min} \quad & f = p_1(d_1^+ + d_2^+) + p_2 d_3^- + p_3 d_4^+ \\ s.t. \quad & x_1 + d_1^- - d_1^+ = 30 \\ & x_2 + d_2^- - d_2^+ = 15 \\ & 8x_1 + 12x_2 + d_3^- - d_3^+ = 1000 \\ & x_1 + 2x_2 + d_4^- - d_4^+ = 40 \\ & x_1, x_2, d_i^-, d_i^+ \ge 0, \quad i = 1, 2, 3, 4 \end{aligned}$$

First of all, we construct a single objective optimisation model to deal with the objectives with the first priority, or $f_1(x)$ and $f_2(x)$.

$$\begin{aligned} \text{Min} \quad & f = d_1^+ + d_2^+ \\ s.t. \quad & x_1 + d_1^- - d_1^+ = 30 \\ & x_2 + d_2^- - d_2^+ = 15 \\ & x_1, x_2, d_i^-, d_i^+ \ge 0, \quad i = 1, 2 \end{aligned}$$

The optimal solution of the above problem is given by

$$\begin{aligned} & \bar{d}_1^- = 30, \quad \bar{d}_2^- = 15, \quad \bar{x}_1 = \bar{x}_2 = \bar{d}_1^+ = \bar{d}_2^+ = 0 \\ & \bar{f}_1 = \bar{f}_2 = 0 \end{aligned}$$

So, the goals of the two objectives $f_1(x)$ and $f_2(x)$ are fully attained.

We now construct a single optimisation model to deal with the objective $f_3(x)$ that has the second priority, while the full attainment of the first priority objectives must be maintained.

$$\begin{array}{ll} \text{Min} & f = d_3^- \\ s.t. & x_1 + d_1^- - d_1^+ = 30 \\ & x_2 + d_2^- - d_2^+ = 15 \\ & 8x_1 + 12x_2 + d_3^- - d_3^+ = 1000 \\ & d_1^+ + d_2^+ = 0 \\ & x_1, x_2, d_i^-, d_i^+ \geq 0, \quad i = 1, 2, 3 \end{array}$$

The optimal solution of the problem is given by

$$\begin{array}{l} \bar{x}_1 = 30, \bar{x}_2 = 15, \quad \bar{d}_1^- = \bar{d}_1^+ = \bar{d}_2^- = \bar{d}_2^+ = \bar{d}_3^+ = 0 \\ \bar{d}_3^- = 580, \quad \bar{f}_1 = 30, \quad \bar{f}_2 = 15, \quad \bar{f}_3 = 420 \end{array}$$

So the goal of the third objective $f_3(x)$ is under attained ($\bar{d}_3^- = 580$).

Finally, we consider the third priority objective $f_4(x)$. Construct the following single objective optimisation problem.

$$\begin{array}{ll} \text{Min} & f = d_4^+ \\ s.t. & x_1 + d_1^- - d_1^+ = 30 \\ & x_2 + d_2^- - d_2^+ = 15 \\ & 8x_1 + 12x_2 + d_3^- - d_3^+ = 1000 \\ & x_1 + 2x_2 + d_4^- - d_4^+ = 40 \\ & d_1^+ + d_2^+ = 0 \\ & d_3^- = 580 \\ & x_1, x_2, d_i^-, d_i^+ \geq 0, \quad i = 1, 2, 3, 4 \end{array}$$

The optimal solution of the above problem is given by

$$\begin{array}{l} \bar{x}_1 = 30, \bar{x}_2 = 15, \quad \bar{d}_1^- = \bar{d}_1^+ = \bar{d}_2^- = \bar{d}_2^+ = \bar{d}_3^+ = \bar{d}_4^- = 0 \\ \bar{d}_3^- = 580, \quad \bar{d}_4^+ = 20, \quad \bar{f}_1 = 30, \quad \bar{f}_2 = 15, \quad \bar{f}_3 = 420, \quad \bar{f}_4 = 60 \end{array}$$

So the goal of the fourth objective $f_4(x)$ is over attained.

In summary, the goals of the first two objectives $f_1(x)$ and $f_2(x)$ are fully attained whilst the goals of the third and the fourth objectives $f_3(x)$ and $f_4(x)$ are not fully attained.

4.3.4 The Minimax Reference Point Method

In the goal programming method described in the previous section, the decision maker's preferences are described using desired values (goals) as well

as relative weights and priority levels for attaining the goals. However, the goal programming method is applicable to linear and convex problems only. For non-convex problems, it will fail to identify efficient solutions on the non-convex efficient solution frontier. In this section, a minimax reference point approach (Yang, 2000) is discussed which is capable of handling the above preferences in non-convex cases as the goal programming method does in convex cases. The approach is based on ∞-norm formulation and can accommodate both pre-emptive and non-pre-emptive goal programming

Suppose a desired value (targeted value or goal) for an objective $f_i(x)$ is provided, denoted by f_i. A reference point is then represented by

$$\hat{f} = [\hat{f}_1, \hat{f}_2, ..., \hat{f}_k]^T \tag{4.3.66}$$

In this section, it is assumed that the best compromise solution is the one that is the closest to the reference point. This decision rule is similar to the rule used in goal programming. However, the distance measure used in the minimax reference point method is based on ∞-norm.

Using the decision rule and the ∞-norm to measure distance, it can be shown in the same way as in section 4.2.1 of this chapter that the best compromise solution can be generated by solving the following minimax problem

$$\begin{aligned} \text{Min} \quad & d_\infty && (4.3.67)\\ s.t. \quad & \omega_i \left| \hat{f}_i - f_i(x) \right| \leq d_\infty, \quad i = 1, ..., k && (4.3.68)\\ & x \in \Omega && (4.3.69) \end{aligned}$$

where $\omega_i (\geq 0)$ is defined as $\hat{\omega}_i / \bar{\omega}_i$ with $\bar{\omega}_i$ being a normalising factor and $\hat{\omega}_i$ a relative weight for an objective $f_i(x)$. If the reference point is given as the ideal point taking the best values of all objectives, then the absolute value sign in the objective constraints of problem (4.3.68) is not needed and the problem degenerates to a minimax problem as discussed in section 4.2.1.

Formulation (4.3.68) is a non-smooth optimisation problem. To facilitate the solution of the problem using existing mathematical programming algorithms, we transform the non-smooth problem to two new formulations, each of which has its own features and drawbacks.

In formulation (4.3.68), a non-smooth objective constraint $\omega_i \left| \hat{f}_i - f_i(x) \right| \leq d_\infty$ can be simply replaced by the following two equivalent smooth constraints

$$\omega_i (\hat{f}_i - f_i(x)) \leq d_\infty \tag{4.3.70}$$

$$-\omega_i (\hat{f}_i - f_i(x)) \leq d_\infty \tag{4.3.71}$$

A new formulation equivalent to formulation (4.3.68) can then be constructed directly using (4.3.70) and (4.3.71) as follows

$$\text{Min} \quad d_\infty \tag{4.3.72}$$

$$s.t. \quad \omega_i(\hat{f}_i - f_i(x)) \leq d_\infty \tag{4.3.73}$$
$$-\omega_i(\hat{f}_i - f_i(x)) \leq d_\infty, \quad i = 1, 2, ..., k \tag{4.3.74}$$
$$x \in \Omega \tag{4.3.75}$$

which is a smooth mathematical programming problem and can be readily solved using existing optimisation software packages.

It should be noted that the underlining preference assumption we made in formulation (4.3.72)-(4.3.74) is that the best compromise solution is the one in the feasible decision space that is the closest to the designated reference point in the sense of ∞-norm. In many decision situations, however, the decision maker may wish to express his preferences in more flexible ways as discussed in section 4.2.3 of this chapter. To accommodate wider range of preferences, we introduce deviation variables to transform formulation (4.3.68) to another formulation different from formulation (4.3.72)-(4.3.74). Similar to what we did in section 2.3, introduce the following deviation variables

$$d_i^+ = \frac{1}{2}\left\{\left|\hat{f}_i - f_i(x)\right| - \left(\hat{f}_i - f_i(x)\right)\right\} \tag{4.3.76}$$
$$d_i^- = \frac{1}{2}\left\{\left|\hat{f}_i - f_i(x)\right| + \left(\hat{f}_i - f_i(x)\right)\right\} \tag{4.3.77}$$

where d_i^+ is a deviation variable, representing the degree to which $f_i(x)$ over attains $\hat{f}_i$, and d_i^- denotes the degree that $f_i(x)$ under attains $\hat{f}_i$. Obviously the following conclusions are true

$$d_i^+ > 0 \quad \text{and} \quad d_i^- = 0 \quad \text{if} \quad f_i(x) > \hat{f}_i \tag{4.3.78}$$
$$d_i^+ = 0 \quad \text{and} \quad d_i^- > 0 \quad \text{if} \quad f_i(x) < \hat{f}_i \tag{4.3.79}$$
$$d_i^+ = 0 \quad \text{and} \quad d_i^- = 0 \quad \text{if} \quad f_i(x) = \hat{f}_i \tag{4.3.80}$$

Equations (4.3.78) to (4.3.80) can be transformed to the following equivalent

$$d_i^+ + d_i^- = \left|\hat{f}_i - f_i(x)\right| \tag{4.3.81}$$
$$d_i^+ - d_i^- = f_i(x) - \hat{f}_i \tag{4.3.82}$$
$$d_i^+ \times d_i^- = 0 \tag{4.3.83}$$
$$d_i^+, d_i^- \geq 0 \tag{4.3.84}$$

Combining problem (4.3.68) with (4.3.81) to (4.3.84), we obtain the following smooth minimax reference point formulation based on the introduction of deviation variables.

$$\text{Min} \quad d_\infty \tag{4.3.85}$$
$$s.t. \quad \omega_i(d_i^+ + d_i^-) \leq d_\infty, \quad i = 1, ..., k \tag{4.3.86}$$
$$d_i^+ - d_i^- = f_i(x) - \hat{f}_i, \quad i = 1, ..., k \tag{4.3.87}$$

$$d_i^+ \times d_i^- = 0, \quad i = 1, ..., k \tag{4.3.88}$$

$$d_i^+, d_i^- \geq 0, \quad i = 1, ..., k \tag{4.3.89}$$

$$x \in \Omega \tag{4.3.90}$$

In problem (4.3.87)-(4.3.90), there are $3k$ additional constraints added to the original constraint set Ω. Formulation (4.3.87)-(4.3.90) is more flexible to represent preferences than formulation (4.3.72)-(4.3.75). However, the former requires the nonlinear complementarity condition (4.3.88). If the original problem is linear, then problem (4.3.72)-(4.3.75) is linear, and so is problem (4.3.87)-(4.3.90) except that (4.3.88) is nonlinear. Fortunately, problem (4.3.87)-(4.3.90) can be solved using a modified simplex method with d_i^+ and d_i^- not selected as basic variables simultaneously. If the original problem is nonlinear, then both formulations (4.3.71)-(4.3.75) and (4.3.87)-(4.3.90) are ordinary nonlinear programming problems and may be solved using for example sequential linear or quadratic programming techniques.

It is easy to show that the minimax (ideal point) formulation is a special case of formulation (4.3.72)-(4.3.75) or (4.3.87)-(4.3.90). In fact, if point $\hat{f} = [\hat{f}_1, \hat{f}_2, ..., \hat{f}_k]^T$ are assigned to the best values of the corresponding objectives, we always have $f_i(x) \geq \hat{f}_i$ for any $x \in \Omega$. Therefore, constraint (4.3.72) becomes redundant, $d_i^- \equiv 0$ and $d_i^+ = \hat{f}_i - f_i(x)$ in formulation (4.3.87). Formulations (4.3.72)-(4.3.75) and (4.3.87)-(4.3.90) are then equivalent to the following ideal point formulation

$$\text{Min} \quad d_\infty \tag{4.3.91}$$

$$s.t. \quad \omega i(f_i(x) - \hat{f}_i) \leq d_\infty, \quad i = 1, ..., k \tag{4.3.92}$$

$$x \in \Omega \tag{4.3.93}$$

Note that an optimal solution of problem (4.3.91)-(4.3.93) is an efficient solution of problem (4.1.1)-(4.1.3) if all ω_i are positive or if the optimal solution is unique (Steuer and Choo, 1983). However, an optimal solution x^* of problem (4.3.72)-(4.3.75) or (4.3.87)-(4.3.90) is not guaranteed to be efficient. For example, the optimal solution will be an inferior solution if the reference point $\hat{f} = [\hat{f}_1, \hat{f}_2, ..., \hat{f}_k]^T$ happens to be an interior point of the original problem. In general, x^* may be an inferior solution if all deviation variables d_i^+ and d_i^- $(i = 1, ..., k)$ are zero at x^*. If a reference point is assigned southwest of the efficient frontier of the original problem, it can be shown that an optimal solution of problem (4.3.86)-(4.3.90) is an efficient solution of the original problem.

Instead of proving the above conclusion and developing rules for guiding the assignment of reference points, an auxiliary problem as defined below is designed to check whether an optimal solution of problem (4.3.72)-(4.3.75) or (4.3.87)-(4.3.90) is efficient. If not, an efficient solution will result from

solving the following problem

$$\text{Min} \quad \sum_{i=1}^{k} \omega_i y_i \tag{4.3.94}$$

$$s.t. \quad f_i(x) - f_i(x^*) \leq y_i, \quad i = 1, ..., k \tag{4.3.95}$$

$$x \in \Omega, \quad y_i \geq 0, \quad i = 1, ..., k \tag{4.3.96}$$

where x^* is an optimal solution of problem (4.3.87)-(4.3.90) and y_i an auxiliary variable to be maximised. Let $[\bar{x}^T, \bar{y}_1, ..., \bar{y}_k]^T$ be the optimal solution of problem (4.3.94)-(4.3.96). If $\omega_i > 0$ and all $\bar{y}_i$ are zero, then x^* is efficient. Otherwise, x^* is not an efficient solution. However, the resultant optimal solution $\bar{x}$ of problem (4.3.94)-(4.3.96) is an efficient solution dominating x^* (Yang *et al.*, 1988; Yang *et al.*, 1990).

In Figure 4.9, a two-objective maximisation problem is illustrated and the shaded area is the feasible objective space as denoted by $f(\Omega)$. Suppose the two objectives are for minimisation. The curve $\tilde{C}\tilde{D}$ (the northeast boundary of $f(\Omega)$) constitutes the efficient frontier that is non-convex. The efficient solutions on the curve $\tilde{A}\tilde{B}$ form the non-convex part of the frontier. Such solutions can never be identified using the simple weighting method. A reference point is defined by $\hat{f} = [\hat{f}_1, \hat{f}_2]$.

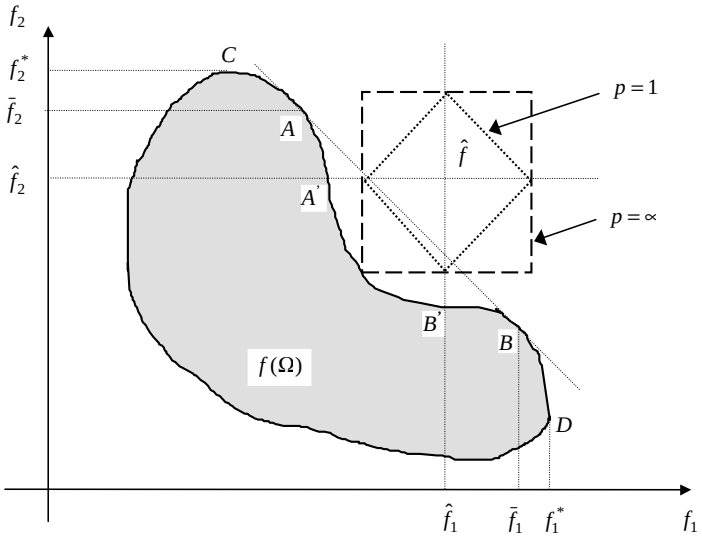

Figure 4.9: Interpretation of weighted 1-norm and ∞-norm methods

In Figure 4.9, the contour of a goal programming problem is as shown by a diamond with $\hat{f}$ as its centre. Solving the problem is equivalent to expanding the diamond around the reference point $\hat{f}$ until it just touches the feasible

objective space $f(\Omega)$. It is clear from Figure 4.9 that if the reference point is located northeast of the efficient frontier and $\hat{f}_1 \leq \bar{f}_1$, $\hat{f}_2 \leq \bar{f}_2$, then solving the traditional goal programming problem can only identify either solution A' or B' whatever weights $\hat{\omega}_i \geq 0$ $(i = 1, 2)$ may be assigned.

Similarly, the contour of ∞-norm is as shown by a rectangle with $\hat{f}$ as its centre. Solving problem (4.3.68) ((4.3.72)-(4.3.75) or (4.3.87)-(4.3.90)) is equivalent to expanding the rectangle around the reference point $\hat{f}$ until it just touches the feasible objective space. It is clear from Figure 4.9 that any efficient solution on the curve $\tilde{A}'\tilde{B}'$ can be identified by solving problem (4.3.68) with regulating ω_i $(i = 1, 2)$. From Figure 4.9, it is also clear that if the reference point is assigned to satisfy $\hat{f}_1 \leq f_1^*$,$\hat{f}_2 \leq f_2^*$, then formulation (4.3.68) will be equivalent to the ideal point formulation.

The computational steps of the reference point methods are listed below.

S1 Define a multiple objective optimisation problem as in (4.3.26).

S2 Set reference values $\hat{f}_j$ for all objectives $f_j(x)$, $j = 1, ..., k$.

S3 Assign a pre-emptive weight p_l to an objective, where $p_l >> p_{l+1}$, and relative weights to objectives that have the same pre-emptive weight.

S4 Indicate whether each objective should be attained as closely to or as above or as below its goal value as possible.

S5 Use the above preference information to construct a reference point problem as defined in formulation (4.3.72)-(4.3.75) or (4.3.87)-(4.3.90)

S6 Solve the reference point problem using a constrained optimisation method such as sequential linear programming or penalty methods.

Note that the decision maker can change the settings of reference points, pre-emptive weights and relative weights. This will lead to an interactive decision making process.

Example 4.5 A three-objective design problem (Yang, 2000) is given by

$$
\begin{aligned}
\text{Min} \quad & f_1(x) = \frac{0.2\gamma_6(x) + 40000\gamma_7^{0.3}(x) + \gamma_9(x)\gamma_{11}(x)}{\gamma_{10}(x)\gamma_{11}(x)} \\
\text{Min} \quad & f_2(x) = (\gamma_3(x) + \gamma_4(x) + \gamma_5(x))/10,000 \\
\text{Max} \quad & f_3(x) = \gamma_{10}(x)\gamma_{11}(x)/1,000,000 \\
s.t. \quad & -x_1 + 6x_5 \leq 0 \\
& x_1 - 15x_3 \leq 0 \\
& x_1 - 19x_2 \leq 0 \\
& x_2 - 0.45\gamma_7^{0.31}(x) \leq 0 \\
& x_2 - 0.7x_3 - 0.7 \leq 0 \\
& 3,000 \leq \gamma_7^{0.31}(x) \leq 500,000 \\
& 0.63 \leq x_4 \leq 0.75
\end{aligned}
$$

$$
\begin{aligned}
&14 \le x_6 \le 18\\
&x_6 - 0.32(9.8065x_1)^{0.5} \le 0\\
&-0.53x_2 + 0.52x_3 + 0.07x_5\\
&-\frac{(0.085x_4 - 0.002)x_5^2}{x_2x_4} + 1 \le 0\\
&x_1, x_2, x_3, x_4, x_5, x_6 \ge 0
\end{aligned}
$$

where x_i $(i = 1, ..., 6)$ are independent decision variables and γ_j $(j = 1, ..., 6)$ are dependent intermediate variables. Let $x = [x_1, x_2, x_3, x_4, x_5, x_6]^T$. Then γ_j $(j = 1, ..., 6)$ are functions of x defined as follows:

$$
\begin{aligned}
\gamma_1(x) &= 1.025x_1x_2x_4x_5\\
\gamma_2(x) &= \gamma_1^{2/3}(x)x_6^3\frac{(9.8065x_1)^{0.5}}{b(x_4)x_6 + a(x_4)(9.8065x_1)^{0.5}}
\end{aligned}
$$

where $b(x_4)$ and $a(x_4)$ are auxiliary functions obtained as the quadratic functions of the decision variable x_4 as follows (Yang, 2000).

$$
\begin{aligned}
b(x_4) &= -10847.2x_4^2 + 12817x_4 - 6960.32\\
a(x_4) &= 4977.06x_4^2 - 8105.61x_4 + 4456.51
\end{aligned}
$$

Other intermediate variables are given by

$$
\begin{aligned}
\gamma_3(x) &= 0.17\gamma_2^{0.9}(x)\\
\gamma_4(x) &= 1.0x_1^{0.8}x_3^{0.3}x_4^{0.1}x_5^{0.6}\\
\gamma_5(x) &= 0.034x_1^{1.7}x_3^{0.4}x_4^{0.5}x_5^{0.7}\\
\gamma_6(x) &= 1.3(2000\gamma_5^{0.85}(x) + 3500\gamma_4(x) + 2400\gamma_2^{0.8}(x)\\
\gamma_7(x) &= \gamma_1(x) - \gamma_3(x) - \gamma_4(x) - \gamma_5(x)\\
\gamma_8(x) &= 0.00456\gamma_2(x) + 0.2\\
\gamma_9(x) &= 21875\frac{\gamma_8(x)}{x_6} + 6.3\gamma_7^{0.8}(x)\\
\gamma_{10}(x) &= \gamma_7(x) - 2\gamma_7^{0.5}(x) - \gamma_8(x)\left(\frac{208.33}{x_6} + 5\right)\\
\gamma_{11}(x) &= \frac{1,400,000x_6}{833333.33 + (\gamma_{10}(x) + 4000)x_6}
\end{aligned}
$$

Suppose the three objectives are of equal importance and their goals are set to be $\hat{f}_1 = 10$, $\hat{f}_2 = 2$ and $\hat{f}_3 = 1$. Use the reference point method to find the best compromise solution.

Solution: Define the decision space Ω and a reference point $\hat{F}$ as follows.

$$
\begin{aligned}
\Omega &= \left\{ x \;\middle|\; \begin{array}{l}\text{any} \quad x = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6]^T\\ \text{that satisfies constraints of the problem}\end{array} \right\}\\
\hat{F} &= [\hat{f}_1, \hat{f}_2, \hat{f}_3] = [10,\ 2,\ 1]
\end{aligned}
$$

The above optimisation problem is strongly nonlinear. The reference point method is ideally fitted to dealing with such problems. Suppose the preferences are to find a solution that is the closest to the reference point provided. Then, a reference point model can be constructed as follows:

$$
\begin{aligned}
\text{Min} \quad & d_\infty \\
s.t. \quad & 0.1269\hat{\omega}_1(d_1^+ + d_1^-) \le d_\infty \\
& 0.1124\hat{\omega}_2(d_2^+ + d_2^-) \le d_\infty \\
& 1.1132\hat{\omega}_3(d_3^+ + d_3^-) \le d_\infty \\
& d_1^+ - d_1^- = -f_1(x) + 10 \\
& d_2^+ - d_2^- = -f_2(x) + 2 \\
& d_3^+ - d_3^- = f_3(x) - 1 \\
& d_1^+ \times d_1^- = 0, \quad d_2^+ \times d_2^- = 0, \quad d_3^+ \times d_3^- = 0 \\
& d_1^+, d_1^-, d_2^+, d_2^-, d_3^+, d_3^- \ge 0 \\
& x \in \Omega
\end{aligned}
$$

Solving the above problem leads to the following best compromise solution

$$
\begin{aligned}
& \hat{x}^1 = [275.43, 17.63, 24.19, 0.71, 45.91, 14]^T \\
& f_1(\hat{x}^1) = 10.43, \quad f_2(\hat{x}^1) = 2.485, \quad f_3(\hat{x}^1) = 0.951
\end{aligned}
$$

At $\hat{x}^1$ all the three objectives have been catered for to some extent but none of them fully achieves its targeted value. It is therefore of interest to investigate how the targeted objective values could be attained if the objectives are given different relative weights or priorities.

Suppose the targeted value of the first objective receives first-priority attention and the other two objectives are of equal priority and importance. Then the following two minimax problems can be constructed sequentially:

$$
\begin{aligned}
\text{Min} \quad & d_\infty && (4.3.97)\\
s.t. \quad & 0.1269(d_1^+ + d_1^-) \le d_\infty && (4.3.98)\\
& d_1^+ - d_1^- = -f_1(x) + 10 && (4.3.99)\\
& d_1^+ \times d_1^- = 0 && (4.3.100)\\
& d_1^+, d_1^- \ge 0 && (4.3.101)\\
& x \in \Omega && (4.3.102)
\end{aligned}
$$

$$
\begin{aligned}
\text{Min} \quad & d_\infty && (4.3.103)\\
s.t. \quad & 0.1124(d_2^+ + d_2^-) \le d_\infty && (4.3.104)\\
& 1.1132(d_3^+ + d_3^-) \le d_\infty && (4.3.105)\\
& d_2^+ - d_2^- = -f_2(x) + 2 && (4.3.106)
\end{aligned}
$$

$$d_3^+ - d_3^- = f_3(x) - 1 \tag{4.3.107}$$
$$d_2^+ \times d_2^- = 0, \quad d_3^+ \times d_3^- = 0 \tag{4.3.108}$$
$$d_2^+, d_2^-, d_3^+, d_3^- \geq 0 \tag{4.3.109}$$
$$x \in \Omega \tag{4.3.110}$$
$$f_1(x) = 10 - \tilde{d}_1^+ + \tilde{d}_1^- \tag{4.3.111}$$

In (4.3.111) $\tilde{d}_1^+$ and $\tilde{d}_1^-$ are obtained by solving problem (4.3.97)-(4.3.102).

Solving problem (4.3.97)-(4.3.102) first and then problem (4.3.103)-(4.3.111) results in the following efficient design $\hat{x}^2$

$$\hat{x}^2 = [280.85 \quad 17.56 \quad 24.09 \quad 0.68 \quad 46.81 \quad 14]^T$$
$$f_1(\hat{x}^2) = 10.0, \quad f_2(\hat{x}^2) = 2.514, \quad f_3(\hat{x}^2) = 0.9481$$

At $\hat{x}^2$, the first objective is fully achieved but the other two objectives are both under-achieved. We could change the priority of the objectives or the reference values to generate other efficient solutions. This leads to an interactive decision making process as discussed in the next section.

4.4 Interactive Methods

In the previous section, we mentioned that several methods could be implemented in an interactive fashion. In general, interactive methods are desirable in decision situations where *a priori* knowledge about decision problems in hand is not available. In this section, we introduce four interactive methods: Geoffrion's method, the STEM method, the ISTM method and the interactive gradient projection method.

4.4.1 Geoffrion's Method

Geoffrion's method (Geoffrion *et al.*, 1972) is among the earliest interactive methods. It is applicable to problems with convex decision spaces. It also assumes that a differentiable and concave utility function $u(f_1(x) \cdots f_k(x))$ exists. This method is an adoption of the Frank-Wolfe algorithm to multiobjective cases. It only requires local information in search for the best compromise solutions. The computational steps of the method may be summarised as follows:

S1 Define a multiple objective optimisation problem as in (4.3.26).

S2 Choose an initial solution x^0 arbitrarily. Let $F^0 = [f_1^0 \cdots f_j^0 \cdots f_k^0]$ where $f_j^0 = f_j(x^0)$ $(i = 1, ..., k)$. Select an objective as the reference function denoted by $f_r(x)$. Let $t = 0$.

S3 Estimate the marginal rates of substitution (or indifferent trade-offs) ω_{lr}^t between an objective $f_l(x)$ and the reference objective $f_r(x)$ at the

solution x^t. ω_{lr}^t is defined by

$$\omega_{lr}^t = \frac{\partial u(f(x^t))}{\partial f_l(x)} \Big/ \frac{\partial u(f(x^t))}{\partial f_r(x)} \tag{4.4.1}$$

The following procedure may be used to approximate ω_{lr}^t through indifference trade-off analysis. The decision maker is asked to compare the following two solutions

$$F^t = [f_1^t \cdots f_r^t \cdots f_l^t \cdots f_k^t]^T \tag{4.4.2}$$
$$\hat{F}^t = [f_1^t \cdots f_r^t + \Delta_r^t \cdots f_l^t - \Delta_l^t \cdots f_k^t]^T \tag{4.4.3}$$

where Δ_r^t and Δ_l^t are small perturbations for $f_r(x)$ and $f_l(x)$. If the decision maker prefers F^t to $\hat{F}^t$ (or $\hat{F}^t$ to F^t), then Δ_r^t or Δ_l^t is regulated until indifference between the two solutions is reached. When this happens, ω_{lr}^t is given by Δ_r^t/Δ_l^t.

S4 Search for a direction along which the utility function u may be improved from x^t. First construct the following direction problem:

$$\text{Max} \quad \sum_{j=1}^{k} \omega_{jr}^t \nabla f_j(x^t) x \tag{4.4.4}$$
$$s.t. \quad x \in \Omega \tag{4.4.5}$$

Suppose $\bar{x}^t$ is the optimal solution of (4.4.4)-(4.4.5). Then $d^t = \bar{x}^t - x^t$ is a direction along which the utility function can be improved.

S5 Formulate the following problem to find the optimal step-size

$$\text{Max} \quad u[f_1(x^t + td^t), ..., f_k(x^t + td^t)] \tag{4.4.6}$$
$$s.t. \quad 0 \le t \le 1 \tag{4.4.7}$$

Since u is not known explicitly, however, the solution of (4.4.6)-(4.4.7) can only be solved using the decision maker's judgements. One way to do this is to construct a table such as Table 4.3. Then the decision maker is required to select a t^q from the table at which the values of all the objectives are most preferred.
Suppose t^* is the chosen step size. Then $[f_1(x^t + t^* d^t) \cdots f_k(x^t + t^* d^t)]$ is the vector of the objective values at the best solution as listed in Table 4.3.

S6 If $x^{t+1} = x^t$ or $F(x^{t+1}) = F(x^t)$, the iteration is stopped. However, such a theoretical convergence criterion is not easy to satisfy. So the following approximate criterion may be used instead

$$\frac{\Delta^t}{\Delta^0} = \frac{\left|(\omega^t)^T[F(x^{t+1}) - F(x^t)]\right|}{|(\omega^0)^T[F(x^1) - F(x^0)]|} \le \alpha \tag{4.4.8}$$

where α is a small real number. In (4.4.8), Δ^t/Δ^0 represents the ratio of the improvement of the utility function obtained at interaction $t+1$ to that at interaction 1. If (4.4.8) is satisfied, the interaction is stopped. Otherwise, let $t = t+1$ and go to Step 3.

As a searching-oriented method, Geoffrion's method is based on the strict assumption that the preferences provided by the decision maker must be not only consistent but also monotonic with respect to an implicit utility function u. This assumption is difficult to satisfy. For instance, the determination of t^* becomes difficult when $k > 3$. In addition, the solution selected from Table 4.3, or $x^{t+1} = x^t + t^* d^t$, may not be an efficient solution. Finally, the termination condition (4.4.8) is not a theoretical optimality condition. It may cause premature termination of an interactive decision making process.

Table 4.3: Sampling of objectives

t^q	0	0.1	0.2	...	0.9	1
$f_1(x^t + t^q d^t)$	f_1^0	$f_1^{0.1}$	$f_1^{0.2}$	...	$f_1^{0.9}$	f_1^1
$f_2(x^t + t^q d^t)$	f_2^0	$f_2^{0.1}$	$f_2^{0.2}$	...	$f_2^{0.9}$	f_2^1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$f_k(x^t + t^q d^t)$	f_k^0	$f_k^{0.1}$	$f_k^{0.2}$	...	$f_k^{0.9}$	f_k^1

Example 4.6 Use Geoffrion's method to identify a best compromise solution of the problem given in Example 4.2. For the purpose of illustrating the method, suppose the underlying utility function is given by $u(f_1, f_2) = cf_1(x)f_2(x)$.

Solution: The original problem is given by

$$\begin{array}{ll} \text{Max} & F = [f_1(x), f_2(x)] = [5x_1 - 2x_2, \ -x_1 + 4x_2] \\ s.t. & x \in \Omega \end{array}$$

where

$$\Omega = \left\{ x \left| \begin{array}{l} -x_1 + x_2 \le 3; x_1 + x_2 \le 8 \\ x_1 \le 6; x_2 \le 4; x_1, x_2 \ge 0 \end{array} \right. \right\}$$

First of all, we generate an efficient solution using the weighting method, assuming that the two objectives are of equal importance. The weighting problem is

$$\begin{array}{ll} \text{Max} & f_1(x) + f_2(x) = 4x_1 + 2x_2 \\ s.t. & x \in \Omega \end{array}$$

The optimal solution of the weighting problem is

$$\begin{aligned} x^0 &= (x_1^0, x_2^0) = (6, 2) \\ F(x^0) &= (f_1(x^0), f_2(x^0)) = (26, 2) \end{aligned}$$

In the first interaction, construct an optimisation problem to find the search direction using the decision maker's marginal rates of substitution. Using the underlying utility function, we get

$$m_{21}^0 = \frac{\partial u/\partial f_2}{\partial u/\partial f_1} = \frac{f_1(x^0)}{f_2(x^0)} = \frac{26}{2} = 13$$

Note that

$$m_{11} = m_{22} = 1$$

Since the objectives are linear functions of the decision variables, the objective gradients are given by

$$\nabla f_1(x) = [5, -2] \quad \text{and} \quad \nabla f_2(x) = [-1, 4]$$

The direction problem is then given by

$$\begin{array}{ll} \text{Max} & \left\{ 1 \times [5, -2] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 13 \times [-1, 4] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right\} \\ s.t. & x \in \Omega \end{array}$$

or

$$\begin{array}{ll} \text{Max} & -8x_1 + 50x_2 \\ s.t. & x \in \Omega \end{array}$$

The optimal solution of the above problem is $\bar{x}^0 = [1, 4]$. Therefore, the ascent direction of the utility function at x^0 is given by

$$d^0 = \bar{x}^0 - x^0 = [-5, 2]$$

To complete the first interaction, we now construct an optimisation problem for identifying the step size. If the utility function is not known, then one needs to construct Table 4.3 to identify the step size using the decision maker's judgements. Since the underlying utility function is provided in this example, we can find the best step size using the following one-dimensional search

$$\begin{array}{ll} \text{Max} & u(f_1(x^0 + \alpha d^0), f_1(x^0 + \alpha d^0)) \\ s.t. & 0 \leq \alpha \leq 1 \end{array} \tag{4.4.9}$$

In u, let $c = 1$. Then, the problem is equivalent to the following

$$\begin{array}{ll} \text{Max} & -0.277\alpha^2 + 280\alpha + 52 \\ s.t. & 0 \leq \alpha \leq 1 \end{array}$$

The optimal solution of the above problem is $\alpha^* = 0.37$. Therefore, the new solution is given by

$$x^1 = x^0 + \alpha^* d^0 = (4.2, 2.7)$$
$$F(x^1) = (f_1(x^1), f_2(x^1)) = (15.6, 6.6)$$

In the second interaction, we use x^1 as a new starting point to construct a direction problem. Using the defined utility function, the marginal rate of substitution at x^1 is given by

$$m_{21}^1 = \frac{f_1(x^1)}{f_2(x^1)} = \frac{15.6}{6.6} = 2.4$$

The new direction problem is then given by

$$\begin{array}{ll} \text{Max} & 2.6x_1 + 7.6x_2 \\ s.t. & x \in \Omega \end{array}$$

The optimal solution of the problem and the new search direction are

$$\bar{x}^1 = [4, 4]$$
$$d^1 = \bar{x}^1 - x^1 = [-0.2, 1.3]$$

The new step size problem is given by

$$\begin{array}{ll} \text{Max} & -19.4\alpha^2 + 60.5\alpha + 103 \\ s.t. & 0 \leq \alpha \leq 1 \end{array}$$

Its optimal solution is $\alpha^* = 1$. Therefore, the new solution is given by

$$x^2 = x^1 + \alpha^* d^1 = (4, 4)$$
$$F(x^2) = (12, 12)$$

Since $x^2 \neq x^1$, the interactive process should continue. In the third interaction, the marginal rate of substitution at x^2 is given by

$$m_{21}^2 = \frac{f_1(x^2)}{f_2(x^2)} = \frac{12}{12} = 1$$

The new direction problem is then given by

$$\begin{array}{ll} \text{Max} & 4x_1 + 2x_2 \\ s.t. & x \in \Omega \end{array}$$

The optimal solution of the problem and the new search direction are

$$\bar{x}^2 = [6, 2]$$
$$d^2 = \bar{x}^2 - x^2 = [2, -2]$$

The new step size problem is given by

$$\begin{aligned} \text{Max} \quad & -14\alpha^2 + 48\alpha + 144 \\ s.t. \quad & 0 \leq \alpha \leq 1 \end{aligned}$$

Its optimal solution is $\alpha^* = 0.17$. Therefore, the new solution is given by

$$\begin{aligned} x^3 &= x^2 + \alpha^* d^2 = (4.3, 3.7) \\ F(x^3) &= (14.4, 10.3) \end{aligned}$$

Since $x^3 \neq x^2$, the interactive process should not be terminated. In the fourth interaction, the marginal rate of substitution at x^3 is given by

$$m_{21}^3 = \frac{f_1(x^3)}{f_2(x^3)} = \frac{14.4}{10.3} = 1.4$$

The new direction problem is then given by

$$\begin{aligned} \text{Max} \quad & 3.6x_1 + 3.6x_2 \\ s.t. \quad & x \in \Omega \end{aligned}$$

The optimal solution of the problem is $\bar{x}^3 = [4.3, 3.7] = x^3$. So $d^3 = \bar{x}^3 - x^3 = [0, 0]$. Therefore, $x^4 = x^3$ and the interactive process is terminated. The best compromise solution generated and the corresponding objective values are given by

$$\begin{aligned} x^4 &= x^3 = (4.3, 3.7) \\ F(x^4) &= (14.4, 10.3) \end{aligned}$$

4.4.2 The STEM Method

The STEM method is a search-oriented interactive method. It allows the decision maker to search for desirable solutions. The method is based on ∞-norm and provides a procedure to reduce the search space progressively.

The STEM method uses the following basic minimax model to search for efficient solutions

$$\begin{aligned} \text{Max} \quad & d_\infty & (4.4.10) \\ s.t. \quad & \omega_j \left(f_j(x) - f_j^* \right) \leq d_\infty, \quad j = 1, ..., k & (4.4.11) \\ & x \in \Omega^0 = \Omega & (4.4.12) \end{aligned}$$

When the STEM method is used to deal with linear problems, the weighting factors in (4.4.11) can be calculated as follows. For a linear problem, its objectives and decision space can be represented as follows

$$f_j(x) = \sum_{j=1}^{n} c_{ji} x_j, \quad j = 1, ..., k \tag{4.4.13}$$

$$\Omega = \{ x \,|\, Ax = b, x \geq 0 \} \tag{4.4.14}$$

where $A = (a_{ij})_{m\times n}$ is $m \times n$ matrix and $b = (b_i)_{m\times 1}$ is a vector. First, construct a payoff table as shown in Table 4.2, where $\bar{x}^j$ is the optimal solution obtained by minimising the objective $f_j(x)$ and $f_j(\bar{x}^j)$ is the minimal value of $f_j(x)$. Suppose f_j^{max} is the maximum (worst) value of $f_j(x)$ in the payoff table. Then, the weighting factors can be calculated as follows

$$\omega_j = \frac{\beta_j}{\sum_{i=1}^{k} \beta_i} \tag{4.4.15}$$

where

$$\beta_j = \left| \frac{f_j^{\max} - f_j(\bar{x}^j)}{f_j^{\max}} \right| \left(\sum_{i=1}^{n} c_{ji}^2 \right)^{-1/2} \tag{4.4.16}$$

The weighting factors defined as above are normalised, that is they satisfy the following conditions

$$0 \le \omega_j \le 1 \quad (j = 1, ..., n) \quad \text{and} \quad \sum_{j=1}^{k} \omega_j = 1 \tag{4.4.17}$$

The weights defined above reflect the impact of the differences of objective values on decision analysis. If the value $\left|(f_j^{\max} - f_j(\bar{x}^j))/f_j^{\max}\right|$ is relatively small, then the objective $f_j(x)$ will be relatively insensitive to the changes of solutions x. In other word, $f_j(x)$ will not play an important role in determining the best compromise solution.

For the weights given by (4.4.15), solve the minimax problem (4.4.10)-(4.4.11). If the decision maker is satisfied with the optimal solution, it can be used as the best compromise solution. Otherwise, the decision maker needs to assess the results, conduct trade-off analysis among the objectives and determine the search direction.

Improving an objective from an efficient solution can only be achieved at the expense of other objectives. This means that the decision maker must sacrifice at least one objective in order to improve another objective. In the STEM method, the decision maker is also expected to estimate the extent to which an objective can be sacrificed.

The interactive process includes two main stages: elicitation of preference information and generation of efficient solutions. Based on the current values of objectives and decision variables, the decision maker has to decide which objective could be sacrificed and to what degree it could be sacrificed. Then, a new constraint will be created to take into account the trade-off analysis. The constraint is then added to the original decision space of problem (4.4.10)-(4.4.12). Solving the updated problem will lead to a new efficient solution.

Suppose the objective $f_l(x)$ could be sacrificed from its current value $f_l(x^i)$ by as much as $\Delta f_l(x^i)$, while all other objectives should be kept at least at

their current levels. Then, a new single optimisation problem can be formulated as follows

$$\begin{aligned} \text{Min} \quad & d_\infty && (4.4.18)\\ s.t. \quad & \omega_j \left(f_j(x) - f_j^*\right) \le d_\infty, \quad j = 1, ..., k; \quad j \ne l && (4.4.19)\\ & x \in \Omega^{i+1} && (4.4.20) \end{aligned}$$

where

$$\Omega^{i+1} = \left\{ x \;\middle|\; \begin{array}{l} f_j(x) \le f_j(x^i), j = 1, ..., k; j \ne l \\ f_l(x) \le f_l(x^i) + \Delta f_l(x^i); x \in \Omega^t \end{array} \right\} \qquad (4.4.21)$$

Solving problem (4.4.18) will result in a new efficient solution x^{i+1}. If the preference information provided by the decision maker is consistent, there should be $x^{i+1} \succ x^i$.

The computational steps of the STEM method are summarised as follows.

S1 Optimise individual objectives to construct a payoff table. Identify the best and the worst values in the table for each objective.

S2 For a linear problem, use (4.4.15)-(4.4.16) to calculate the coefficients β_j and the weighting factors ω_j $(j = 1, ..., k)$. Let $i = 1$.

S3 Solve problem (4.4.10)-(4.4.12), the optimal solution of which is denoted by x^i.

S4 Show the results $f_j(x^i)$ $(j = 1, ..., k)$ to the decision maker, who will provide his judgements about the results. There may be three possible cases.

a) The decision maker is satisfied with all $f_j(x^i)$ $(j = 1, ..., k)$. Then the interactive process is terminated and x^t is the best compromise solution.

b) The decision maker may not be satisfied with some objectives. If $i < k - 1$, go to S5.

c) The decision maker may not be satisfied with some objectives. If $i = k - 1$, then terminate the interactive process and use other methods to search for the best compromise solutions.

S5 The decision maker decides which objective could be sacrificed and by how much, that is he has to select $f_l(x)$ and determine $\Delta f_l(x^i)$. If the decision maker cannot find an objective to sacrifice, then the interactive process will be terminated and other methods have to be used for identifying the best compromise solution. Otherwise, go to S6.

S6 Define a new search space as shown in (4.4.21), let $i = i + 1$ and go to S3.

In S4 and S5, the STEM method may fail to identify the best compromise solution. This does not necessarily mean that the original problem does not have such a solution. The interactive process provided in the STEM method only allows the decision maker to search the efficient frontier in a restricted

and irreversible manner, determining which objective could be sacrificed at an interaction whilst no guideline is provided as to what benefit could be expected following the trade-off. The methods discussed in the rest of the chapter will address these issues.

Example 4.7 Use the STEM method to solve Example 4.2.

Solution: The original problem is given by

$$\begin{array}{ll} \text{Min} & F = [f_1(x), f_2(x)] = [-5x_1 + 2x_2, x_1 - 4x_2] \\ s.t. & x \in \Omega \end{array}$$

where

$$\Omega = \left\{ x \;\middle|\; \begin{array}{l} -x_1 + x_2 \leq 3; x_1 + x_2 \leq 8 \\ x_1 \leq 6; x_2 \leq 4; x_1, x_2 \geq 0 \end{array} \right\}$$

The pay-off table of the problem is as shown in Table 4.4.

Table 4.4: Pay-off table of Example 4.7

	$f_1(x^j)$	$f_2(x^j)$
$x^1 = (6, 0)$	-30	+6
$x^2 = (1, 4)$	3	-15

Since $f_1(x^1) = -30$, $f_1^{max} = 3$, $c_{11} = -5$ and $c_{12} = 2$, from (4.4.16) we have

$$\beta_1 = \left| \frac{3 - (-30)}{-30} \right| \Big/ \left[(-5)^2 + 2^2 \right]^{1/2} = 0.2$$

Similarly, we can get $\beta_2 = 0.34$. From (4.4.15), there are

$$\omega_1 = \frac{0.2}{0.54} = 0.37 \quad \text{and} \quad \omega_2 = \frac{0.34}{0.54} = 0.63$$

Now, we can start the interactive process. Let $i = 0$ and solve the following problem:

$$\begin{array}{ll} \text{Min} & d_\infty \\ s.t. & 0.37(30 - 5x_1 + 2x_2) - d_\infty \leq 0 \\ & 0.63(15 + x_1 - 4x_2) - d_\infty \leq 0 \\ & x \in \Omega^0 = \Omega, \quad d_\infty \geq 0 \end{array}$$

The optimal solution of the problem is

$$\begin{aligned} x^0 &= [x_1^0, x_2^0] = [4.83, 3.17] \\ f(x^0) &= [f_1(x^0), f_2(x^0)] = [-17.9, -1.9] \end{aligned}$$

The results x^0 and $f(x^0)$ are shown to the decision maker. Suppose the solution is not satisfied as $f_2(x^0) = -1.9$ is too large. Suppose $f_1(x)$ can be sacrificed by 3 units, or $\Delta f_1 = 3$. Then, the new search space is given by

$$\Omega^1 = \left\{ x \;\middle|\; \begin{array}{l} f_1(x) = -5x_1 + 2x_2 \leq -17.9 + 3 = -14.9 \\ f_2(x) = x_1 - 4x_2 \leq -1.9; x \in \Omega \end{array} \right\}$$

and the new single objective optimisation problem is

$$\begin{array}{ll} \text{Min} & d_\infty \\ s.t. & 15 + x_1 - 4x_2 - d_\infty \leq 0 \\ & x \in \Omega^1, \quad d_\infty \geq 0 \end{array}$$

The above problem is equivalent to the following problem

$$\begin{array}{ll} \text{Max} & f_2(x) = -x_1 + 4x_2 \\ s.t. & x \in \Omega^1 \end{array}$$

Its optimal solution is given by

$$\begin{aligned} x^1 &= [x_1^1, x_2^1] = [4.41, 3.59] \\ f(x^1) &= [f_1(x^1), f_2(x^1)] = [-14.9, -10] \end{aligned}$$

According to the behavioural assumptions of the STEM method, the decision maker should be satisfied with the solution x^1; otherwise, there would be no best compromise solution. For this two-objective problem, this conclusion may be acceptable as $f_1(x)$ can be sacrificed by as much as 3 units from $f_1(x^0)$ and this sacrifice has been fully used to benefit the objective $f_2(x)$. In general, such conclusion may not be rational for problems having more than two objectives. In such circumstances, whether the decision maker is satisfied with a solution depends on the range of solutions he has investigated. Also, the sacrifices of multiple objectives should also be investigated in addition to the sacrifice of a single objective at each interaction.

4.4.3 The ISTM Method

The Interactive Step Trade-off Method (ISTM) (Yang *et al.*, 1990; Yang and Sen, 1996a,b) provides a learning-oriented interactive procedure where the decision maker can investigate the efficient frontier of a multiple objective optimisation problem by means of implicit trade-off analysis. In ISTM, the decision maker must decide whether an objective needs to be improved, or should be kept at least at the current level, or may be sacrificed by a certain amount. Based on such a trade-off analysis, ISTM will search for a new efficient solution that can satisfy the decision maker's preferences.

The computational steps of the ISTM method can be summarised as follows:

S1 Define a multiple objective optimisation problem as in (4.1.1)-(4.1.3) with "optimisation" replaced by "maximisation".

S2 Use the minimax method to generate an efficient solution x^0 for given values of ω_j $(j = 1, ..., k)$. Let $t = 1$.

S3 Classify the objective functions into the following three subsets:
W - the index subset of objective functions that need to be improved from the current level $f_i(x^{t-1})$,
R - the index subset of objective functions that should be kept at least at the current level $f_j(x^{t-1})$ and
Z - the index subset of objective functions that may be sacrificed from the current level $f_l(x^{t-1})$.
Let

$$\begin{cases} W = \{i | i = i_1, i_2, ..., i_w\} \\ R = \{j | j = j_1, j_2, ..., j_r\} \\ Z = \{l | l = l_1, l_2, ..., l_z\} \end{cases} \tag{4.4.22}$$

where $W \cup R \cup Z = \{1, 2, \cdots, k\}$ and $W \cap R \cap Z = \emptyset$. Suppose $df_l(x^{t-1})$ is the current maximum decrement of $f_l(x)$ $(l \in Z)$, and $df_l(x^{t-1}) \geq 0$.

S4 Suppose u_i $(i \in W)$ is an auxiliary variable, then an auxiliary problem can be defined as follows

$$\text{Max} \quad u = \sum_{i \in W} \sigma_i u_i \tag{4.4.23}$$

$$s.t. \quad x_a \in \Omega_a \tag{4.4.24}$$

$$\Omega_a = \left\{ x_a \;\middle|\; \begin{array}{l} f_i(x) - h_i u_i \geq f_i(x^{t-1}), u_i \geq 0, i \in W \\ f_j(x) \geq f_j(x^{t-1}), j \in R \\ f_l(x) \geq f_l(x^{t-1}) - df_l(x^{t-1}), l \in Z \\ x_a = [x^T u_{i_1} \cdots u_{i_w}]^T \end{array} \right\} \tag{4.4.25}$$

where u_i is maximised to improve $f_i(x)$ as greatly as possible, u the auxiliary objective function, and σ_i a positive weighting factor which is determined according to the relative importance of the objective functions in the subset W. Normally, we let $\sigma_i = 1$ $(i \in W)$. h_i is a normalising factor that can be given by

$$h_i = \left| f_i^* - f_i^- \right| \tag{4.4.26}$$

where f_i^* and f_i^- are the best and worst values of $f_i(x)$ in the pay-off table.

S5 Solve the auxiliary problem (4.4.23). The optimal solution is denoted by x^t, which is guaranteed to be a new (weakly) efficient solution of the original problem.

S6 If the decision maker is not satisfied with x^t, let $t = t + 1$ and go to S3. The interactive process is terminated if a) no objective is required to improve, or b) no objective is allowed to be sacrificed.

Figure 4.10 illustrates how the ISTM method works and the interpretation of the trade-offs. In Figure 4.10, the feasible objective space of a problem with two objectives is denoted by $F(\Omega)$. The search space Ω_a of the auxiliary problem (4.4.23) defined at x^{t-1} (point A) is the shaded area within the original feasible objective space. If objective l needs to be improved from its value at point A, this can only be done at the expense of objective k. If S is the limiting value of sacrifice for objective k, the new solution is B. This trade-off also implies a change in relative importance of the two objectives denoted by a shift from the tangent line at point A to the tangent line at point B.

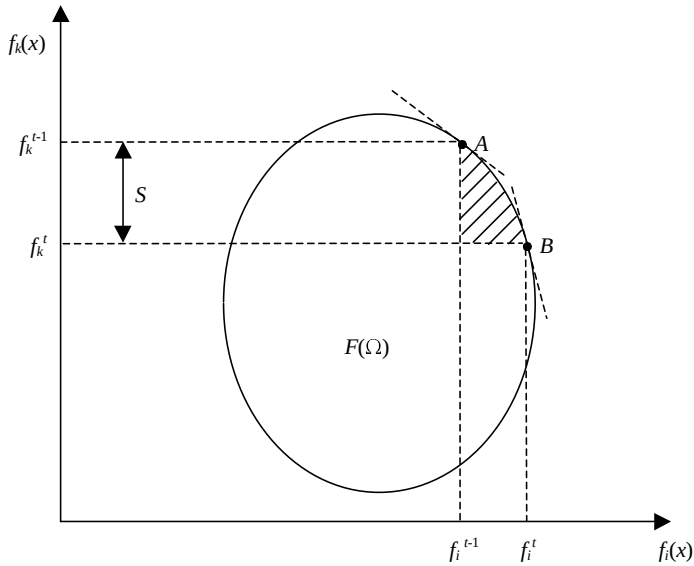

Figure 4.10: Trade-off analysis of ISTM

Example 4.8 Use the ISTM method to solve the following three-objective optimisation problem:

$$\begin{aligned} \text{Max} \quad & F(x) = \left\{ \begin{array}{l} f_1(x) = x_1 - x_2 + x_3 \\ f_2(x) = -x_1 + 2x_2 + 3x_3 \\ f_3(x) = x_1 + 4x_2 - x_3 \end{array} \right. \\ s.t. \quad & x \in \Omega \\ & \Omega = \left\{ x \;\middle|\; \begin{array}{l} 2x_1 + x_2 + x_3 \leq 1 \\ x_1 + 3x_2 + x_3 \leq 1 \\ x_1 + x_2 + 4x_3 \leq 1 \end{array} \;,\; x_1, x_2, x_3 \geq 0 \right\} \end{aligned}$$

Solution: First, optimise each of the three objectives to generate the pay-off table as shown in Table 4.5.

Table 4.5: Pay-off table of Example 4.8

	$f_1(\bar{x}^i)$	$f_2(\bar{x}^i)$	$f_3(\bar{x}^i)$
$\bar{x}^1$	0.5714	0.0	0.2857
$\bar{x}^2$	-0.0909	1.0909	0.909
$\bar{x}^3$	-0.3333	0.6666	1.3332

Then, the normalising factors h_i are given by

$$[h_1, h_2, h_3] = [0.9047, 1.0909, 1.0476]$$

Suppose the relative weights of the three objectives are

$$[\omega_1, \omega_2, \omega_3] = [0.3316, 0.3667, 0.2804]$$

Now, we can use the minimax method to find an initial efficient solution. The minimax problem (see (4.3.22)-(4.3.23)) can be written as follows

$$\begin{aligned} \text{Min} \quad & d_\infty \\ s.t. \quad & x_w \in \Omega_w \\ & \Omega_w = \left\{ x_w \;\middle|\; \begin{array}{l} x_1 - x_2 + x_3 + 3.0157 d_\infty \geq 0.5714 \\ -x_1 + 2x_2 + 3x_3 + 2.727 d_\infty \geq 1.099 \\ x_1 + 4x_2 - x_3 + 3.496 d_\infty \geq 1.333 \\ x \in \Omega, d_\infty \geq 0 \end{array} \right\} \end{aligned}$$

Solving the problem results in the following initial efficient solution

$$\begin{aligned} & x^0 = [0.1761,\ 0.2008,\ 0.1558] \\ & d_\infty = 0.146, \quad F(x^0) = [0.1311,\ 0.6929,\ 0.8235] \end{aligned}$$

To conduct the first interaction, construct the first interaction table (Table 4.6).

Table 4.6: First interaction

$f_1(\bar{x}^0)$	$f_2(\bar{x}^0)$	$f_3(\bar{x}^0)$
0.1311	0.6923	0.8235
W	Z	Z
	$df_2(\bar{x}^0) = 0.1$	$df_2(\bar{x}^0) = 0.1$

The trade-offs as shown in Table 4.6 mean that the decision maker wishes to improve $f_1(x)$ at the expense of $f_2(x)$ and $f_3(x)$, and both $f_2(x)$ and $f_3(x)$

could be reduced by 0.1 units. The first auxiliary problem is constructed as follows

$$\begin{aligned} \text{Max} \quad & u_1 \\ s.t. \quad & x_a \in \Omega_0 \\ & \Omega_0 = \left\{ x_a \;\middle|\; \begin{array}{l} x_1 - x_2 + x_3 - 0.9047u_1 \geq 0.1311 \\ -x_1 + 2x_2 + 3x_3 \geq 0.5929 \\ x_1 + 4x_2 - x_3 \geq 0.7235 \\ x_a = [x^T, u_1]^T, x \in \Omega, u_1 \geq 0 \end{array} \right\} \end{aligned}$$

Solving the problem leads to the following new efficient solution

$$\begin{aligned} & x^1 = [0.2094, \quad 0.1675, \quad 0.1558] \\ & u_1 = 0.07362, \quad F(x^1) = [0.1977, \quad 0.5929, \quad 0.7235] \end{aligned}$$

Suppose the decision maker is not happy with $f_1(x)$ and $f_2(x)$ and $f_3(x)$ can be reduced further. Then, construct the second interaction table (Table 4.7)

Table 4.7: First interaction

$f_1(\bar{x}^1)$	$f_2(\bar{x}^2)$	$f_3(\bar{x}^3)$
0.1977	0.5929	0.7235
W	Z	Z
	$df_2(\bar{x}^1) = 0.2$	$df_2(\bar{x}^1) = 0.1$

The second auxiliary problem is then constructed as follows

$$\begin{aligned} \text{Min} \quad & u_1 \\ s.t. \quad & x_a \in \Omega_1 \\ & \Omega_1 = \left\{ x_a \;\middle|\; \begin{array}{l} x_1 - x_2 + x_3 - 0.9047u_1 \geq 0.1977 \\ -x_1 + 2x_2 + 3x_3 \geq 0.3929 \\ x_1 + 4x_2 - x_3 \geq 0.6235 \\ x_a = [x^T, u_1]^T, x \in \Omega, u_1 \geq 0 \end{array} \right\} \end{aligned}$$

The optimal solution of the problem is given by

$$\begin{aligned} & x^2 = [0.29, \quad 0.1202, \quad 0.1457] \\ & u_1 = 0.132, \quad F(x^2) = [0.3173, \quad 0.3929, \quad 0.6236] \end{aligned}$$

Suppose the decision maker is satisfied with $F(x^2)$. Then, x^2 is the best compromise solution.

4.4.4 The Gradient Projection Method

The lack of a rigorous termination criterion is a common problem associated with most interactive methods. The gradient projection method (Yang, 1999)

was proposed to address the problem. Similar to Geoffrion's method, the gradient projection method uses marginal rates of substitution (utility gradient) to represent the decision maker's preferences. However, the method uses the projection of the gradient onto the tangent plane of the efficient frontier to identify the trade-off direction (Li and Yang, 1996; Yang, 1999; Li *et al.*, 1999). The interaction process is terminated when the projection is zero, indicating that the local minimum of the underlying utility function is reached.

The calculation steps of the gradient projection method can be summarised as follows:

S1 Define a multiple objective optimisation problem as shown in (4.1.1)-(4.1.3) and assign the best and worst values for each objective by constructing for example the pay-off table.

S2 Generate an initial efficient solution x^0. For instance, use the minimax method to generate an initial efficient solution, assuming that all objectives are of equal importance. Let $t = 1$.

S3 At x_{t-1}, calculate the normal vector N^{t-1} as follows. Let

$$\omega_i^{t-1} = \frac{1}{|f_i^* - f_i(x^{t-1})|} \quad (i = 1, ..., k) \tag{4.4.27}$$

Then, the normal vector of the efficient frontier in the objective space at the point $F(x^{t-1})$ is give by

$$N^{t-1} = [\omega_1^{t-1}\lambda_1^{t-1} \cdots \omega_i^{t-1}\lambda_i^{t-1} \cdots \omega_k^{t-1}\lambda_k^{t-1}]^T \tag{4.4.28}$$

where λ^{t-1} is the solution of the following linear equations

$$\sum_{i=1}^{k} \lambda_i = 1 \tag{4.4.29}$$

$$-\sum_{i=1}^{k} \omega_i^{t-1} \frac{\partial f_i(x^{t-1})}{\partial x_l} \lambda_i + \sum_{j \in J_g} \frac{\partial g_j(x^{t-1})}{\partial x_l} \beta_j + \sum_{p=1}^{m_2} \frac{\partial h_p(x^{t-1})}{\partial x_l} \gamma_p = 0, \quad l = 1, ..., n \tag{4.4.30}$$

with $\lambda_i, \beta_j, \gamma_p \geq 0$, $i = 1, ..., k$, $j = 1, ..., m_1$, $p = 1, ..., m_2$ and $\beta_j = 0$ for $j \notin J_g$, where $J_g = \{j | g_j(x^{t-1}) = 0, j \in \{1, ..., m_1\}\}$.

S4 Articulate the decision maker's indifference trade-offs σ_i^{t-1} for $i = 1, ..., k$ using the following approximation

$$\sigma_l^{t-1} = 1; \quad \sigma_i^{t-1} \approx -\frac{\Delta f_l^{t-1}}{\Delta f_i^{t-1}}, \quad i = 1, ..., k; \ i \neq l \tag{4.4.31}$$

where Δf_l is a small change in f_l that can be exactly offset by a change Δf_i in f_i (i.e. the utility function is kept constant) while all other objectives remain unchanged. The best compromise solution maximising

an underlying utility function is reached if the following conditions are met

$$\Delta f_i^{t-1} = -\Delta f_l^{t-1} \frac{N_l^{t-1}}{N_i^{t-1}}, i = 1, ..., k; i \neq l \tag{4.4.32}$$

Given a unit change in f_l, or $\left|\Delta f_l^{t-1}\right| = 1$, Δf_i^{t-1} defined by (4.4.32) is referred to as an optimal indifference trade-off between f_i and f_l. (4.4.32) can be used to guide the decision maker to provide his indifference trade-offs.

S5 Calculate the projection $d\bar{F}^{t-1}$ of σ^{t-1} as follows

$$d\bar{F}^{t-1} = [d\bar{f}_1^{t-1} \cdots d\bar{f}_k^{t-1}] = \sigma^{t-1} - \frac{\left[(\sigma^{t-1})^T N^{t-1}\right]}{\left[(N^{t-1})^T N^{t-1}\right]} N^{t-1} \tag{4.4.33}$$

If $d\bar{F}^{t-1} = 0$, condition (4.4.32) is met, the best compromise solution is obtained and stop the interactive process. Otherwise, go to S6.

S6 Assign the step size $\hat{\alpha}_l^{t-1}$ using the following equations and table. Initially let $\alpha_2 = 1$.

$$\bar{\alpha}_{\max}^{t-1} = \min_{i \in \bar{I}_2} \left\{\bar{\alpha}_i^{t-1}\right\}, \quad \bar{\alpha}_i^{t-1} = \frac{f_i^{t-1} - f_i^-}{\left|d\bar{f}_i^{t-1}\right|}, i \in \bar{I}_2 \tag{4.4.34}$$

Table 4.8: Trade-off table for assignment of step size $\bar{\alpha}^{t-1}$

	$f_j (j \in I_2)$			$f_i (i \in I_1)$		
$\hat{\alpha}_l^{t-1}$	...	$f_j^{t-1} - \alpha_l^{t-1} \lvert df_j^{t-1}\rvert$	...	...	$f_i^{t-1} + \alpha_l^{t-1} \lvert df_i^{t-1}\rvert$	...
$\hat{\alpha}_0^{t-1}$	...	$f_j(\hat{\alpha}_0^{t-1})$	...	...	$f_i(\hat{\alpha}_0^{t-1})$	...
$\hat{\alpha}_1^{t-1}$	...	$f_j(\hat{\alpha}_1^{t-1})$	...	...	$f_i(\hat{\alpha}_1^{t-1})$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\hat{\alpha}_{C_\alpha}^{t-1}$	...	$f_j(\hat{\alpha}_{C_\alpha}^{t-1})$	...	...	$f_i(\hat{\alpha}_{C_\alpha}^{t-1})$	...

where $\bar{I}_1 = \{i | d\bar{f}_i^{t-1} \geq 0 \quad i \in \{1, \cdots, k\}\}$, $\bar{I}_2 = \{i | d\bar{f}_i^{t-1} < 0, \quad i \in \{1, ..., k\}\}$, C_α is an integer and f_i^- is the worst permissible value of objective i. The step size $\hat{\alpha}_l^{t-1}$ may thus be taken as a value of the interval $[0, \bar{\alpha}_{\max}^{t-1}]$ by the decision maker from the above table.

S7 Let $\bar{\alpha}^{t-1} = \alpha_2 \hat{\alpha}_l^{t-1}$ and define a local region by

$$\Omega_a = \left\{ x_a \;\middle|\; \begin{array}{l} f_i(x) \geq f_i(x^{t-1}) - \Delta f_i^{t-1} + y_i \\ y_i \geq 0, i = 1, ..., k; x \in \Omega \end{array} \right\} \tag{4.4.35}$$

$$\Delta f_i^{t-1} = \frac{\bar{\alpha}^{t-1}}{2} \left(\left|d\bar{f}_i^{t-1}\right| - d\bar{f}_i^{t-1}\right) i = 1, ..., k \tag{4.4.36}$$

Then, construct and solve the following auxiliary problem, yielding a new efficient solution x^t.

$$\text{Max} \quad \sum_{i=1}^{k} \sigma_i^{t-1} y_i \tag{4.4.37}$$

$$s.t. \quad x_a \in \Omega_a \quad x_a = [x^T, y_1, ..., y_k]^T \tag{4.4.38}$$

S8 If x^t is preferred to x^{t-1} (thus $u(x^t) \geq u(x^{t-1})$), let $t = t+1$ and go to S3. Otherwise, let $\alpha_2 = \alpha_2/2$ and go to S7.

Note that the above interactive process is not necessarily irreversible as the optimality condition could be tested separately at each generated efficient solution, independent of other solutions generated before. In other words, the process does not necessarily require that the utility function always increase as the process proceeds. Rather, the process can be terminated at any individual point by assessing the optimal indifference trade-offs defined by (4.4.32). Such flexibility preserves the favourable features of the ISTM method and leads to a progressive and explicit articulation of the decision maker's priorities. However, the assumption that the utility function increases monotonically in the interactive process guarantees the convergence of the process.

Example 4.9 Use the gradient projection method to solve Example 4.2 defined as follows

$$\text{Max} \quad F(x) = \left\{ \begin{array}{l} f_1(x) = 5x_1 - 2x_2 \\ f_2(x) = -x_1 + 4x_2 \end{array} \right\}$$

$$s.t. \quad x \in \Omega \quad x = [x_1, x_2]^T$$

$$\Omega = \left\{ x \;\middle|\; \begin{array}{l} -x_1 + x_2 \leq 3; x_1 + x_2 \leq 8 \\ x_1 \leq 6; x_2 \leq 4; x_1, x_2 \geq 0 \end{array} \right\}$$

Solution: An initial efficient solution can be generated using, for example, the minimax method. Suppose the starting point is given by $x^0 = [2, 4]^T$. Then, we can have

$$F^0 = [2, 14]^T, \quad N^0 = \frac{1}{33}[1, 15]^T$$

Suppose $f_2(x)$ is treated as the reference objective. If the following indifference trade-off is provided: "A unit change in $f_2(x)$ is exactly offset by a change of $1/20$ units in f_1 at x^0", or

$$[2, 14]^T \Leftrightarrow [2 - \frac{1}{20}, 14 + 1]^T$$

("$\Leftrightarrow$" reads "is indifferent to"), then the normalised gradient of the utility function at x^0 can be estimated by

$$\sigma^0 = [\sigma_1^0, \sigma_2^0]^T \approx [-\frac{1}{-1/20}, 1]^T = [20, 1]^T$$

The projection of σ^0 onto the tangent plane at $F(x^0)$ is then given by

$$d\bar{F}^0 = \sigma^0 - \frac{\left[(\sigma^0)^T N^0\right]}{\left[(N^0)^T N^0\right]} N^0 = [19.038, -3.808]^T$$

Thus, the utility function could be improved from $u(x^0)$ by increasing f_1 at the expense of f_2, or $\bar{I}_1 = \{1\}$ and $\bar{I}_2 = \{2\}$. The amount in which f_2 could be sacrificed is determined as follows. The maximum permissible amount of sacrifice in f_2 and the maximum step size are given by

$$\Delta f_2^0 = f_2^0 - f_2^- = 14 - (-6) = 20$$
$$\bar{\alpha}^0_{\max} = \frac{\Delta f_2^0}{\left|d\bar{f}_2^0\right|} = \frac{20}{3.808} = 5.252$$

Table 4.9: Assignment of step size α^0 given $C_\alpha = 10$

l	$\hat{\alpha}_l^{t-1}$	f_2	f_1
		$f_j^{t-1} - \alpha_l^{t-1}\lvert df_j^{t-1}\rvert$	$f_i^{t-1} + \alpha_l^{t-1}\lvert df_i^{t-1}\rvert$
0	0	14	2
1	0.5252	12	12
2	1.0504	<u>10</u>	<u>22</u>
3	1.5756	<u>8</u>	<u>32</u>
4	2.1008	6	42
5	2.6260	4	52
6	3.1512	2	62
7	3.6764	0	72
8	4.2016	-2	82
9	4.7268	-4	92
10	5.2520	-6	102

Let $C_\alpha = 10$. $\bar{\alpha}^0$ can then be assigned using Table 4.9. In Table 4.9, one can find that the increase of f_1 along the tangent plane at $F(x^0)$ is much faster than along the efficient frontier. This is because the maximum feasible value of f_1 (i.e. 30) has already been exceeded at $l = 3$ for $C_\alpha = 10$ while f_2 is only reduced to 8. In this case, $\hat{\alpha}_2^0 = 1.0504$ may be used as the step size. If the decision maker wishes to find a step size such that f_1 is nearer its maximum feasible value 30, C_α could be increased to 100 and Table 4.10 is thus constructed. In Table 4.10, $\bar{\alpha}^0 = \hat{\alpha}_{28}^0 = 1.47056$ is taken as the current step size.
Given $\bar{\alpha}^0 = \hat{\alpha}_{28}^0$, an auxiliary problem can be constructed as follows

$$\begin{aligned} \text{Max} \quad & \left(\sigma_1^0 y_1 + \sigma_2^0 y_2\right) \\ s.t \quad & x_a \in \Omega_a \end{aligned}$$

Table 4.10: Assignment of step size α^0 given $C_\alpha = 100$

l	$\hat{\alpha}_l^{t-1}$	f_2	f_1
		$f_j^{t-1} - \alpha_l^{t-1}\lvert df_j^{t-1}\rvert$	$f_i^{t-1} + \alpha_l^{t-1}\lvert df_i^{t-1}\rvert$
20	1.05040	10	22
21	1.10292	9.8	23
22	1.15544	9.6	24
23	1.20796	9.4	25
24	1.26048	9.2	26
25	1.31300	9.0	27
26	1.36552	8.8	28
27	1.41804	8.6	29
28	1.47056	8.4	30
29	1.52308	8.2	31
30	1.57560	8.0	32

$$\Omega_{\mathrm{a}} = \left\{ x_a \;\middle|\; \begin{array}{l} f_1(x) \geq f_1^0 + y_1 \\ f_2(x) \geq f_2^0 - \alpha_2 \hat{\alpha}_{28}^0 \left| d\bar{f}_2^0 \right| + y_2 \\ x \in \Omega; y_1, y_2 \geq 0 \end{array} \right\}$$

Let $\alpha_2 = 1$. Then, we have

$$\begin{array}{ll} \text{Max} & (20\mathrm{y}_1 + \mathrm{y}_2) \\ s.t & x_a \in \Omega_a \\ & \Omega_{\mathrm{a}} = \left\{ x_a \;\middle|\; \begin{array}{l} f_1(x) \geq 2 + y_1 \\ f_2(x) \geq 8.4 + y_2 \\ x \in \Omega; y_1, y_2 \geq 0 \end{array} \right\} \end{array}$$

The optimal solution of the above problem is given by

$$x^1 = [x_1^1, x_2^1]^T = [4.72,\ 3.28]^T, F^1 = [f_1^1, f_2^1]^T = [17.04,\ 8.4]^T$$

which is an efficient solution of the original problem. Suppose the decision maker still prefers x^1 to x^0 (thus $u(x^1) \geq u(x^0)$) although f_1^1 is smaller than expected. In this case, α_2 need not be reduced. If this were not the case, α_2 would need to be reduced and the process would then be repeated, resulting in another efficient solution. This completes the first interaction.

In the second interaction, the normal vector at F^1 can be obtained by $N^1 = 0.063[1/1.4, 1]^T$. Suppose the decision maker provides the following indifference trade-off at F^1

$$[17.04,\ 8.4]^T \Leftrightarrow [17.04 - 1,\ 8.4 + 1]^T$$

Then, it is easy to show that the optimal condition is not satisfied. In fact,

$$\sigma^1 = [\sigma_1^1,\ \sigma_2^1]^T = [1, 1]^T,\ d\bar{F}^1 = 0.135[1.4, -1]^T$$

Hence, the utility function can still be improved from $u(x^1)$ by increasing f_1 at the expense of f_2, or $\bar{I}_1 = \{1\}$ and $\bar{I}_2 = \{2\}$. The new maximum permissible amount of sacrifice in f_2 and the new maximum step size are given by

$$\Delta f_2^1 = 14, \quad \bar{\alpha}_{\max}^1 = 106.667$$

Table 4.11: Assignment of step size α^0 given $C_\alpha = 10$

l	$\hat{\alpha}_l^{t-1}$	f_2	f_1
		$f_j^{t-1} - \alpha_l^{t-1}\lvert df_j^{t-1}\rvert$	$f_i^{t-1} + \alpha_l^{t-1}\lvert df_i^{t-1}\rvert$
0	0	8.40	17.040
1	10.6667	6.96	19.056
2	21.3334	5.52	21.072
3	32.0001	4.08	23.088
4	42.6668	2.64	25.104
5	53.3335	1.20	27.120
6	64.0002	-0.24	29.136
7	74.6669	-1.68	31.152
8	85.3336	-3.12	33.168
9	96.0003	-4.56	35.184
10	106.667	-6.00	37.200

The step size may then be determined using Table 4.11 and Table 4.12. Table 4.12 is constructed because the decrease of f_2 will no longer be offset by the increase of f_1 when $l \geq 3$ for $C_\alpha = 10$ but f_2 can still be decreased from the value of 5.52. $\alpha_{27}^1 = 28.8$ is taken as the current step size as 4.5 is regarded as the acceptable lower bound of f_2.
Then, an auxiliary problem can be constructed as follows, given $\alpha_2 = 1$

$$\begin{aligned} \text{Max} \quad & (y_1 + y_2) \\ s.t \quad & x_a \in \Omega_a \\ & \Omega_a = \left\{ x_a \;\middle|\; \begin{array}{r} f_1(x) \geq 17.04 + y_1 \\ f_2(x) \geq 4.512 + y_2 \\ x \in \Omega; y_1, y_2 \geq 0 \end{array} \right\} \end{aligned}$$

The optimal solution of the above problem is given by

$$\begin{aligned} x^2 &= [x_1^2, x_2^2]^T = [5.498, 2.502]^T \\ F^2 &= [f_1^2, f_2^2]^T = [22.283, 4.512]^T \end{aligned}$$

By examining F^2, it is clear that the actual achievement levels of f_1 and f_2 are both the same as expected in Table 4.12. This is because $d\bar{F}^1$ is on the efficient frontier. Thus, α_2 need not be reduced.

Table 4.12: Assignment of step size α^0 given $C_\alpha = 100$

l	$\hat{\alpha}_l^{t-1}$	f_2	f_1
		$f_j^{t-1} - \alpha_l^{t-1}\lvert df_j^{t-1}\rvert$	$f_i^{t-1} + \alpha_l^{t-1}\lvert df_i^{t-1}\rvert$
20	21.33340	5.520	21.072
21	22.40007	5.376	21.274
22	23.46674	5.232	21.475
23	24.53341	5.088	21.677
24	25.60008	4.944	21.878
25	26.66675	4.800	22.107
26	27.73342	4.656	22.282
27	28.80009	4.512	22.483
28	29.86667	4.368	22.685
29	30.93343	4.224	22.886
30	32.00010	4.080	23.088

In the third interaction, the normal vector at F^2 is given by

$$N^2 = 0.063[1/1.4, 1]^T$$

The decision maker is expected to provide the following optimal indifference trade-off at F^2

$$[22.483, 4.512]^T \Leftrightarrow [22.483 - 1.4, 4.512 + 1]^T$$

If the decision maker accepts this trade-off, then the optimal condition is satisfied. This is because

$$\sigma^2 = [\sigma_1^2, \sigma_2^2]^T = [-\frac{1}{-1.4}, 1]^T = \frac{1}{0.063}N^2$$

Otherwise, he should provide another indifferent trade-off and the interactive process will then continue until the optimality condition is satisfied.

4.5 Summary

In this chapter, we first discussed the basic concepts and methods in multiple objective optimisation. Common to any multiple objective optimisation problems are their non-commensurability and conflict among objectives. Consequently, in such problems there is no single solution that could optimise all objectives simultaneously. What can be found are non-dominated (efficient, non-inferior or Pareto-optimal) solutions, among which the best compromise solutions should be sought. Three types of methods were discussed in this chapter. The simple weighting method, although widely used, is only applicable to convex problems with smooth efficient frontiers. In this chapter,

this method was described for generating efficient solutions for illustration purposes.

Four multiple objective optimisation methods based on p-norm were discussed by assuming the availability of global preference information. The minimax (ideal point) method is a widely used approach and forms a basis for several other multiple objective optimisation methods. The goal attainment method is one of the variants of the minimax method, where the canonical weights are used to represent decision maker's preferences. Goal programming provides more flexible ways for accommodating different types of preference information, though it is only applicable to convex problems. The minimax reference point method facilitates goal programming in non-convex cases and also provides a basis for generating efficient solutions on both convex and non-convex efficient frontiers.

Interactive methods are desirable in decision situations where there is little *a priori* knowledge about optimisation problems in hand, for example problems of designing new systems or products. Among the four interactive methods, Geoffrion's method is the earliest, which is applicable to convex problems. The STEM method is based on the formulation of the minimax method and provides a systematic procedure to converge to the best compromise solutions. However, it does not guarantee to produce the best compromise solutions. The ISTM method provides a flexible way for implicit trade-off analysis and allows the decision maker to explore the efficient frontier in a natural manner, though it does not provide a theoretical criterion for terminating the interactive process. The gradient projection method provides a rigorous termination criterion whilst also preserving the favourable features of the ISTM method and Geoffrion's method. However, it does require extra effort to identify normal vectors for calculating utility gradient projections.

Chapter 5

Genetic Algorithms and Optimisation

5.1 Introduction

Many different techniques are used today for optimising the design space associated with various control systems. Most of these techniques can broadly be classified under either calculus-based techniques or direct-search methods. However, the calculus-based methods lack robustness over the broad spectrum of optimisation functions that arise in engineering optimisation. In recent years, the direct-search techniques, which are problem-independent, have been proposed as a panacea for the difficulties associated with the traditional techniques. One of these techniques is genetic algorithms (GAs) (Goldberg, 1989; Davis, 1991).

The genetic algorithm filed has three major thrusts: research into the basic genetic algorithm, optimisation using genetic algorithm, and machine learning with classifier systems. The research thrust is well described in Goldberg (1989). The classifier system work is also described in Goldberg's text. This chapter does not deal with those two areas. Instead, it is directed to the optimisation field, and its goal is to introduce some concepts about genetic algorithms and show how to apply them effectively to optimisation problems.

5.2 What are Genetic Algorithms

Genetic algorithms are invented by simulating some of the processes observed in natural evolution. Biologists have been intrigued with the mechanism of evolution since the evolutionary theory of biological change was accepted. Many people are astonished that life at the level of complexity could have evolved in the relatively short time suggested by the fossil record. The mech-

anisms that drive this evolution are not fully understood, but some of its features are known. Evolution takes place on chromosomes, which are organic devices for encoding the structure of living beings. A living being is partly created through a process of decoding chromosomes.

Although the specificities of chromosomal encoding and decoding processes are not fully known, the following general features of the evolution theory are widely accepted: a) Evolution process operates on chromosomes rather than on the living beings which they encode. b) Natural selection process causes the chromosomes that encode successful structures to reproduce more often than ones that do not. c) Reproduction process is the point at which evolution takes place. Recombination process may create quite different chromosomes in the children by combining material from the chromosomes of two parents. Mutations may result in the chromosomes of biological children to be different from those of their biological parents, d) Biological evolution has no memory. Whatever it knows about producing individuals that will function well in their environment is contained in the gene pool, which is the set of chromosomes carried by the current individuals, and in the structure of the chromosome decoders.

In the early 1970s, the above features of natural evolution intrigued the scientist John Holland (1975). He believed that it might yield a technique for solving difficult problems to appropriately incorporating these features in a computer algorithm in the way that nature has done through evolution. So, he began the research on algorithms that manipulated strings of binary digits (1s and 0s) that represent chromosomes. Holland's algorithms carried out simulated evolution on populations of such chromosomes. Using simple encodings and reproduction mechanisms, his algorithms displayed complicated behaviour and solved some extremely difficult problems. Like nature, they knew nothing about the type of problems they were solving. They were simple manipulators of simple chromosomes. When the descendants of those algorithms are used today, it is found that they can evolve better designs, find better schedules and produce better solutions to a variety of other important problems that we cannot solve using other techniques.

When Holland first began to study these algorithms, they did not have a name. As these algorithms began to demonstrate their potential, however, it was necessary to give them a name. In reference to their origins in the study of genetics, Holland named them genetic algorithms. After a great amount of research work in this field was carried out, the genetic algorithms have been developed. Now, the genetic algorithm is a stochastic global search method that mimics the metaphor of natural biological evolution. Applying the principle of survival of the fittest to produce better and better approximations to a solution, genetic algorithms operate on a population of potential solutions. A new set of approximations at each generation is created by the process of selecting individuals, which actually are chromosomes in GAs, according to their fitness level in the problem domain and breeding them together using

operators borrowed from natural genetics, for example, crossover and mutation. This process results in the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

5.3 Basic Structure of Genetic Algorithms

It is well known that natural phenomena can be abstracted into an algorithm in many ways. Similarly, there are a number of ways to embody the preceding features of the theory of natural evolution in genetic algorithms. To begin with, let us consider two mechanisms that link a genetic algorithm to the problem it is solving. One is the way of encoding solutions to the problem on chromosomes and the other is the evaluation function that returns a measurement of the worth of any chromosome in the context of the problem.

The way of encoding solutions plays an important role in genetic algorithms. The technique for encoding solutions may vary from problem to problem and from genetic algorithm to genetic algorithm. In early genetic algorithms, encoding is carried out using bit strings. Later, genetic algorithm researchers have developed many other types of encoding techniques. Probably no one technique works best for all problems, and a certain amount of skill is involved in selecting a good decoding technique when a problem is being studied. Thus, when selecting a representation technique in the context of a real-world problem, several factors should be considered.

The evaluation function is the link between the genetic algorithm and the problem to be solved. An evaluation function takes a chromosome as input and returns a number or list of numbers that is a measure of the chromosome's performance. Evaluation functions play the same role in genetic algorithms as the environment plays in natural evolution. The interaction of an individual with its environment gives a measure of its fitness, and the interaction of a chromosome with an evaluation function provides a measure of fitness that the genetic algorithm uses when carrying out reproduction.

It is assumed that the following initial components are given: a problem, a way of encoding solutions to it, and a function that returns a measure of how good any encoding is. We can use a genetic algorithm to carry out simulated evolution on a population of solutions. Here is the basic structure of genetic algorithms that uses these components to simulate evolution:

a) Initialise a population of chromosomes.
b) Evaluate each chromosome in the population.
c) Create new chromosomes by mating current chromosomes.
d) Remove some members of the population to make room for the new chromosomes.
e) Insert the new chromosomes into the population.
f) Stop and return the best chromosome if time is up, otherwise, go to c).

Following the above structure, a pseudo-code outline of genetic algorithms is shown below. The population of chromosomes at time t is represented by the time-dependent variable $P(t)$, with the initial population of random estimates $P(0)$.

```
procedure GA
begin
    t=0;
    initialise P(t) = P(0);
    evaluate P(t);
    while not finished do
    begin
        t=t+1;
        select P(t) from P(t − 1);
        reproduce pairs in P(t) by
        begin
            crossover;
            mutation;
            reinsertion;
        end
        evaluate P(t);
    end
end
```

If all goes well through this process of simulated evolution, an initial population of unexceptional chromosomes will improve as the chromosomes are replaced by better and better ones. The best individual in the final population produced can be a highly evolved solution to the problem.

The genetic algorithm differs substantially from more traditional search and optimisation methods, for example, gradient-based optimisation. The most significant differences are the following.

a) GAs search a population of points in parallel rather than a single point.

b) GAs do not require derivative information on an objective function or other auxiliary knowledge. Only the objective function and corresponding fitness levels influence the directions of search.

c) GAs use probabilistic transition rules, not deterministic ones.

d) GAs can work on different encodings of the parameter set rather than the parameter set itself.

It is important to note that the GA provides many potential solutions to a given problem and the choice of the final solution is left to the designer. In cases where a particular optimisation problem does not have one individual solution, then the GA is potentially useful for identifying these alternative solution simultaneously.

5.4 Population Representation and Initialisation

Genetic algorithms operate on a number of potential solutions, called a population, which consists of some encoding of the parameter set simultaneously. Generally speaking, a population is composed of 21 individuals or slightly more. Sometimes a variant called 'the micro GA' uses very small populations (less than 10 individuals) with a restrictive reproduction and replacement strategy in an attempt to reach real-time execution (Karr, 1991). There are various types of chromosome representations in GAs. Two representations are introduced here: the binary representation and the real-valued representation.

5.4.1 Binary Representation

The most commonly used representation of chromosomes is that of the single-level binary string. In this representation, each chromosome is encoded as a binary string, for example,

1 0 1 1 0 1 1 0

The use of Gray coding has been advocated as a method that can overcome the hidden representational bias in conventional binary representation as the Hamming distance between adjacent values is constant (Holstien, 1971). Empirical evidence of Caruana and Schaffer (1988) points out that large Hamming distances in the representational mapping between adjacent values can result in the search process being deceived or unable to efficiently locate the global minimum, which is the case in the standard binary representation. A further approach is the use of logarithmic scaling in the conversion of binary-coded chromosomes to their real phenotypic values (Schmitendorgf *et al.*, 1992). The precision of the parameter values is possibly less consistent over the desired range in problems, where the spread of feasible parameters is unknown. But, a larger search space may be covered with the same number of bits than a linear mapping scheme, allowing the computational burden of exploring unknown search spaces to be reduced to a more manageable level.

5.4.2 Real-Valued Representation

Although binary-coded GAs are most commonly used, there is an increasing interest in alternative encoding strategies, such as integer and real-valued

representations. For some problem domains, it is argued that the binary representation is in fact deceptive in that it obscures the nature of the search (Bramlette, 1991). For example, in the subset selection problem (Lucasius and Kateman, 1992), the use of an integer representation and look-up tables provides a convenient and natural way of expressing the mapping from representation to problem domain. Real-valued representation is claimed by Wright (1991) to offer many advantages in numerical function optimisation over binary encodings. It increases efficiency of the GA as there is no need to convert chromosomes to phenotypes before each function evaluation and requires less memory; as efficient floating-point internal computer representations can be used directly. There is no loss in precision by discretisation to binary or other values and there is greater freedom to use different genetic operators. The use of real-valued encodings is detailed by Michalewicz (1992) and in the literature on Evolution Strategies (see, for example, Back *et al.*, 1991).

5.4.3 Initialisation

When the representation of the GA chromosomes has been decided, the first step in the GAs is to create an initial population. The initialisation of the population is usually achieved by generating the required number of individuals using a random number generator that uniformly distributes numbers in the desired range. For example, a binary population of N individuals whose chromosomes are M bits long would be initialised by $N \times M$ random numbers uniformly distributed from the set $\{0, 1\}$. Some other initialisation procedures are also available. In the extended random initialisation procedure of Bramlette (1991), many random initialisations are tried for each individual and the one with the best performance is chosen for the initial population. Another initialisation approach seeds the initial population with some individuals that are known to be in the vicinity of the global minimum (see, for example, Grefenstette, 1987; Whitley *et al.*, 1991). Of course, this approach is only applicable if the nature of the problem is well understood beforehand or if the GA is used in conjunction with a knowledge based system.

5.5 Fitness Functions

The objective function is a measure of how individuals have performed in the problem domain. For a minimisation problem, the most fit individuals will have the smallest numerical value of the associated objective function. This measure is usually only used as an intermediate stage in determining the relative performance of individuals in a GA. Another function, called the fitness function, is normally used to transform the objective function value into a measure of relative fitness (De Jong, 1975), thus:

$$F(c) = f(\phi(c)) \tag{5.5.1}$$

where $\phi(.)$ is the objective function, $f(.)$ converts the value of the objective function to a non-negative number and $F(.)$ is the resulting relative fitness. For the problem with the objective function to be minimised, this mapping is always necessary as the smaller objective function values correspond to fitter individuals. In a number of cases, the fitness function value corresponds to the number of offspring that an individual can expect to produce in the next generation. The proportional fitness assignment (see, for example, Goldberg, 1989) is a widely used transformation. The individual fitness $F(c_i)$ of each individual is computed as the individual's raw performance $F(c_i)$ relative to the whole population, that is,

$$F(c_i) = \frac{\phi(c_i)}{\sum_{i=1}^{N} \phi(c_i)} \tag{5.5.2}$$

where N is the population size and c_i is the phenotypic value of individual i. Although this fitness assignment ensures that each individual has a probability of reproducing according to its relative fitness, it fails to account for negative objective function values.

The following linear transformation, which offsets the objective function (Goldberg, 1989), is commonly used prior to fitness assignment:

$$F(c) = \alpha\phi(c) + \beta \tag{5.5.3}$$

where α is a positive scaling factor if the optimisation is maximising and negative if we are minimising, and the offset β is used to ensure that the resulting fitness values are non-negative.

However, the above linear scaling and offsetting are susceptible to rapid convergence. For most selection algorithms, they select individuals for reproduction on the basis of their relative fitness. Using the linear scaling, the expected number of offspring is approximately proportional to that individual's performance. Since there is no constraint on an individual's performance in a given generation, highly fit individuals in early generations can dominate the reproduction which may cause rapid convergence to possibly sub-optimal solutions. Similarly, if there is little deviation in the population, then the linear scaling provides only a small bias towards the most fit individuals.

It is suggested by Baker (1985) that the premature convergence may be prevented by limiting the reproductive range so that no individuals generate an excessive number of offspring. Here, individuals are assigned a fitness according to their rank in the population instead of their raw performance. One variable B_{ias}, is used to determine the bias, or selective pressure, towards the most fit individuals and the fitness of the others is determined by the following rules:

$$\begin{aligned} L_b &= 2 - B_{ias} \\ D_{if} &= 2(B_{isa} - 1)/N \\ N_{trail} &= D_{if}/2 \end{aligned}$$

where L_b is the lower bound, B_{ias} is typically chosen in the interval [1.1, 2.0], D_{if} is the difference between the fitness of adjacent individuals, N is the population size and N_{trial} is the expected number of trials (number of times selected) of the least fit individual. For example, for a population size of $N = 40$ and $B_{ias} = 1.1$, we obtain $L_b = 0.9$, $D_{if} = 0.05$ and $N_{trial} = 0.025$.

The fitness of individuals in the population may also be calculated directly as,

$$F(c_i) = 2 - B_{ias} + 2(B_{ias} - 1)\frac{c_i - 1}{N - 1} \tag{5.5.4}$$

where c_i is the position in the ordered population of the i-th individual.

5.6 Selection

Selection is the process of determining: the number of times (or trials), a particular individual chosen for reproduction and the number of offspring that an individual will produce. The selection of individuals can be viewed as two following separate processes:

a) determine the number of trials that an individual can expect to receive;

b) convert the expected number of trials into a discrete number of offspring.

The first process takes account of the transformation of raw fitness values into a real-valued expectation of an individual's probability to reproduce and is dealt with in the previous subsection as fitness assignment. The second process is concerned with the probabilistic selection of individuals for reproduction based on the fitness of individuals relative to one another and is sometimes known as sampling.

Three measures of performance for selection algorithms: bias, spread and efficiency, have been presented by Baker (1987). Bias is the absolute difference between an individual's actual and expected selection probability. Optimal zero bias is thus achieved when an individual's selection probability equals its expected number of trials.

Spread is defined as the range in the possible number of trials that an individual may achieve. If $f(i)$ is the actual number of trials that the i-th individual receives, then the "minimum spread" is the smallest spread that theoretically permits zero bias, that is,

$$f(i) \in \{\underline{e}_{nt}(i), \bar{e}_{nt}(i)\} \tag{5.6.1}$$

where $e_{nt}(i)$ is the expected number of trials of the i-th individual, $\underline{e}_{nt}(i)$ is the floor of $e_{nt}(i)$ and $\bar{e}_{nt}(i)$ is the cell. Thus, the bias is an indication of accuracy and the spread of a selection method measures its consistency.

Efficient selection methods are motivated by the need to maintain a GA overall time complexity. It has been shown in the literature that the other phases of a GA (excluding the actual objective function evaluations) have better time complexity. A good selection algorithm should thus achieve zero bias

whilst maintaining a minimum spread and not contributing to an increased time complexity of the GA.

In this section, two commonly used selection methods will be reviewed: Roulette wheel selection methods and stochastic universal sampling.

5.6.1 Roulette Wheel Selection Methods

The roulette wheel selection method is one of the most commonly used selection methods. Many selection techniques in genetic algorithms employ a "roulette wheel" mechanism to probabilistically select individuals based on some measure of their performance. The basic principle of the roulette wheel mechanism is the following.

A real-valued interval S_{um} is defined as either the sum of the individuals' expected selection probabilities or the sum of the raw fitness values over all the individuals in the current population. The individuals are then transformed into contiguous intervals in the range $[0, S_{um}]$ in a one-to-one mapping. The size of each individual interval represents the fitness value of the associated individual. For example, the circumference of the roulette wheel in Figure 5.2 is the sum of all seven individual's fitness values. Individual 6 is the fittest individual and occupies the largest interval, whereas individual 3 is the weakest one and has a correspondingly smaller interval within the roulette wheel. To select an individual, a random number is firstly generated in the interval $[0, S_{um}]$ and then the individual whose segment spans the random number is selected. This process is repeated until a desired number of individuals have been selected.

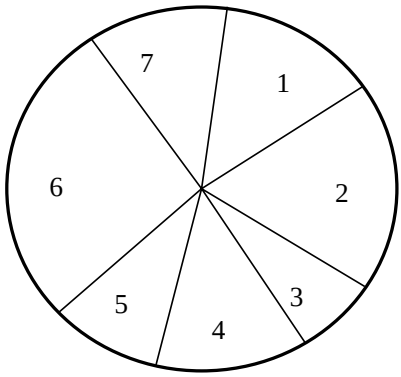

Figure 5.1: Roulette wheel

The basic roulette wheel selection method uses stochastic sampling with replacement (SSR). Here, the segment size and selection probability remain the same throughout the selection phase and individuals are selected using the

procedure outlined above. SSR yields zero bias but a potentially unlimited spread. Any individual with a positive segment size could entirely fill the next population.

The stochastic sampling with partial replacement (SSPR) extends upon the stochastic sampling with replacement by resizing an individual's segment if it is selected. At each time when an individual is selected, the size of its segment is reduced by 1.0. If the segment size becomes negative, then it is set to be 0.0 and there is no more size reduction. Although this provides an upper bound on the spread of $\bar{e}_{nt}(i)$, the lower bound is zero and the bias is higher than that of SSR.

Remainder sampling methods have two distinct phases: the integral phase and the fractional phase. In the integral phase, individuals are selected deterministically according to the integer part of their expected trials. The remaining individuals are then selected probabilistically from the fractional part of the individuals expected values, which is the fractional phase.

Remainder stochastic sampling with replacement (RSSR) employs roulette wheel selection to sample the individual not assigned deterministically. During the roulette wheel selection phase, individual's fractional parts keep unchanged and, thus, compete for selection between "spins". RSSR yields zero bias and the spread is lower bounded. The upper bound is determined only by the number of fractionally assigned samples and the size of the integral part of an individual. For example, any individual with a positive fractional part could win all the samples in the fractional phase. Remainder stochastic sampling without replacement (RSSWR) forces the fractional part of an individual's expected values to zero if it is sampled during the fractional phase. As a result, this gives RSSWR minimum spread, although this selection method is biased in favour of smaller fractions.

5.6.2 Stochastic Universal Sampling

Stochastic universal sampling (SUS) is a single-phase sampling algorithm that has minimum spread and zero bias. It uses N equally spaced pointers rather than the single selection pointer employed in roulette wheel methods, where N is the number of selections required. The population is shuffled randomly and a single random number S_{rn} in the range $[0, S_{um}/N]$ is generated. The N individuals are then chosen by generating the N pointers spaced by 1, $[S_{rn}, S_{rn}+1, ..., S_{rn}+N-1]$, and selecting the individuals whose fitnesses span the positions of the pointers. An individual is thus guaranteed to be selected for a minimum of $\underline{e}_{nt}(i)$ times and no more than $\bar{e}_{nt}(i)$. So, its minimum spread is achieved. In addition, since the individuals are selected entirely on their position in the population, SUS has zero bias.

5.7 Crossover

Crossover is the basic operator for generating new chromosomes in GAs. Like its counterpart in nature, crossover produces new individuals with some parts of both parent's genetic material. In this section, several variations on crossover are described and discussed and the relative merits of each will be reviewed.

5.7.1 Single-Point Crossover

Single-point crossover is the simplest form of crossover. The single-point crossover creates two children by swapping parts of two parent chromosomes at a randomly selected point. Two examples of the application of single-point crossover during a run of a genetic algorithm are shown below.

Parent1 :	0	1	1	1	0	|	1	1	0
Parent2 :	1	1	0	1	1	|	1	0	1
			⇓						
Child1 :	0	1	1	1	0	|	1	0	1
Child2 :	1	1	0	1	1	|	1	1	0

Parent1 :	0	1	1	1	|	0	1	1	0
Parent2 :	1	1	0	1	|	1	1	1	1
			⇓						
Child1 :	0	1	1	1	|	1	1	1	1
Child2 :	1	1	0	1	|	0	1	1	0

where the children are made by cutting each parent into two parts at the point denoted by the vertical line and exchanging the second parts of each parent.

One important feature of single-point crossover is that it can produce children that are radically different from their parents. The first example above is an instance of this. Another important feature is that single-point crossover will not introduce differences for a bit in a position where both parents have the same value. Thus, in the second example, both parents and both children in bit positions 2 and 3 have the same value, even though crossover has occurred. In an extreme case where both parents are identical, the single-point crossover can introduce no diversity in the children.

5.7.2 Multi-Point Crossover

For multi-point crossover, m crossover positions are chosen at random with no duplicates and sorted into ascending order. The bits between successive

crossover points are then exchanged between the two parents to produce two new offspring. But, the section between the first allele position and the first crossover point is not exchanged between individuals. This process is illustrated as follows:

Parent1 :	0	1	|	1	1	0	|	1	1	0
Parent2 :	1	1	|	0	1	1	|	1	0	1
			⇓							
Child1 :	1	1	|	1	1	0	|	1	0	1
Child2 :	0	1	|	0	1	1	|	1	1	0

There are many variations on the crossover operator. But, the basic idea behind multi-point is that the parts of the chromosome representation contributing the most to the performance of a particular individual may not necessarily be contained in adjacent substrings (Booker, 1987). Further, the disruptive nature of multi-point crossover appears to encourage the exploration of the search space, instead of favouring the convergence to highly fit individuals early in the search. Thus, this makes the search more robust (Spears and De Jong, 1991a).

5.7.3 Uniform Crossover

It has been shown that both single- and multi-point crossovers define cross points as places between loci where a chromosome can be split. Uniform crossover (Syswerda, 1989) generalises this scheme so that every locus can be a potential crossover point. A crossover mask with the same length as the chromosome structures is created at random. The parity of the bits in the crossover mask indicates which parent will supply the offspring with which bits. Consider the following two parents and crossover mask:

Parent1	=	0	1	0	1	0	1	1	0
Parent2	=	1	1	1	0	1	0	0	1
Mask	=	1	1	0	0	1	0	0	1

which results in the offspring below:

Child1	=	0	1	1	0	0	0	0	0
Child2	=	1	1	0	1	1	1	1	1

Here, the first offspring Child1 is created by taking the bit from the parent Parent1 if the corresponding mask bit is 1 or the bit from the parent Parent2 if the corresponding mask bit is 0. The second offspring Child2 is created using the inverse of the mask or, equivalently, swapping Parent1 and Parent2.

Like multi-point crossover, uniform crossover has been claimed to reduce the bias associated with the length of the binary representation and the particular coding for a given parameter set. This makes it possible to overcome the bias in single-point crossover towards short substrings without requiring precise understanding of the significance of individual bits in the chromosome representation. The research work done by Spears and De Jong (1991b) has demonstrated how uniform crossover may be parameterised by applying a probability to the swapping of bits. This extra parameter helps to control the amount of disruption during recombination without introducing a bias towards the length of the representation used. When uniform crossover is employed with real-valued alleles, it is usually referred to as discrete recombination.

5.7.4 Other Crossover Operators

A related crossover operator is that of shuffle (Caruana *et al.*, 1989). This operator works in this way. When a single cross-point is selected, the bits are randomly shuffled in both parents before they are exchanged. After recombination, the bits in the offspring are unshuffled. This also removes positional bias as the bits are randomly reassigned each time crossover is performed.

The reduced surrogate operator (Booker, 1987) constrains crossover so that it always produce new individuals wherever possible. This is often implemented by restricting the location of crossover points such that crossover points only occur where gene values differ.

For real-valued encodings, it is possible to use a crossover operator that incorporates a numerical flavour. For example, an average crossover is introduced. This crossover operator takes two parents and produces one child that is the result of averaging the corresponding fields of two parents (Davis, 1989).

5.7.5 Intermediate Recombination

Intermediate recombination is a method of producing new phenotypes around and between the values of the parents phenotypes for a real-valued encoding of the chromosome structure (Muhlenbein and Schlierkamp-Voosen, 1993). Offspring O_1 is produced according to the rule below,

$$O_1 = P_1 \times \gamma(P_2 - P_1) \tag{5.7.1}$$

where γ is a scaling factor chosen uniformly at random over some interval, typically $[-0.25, 1.25]$ and P_1 and P_2 are the parent chromosomes (see, *e.g.* Muhlenbein and Schlierkamp-Voosen, 1993). According to the above expression, for each pair of parent genes, each variable in the offspring is the result of combining the variables in the parents. In geometric terms, intermediate recombination is able to produce new variables within a slightly larger hyper-

cube than that defined by the parents but constrained by the range of γ, as shown in Figure 5.2.

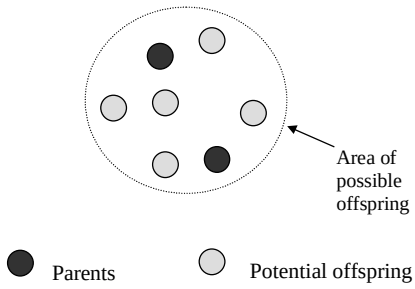

Figure 5.2: Effect of intermediate recombination

5.7.6 Line Recombination

Line recombination (Muhlenbein and Schlierkamp-Voosen, 1993) is a special case of intermediate recombination. For the line recombination, only one value of γ is used in the recombination. It is shown in Figure 5.3 how line recombination can generate any point on the line defined by the parents within the limits of the perturbation, γ, for a recombination in two variables.

To some extent, the binary operators discussed in this section have used a disruption in the representation to help improve exploration during recombination. Whilst these operators may be applied to real-valued populations, the resulting changes in the genetic material after recombination would not extend to the actual values of the decision variables, although offspring may, of course, contain genes from either parent. This limitation by acting on the decision variables themselves is overcome by the intermediate and line recombination operators. Similar to uniform crossover, the real-valued operators may also be parameterised to provide a control over the level of disruption introduced into the offspring. For discrete-valued representations, variations on the recombination operators may be employed to ensure that only valid values are produced as a result of crossover (Furuya and Haftka, 1993).

5.8 Mutation

Mutation is a random process in natural evolution where one allele of a gene is replaced by another to produce a new genetic structure. In GAs, mutation is randomly applied to modify some elements in the chromosomes with low probability, particularly in the range 0.001 and 0.01. Usually it is considered as a background operator. The role of mutation is often regarded as providing a guarantee that the probability of searching any given string will never be

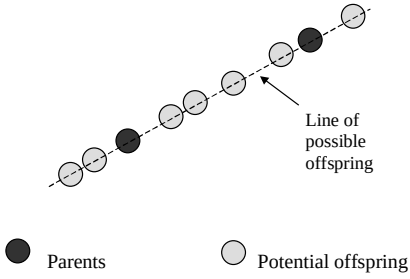

Figure 5.3: Effect of line recombination

zero and acting as a safety net to recover good genetic material that may be lost through the action of selection and crossover (Goldberg, 1989).

The effect of mutation on a binary string is illustrated as the following:

Original	string :	0	1	1	1	**0**	1	1	0
Mutated	string :	0	1	1	1	**1**	1	1	0

where a 8-bit chromosome represents a real value decoded over the interval $[0, 8]$ using both standard and Gray coding. A mutation point is the 4th position in the binary string from the right. Clearly, binary mutation flips the value of the bit at the loci selected to be the mutation point. Generally, given that mutation is generally applied uniformly to an entire population of strings, it is possible that a given binary string may be mutated at more than one point.

For non-binary representations, mutation may be achieved by either perturbing the gene values or random selection of new values within the allowed range. It is demonstrated by Wright (1991) and Janikow and Michalewicz (1991) how real-coded GAs may take advantage of higher mutation rates than binary-coded GAs, increasing the level of possible exploration of the search space without adversely affecting the convergence characteristics. In addition, Tate and Smith (1993) argued that for more complex codings than binary ones, high mutation rates can be both desirable and necessary. They also showed how, for a complex combinatorial optimisation problem, high mutation rates and non-binary coding yielded significantly better solutions than the normal approach.

A number of variations on the mutation operator have been presented. For example, biasing the mutation towards individuals with lower fitness values is used to increase the exploration in the search without losing information from the fitter individuals (Davis, 1989). Parameterising the mutation can result in that the mutation rate decreases with the population convergence (Fogarty, 1989a). A mutation operator for the real-coded GA introduced by Muhlenbein and Schlierkamp-Voosen (1993) uses a nonlinear term for the

distribution of the range of mutation applied to gene values. It is claimed that mutation can be used in conjunction with recombination as a foreground search process by biasing mutation towards smaller changes in gene values. Trade mutation (Lucasius and Kateman, 1992) states that the contribution of individual genes in a chromosome is used to direct mutation towards weaker terms. Reorder mutation (Lucasius and Kateman, 1992) swaps the positions of bits or genes to increase diversity in the decision variable space.

5.9 Reinsertion and Termination

5.9.1 Reinsertion

After the selection and recombination operations on individuals in the old population, a new population is produced and the fitness of its individuals may also be determined. In the case where fewer individuals are produced by recombination than the size of the original population, the fractional difference between the new and old population sizes is then termed as a generation gap (De Jong and Sarma, 1993). If the number of new individuals produced at each generation is one or two, the GA is said to be steady-state (Whitley, 1989) or incremental (Huang and Fogarty, 1991). Further, the GA is said to use an elitist strategy if the fittest individual is deterministically allowed to propagate through successive generations.

To keep the size of the original population, the new individuals may need to be re-inserted into the new population. Similarly, if not all the new individuals are used at each generation or if more offspring are generated than the size of the old population, then a re-insertion scheme must be applied to determine which individuals are to exist in the new population. An important feature of not creating more offspring than the current population size at each generation is that its computational time is reduced. The time reduction will be dramatic in the case of the steady-state GA and the computer memory requirements are smaller as fewer new individuals need to be stored while offspring are produced.

To select which members of the old population should be replaced, the most apparent strategy is to replace the least fit members deterministically. On the other hand, in studies, Fogarty (1989b) has shown that whether the individuals selected for replacement were chosen with inverse proportional selection or deterministically as the least fit, no significant difference in convergence characteristics was found. He further asserts that replacing the least fit members effectively implements an elitist strategy because the fittest one will probabilistically survive through successive generations. Indeed, one of the most successful replacement schemes is to select the oldest members of a population for replacement. Therefore, for an individual to survive successively, it must be sufficiently fit to ensure propagation into future generations.

5.9.2 Termination

Since the GA is a stochastic search method, it is hard to formally specify convergence criteria. In many cases, the fitness of a population may remain static for a number of generations before a superior individual is found. Then, the application of conventional termination criteria becomes problematic. A common practice is to terminate the GA after a prespecified number of generations are carried out and then test the quality of the best members of the population against the problem definition. If the solutions are not acceptable, the GA may be restarted or a fresh search initiated.

5.10 Multiobjective Optimisation with GAs

The multiple performance measures of multiobjective problems must be converted into a scalar fitness measure before GAs can be applied. In problems where there is no global criterion that directly emerges from the original multiobjective formulation, objectives are often artificially combined by means of an aggregating function. A number of such approaches can also be used with GAs although they are initially developed with other optimisers. Optimising a combination of the objectives has the advantage of producing a single compromise solution because it does not require further interaction with the decision maker. However, if the solution cannot be accepted as a good compromise, the aggregating function may be required to be tuned. New runs of the optimiser will follow until a suitable solution is found. As the many candidate solutions are evaluated in a single run of the GA, those non-dominated solutions may provide valuable alternatives (Wilson and Macleod, 1993; Fonseca *et al.*, 1993). However, such alternatives cannot be expected to be optimal in any sense since the algorithm sees them as sub-optimal. Aggregating functions have been widely used with GAs in multiobjective optimisation, from the simple weighted sum approach (*e.g.*, Jakob *et al.*, 1992) to target vector optimisation (Wienke *et al.*, 1992). Among other methods, an implementation of goal attainment has been used by Wilson and Macleod (1993).

5.10.1 Constrained Optimisation

The simplest approach to handling constraints in GAs has been to assign an arbitrarily low fitness to inferior individuals (Goldberg, 1989). This enables GAs to cope with discontinuities, arising on the constraint boundaries. In this approach, if feasible solutions can be easily found, any inferior individuals are selected out and the search is not affected much.

Certain types of constraints, such as bounds on the decision variables and other linear constraints, can be handled by mapping the search space so as to minimise the number of unfeasible solutions it contains. They can also be handled by designing the mutation and recombination operators carefully in

order to minimise the production of inferior offspring from feasible parents (Michalewicz and Janikow, 1991). The above approaches are complementary and often used in combination with each other.

In the case where feasible individuals are unknown and cannot easily be found, it will make the initial stages of evolution degenerate into a random walk by simply assigning low-fitness to inferior individuals. To overcome this problem, the penalty imposed onto inferior individuals can be made to depend on the extent to which they violate the constraints. Such penalty is typically added to the (unconstrained) performance value before fitness is computed (Goldberg, 1989). Although penalty functions give a way of guiding the search towards feasible solutions when these are not known, they are very much problem dependent. In spite of the penalty, some unfeasible solutions can be seen as better than some other feasible ones, which can make the population evolve towards a false optimum. In response to these difficulties, guidelines on the use of penalty functions have been detailed in Richardson *et al.* (1989).

One of the most recent approaches to constraint handling has been proposed by Powell and Skolnick (1993). This approach consists of rescaling the original objective function to assume that its values are less than unity in the feasible region, whilst assigning inferior individuals penalty values greater than one. Subsequent ranking of the population correctly assigns higher fitness to all feasible points than to those unfeasible ones.

5.10.2 Non-Pareto Optimisation

An approach treating objectives separately, known as the Vector Evaluated Genetic Algorithm (VEGA), is proposed by Schaffer (1985), as a move towards finding multiple non-dominated solutions with a single algorithm run. In this approach, appropriate fractions of the next generation or sub-populations are selected in terms of each of the objectives, separately. As usual, crossover and mutation are applied after shuffling all the sub-populations together. Non-dominated individuals are identified by monitoring the population as it evolves. In an application oriented paper, Fourman (1985) also chooses not to combine the different objectives. Selection is performed by comparing pairs of individuals and each pair is compared according to one objective selected at random. Fourman first experimented with assigning different priorities to the objectives and comparing individuals lexically. But it was found that selecting objectives randomly work surprisingly well.

However, shuffling sub-populations together, or having different objectives affecting different tournaments, corresponds to averaging the fitness components associated with each of the objectives. Proportional fitness assignment used in Schaffer(1985) expects that fitness corresponds to a linear combination of the objectives with variable weights, as noted in Richardson *et al.* (1989). On the other hand, the approach in Fourman (1985) corresponds

to an averaging of rank, not objective, values. In both cases, different non-dominated individuals are generally assigned different fitness values. But the performance of the algorithms on problems with concave trade-off surfaces can be qualitatively different. Another approach to selection based on the use of single objectives in alternation has been proposed in Kursawe (1991). Hajela and Lin (1992) elaborate on the VEGA by explicitly including sets of weights in the chromosome.

5.10.3 Pareto-Based Optimisation

Another class of GA optimisation approaches, based on ranking according to the actual concept of Pareto optimality, was proposed later by Goldberg (1989, pp. 201), which guarantees equal probability of reproduction to all non-dominated individuals. Problems with non-convex trade-off surfaces, presenting difficulties to pure weighted sum approaches, do not raise any special issues in Pareto optimisation.

Pareto-based ranking combines dominance with preference information to produce a suitable fitness assignment strategy. The GA optimisation process is seen as the result of the interaction between an artificial selector, here referred to as the decision maker (DM), and a GA search process. The search process creates a new set of candidate solutions in the term of the utility assigned by the DM to the current set of candidates.

The DM can represent any utility assignment strategy, which may range from an intelligent decision maker to a simple weighted sum approach. When the action of the DM influences the production of new individuals, these, as they are evaluated, provide new trade-off information which the DM can use to refine its current preferences. The GA sees the effect of any changes in the decision process, which may or may not result from taking recently acquired information into account, as an environmental change.

The GA is concerned with a different search process, but complementary aspect of the optimisation. In the first instance, genetic algorithms make very few assumptions about the fitness landscape they work on, which justifies and permits a primary concern with fitness assignment. However, GAs are not capable of optimising arbitrary functions (Hart and Belew, 1991). Therefore, some forms of characterisation of the multiobjective fitness landscapes associated with the decision making strategy is important and the design of the GA should take that information into account.

5.11 An Example

To illustrate genetic algorithms, let us use them to find out the minimum of the following function:

$$f(x_1, x_2) = x_1^2 + 2x_2^2 \tag{5.11.1}$$

Each solution in the population is represented as a real number string. Then, the chromosomal representation is expressed as

$$S = (x_1, x_2) \tag{5.11.2}$$

Five random initial solution sets are chosen as follows:

Chromosome S_i	(x_1, x_2)
S_1	(1, 2)
S_2	(-3, 4)
S_3	(2, 3)
S_4	(4, -2)
S_5	(3, 1)

To calculate the fitness values, plug each solution set into the expression $x_1^2 + 2x_2^2$. The fitness values of the above five solution sets are the following:

Chromosome S_i	$x_1^2 + 2x_2^2$
S_1	9
S_2	41
S_3	22
S_4	24
S_5	11

Since the fitness values that are lower are closer to the minimum, these values are more desirable. In this case, higher fitness values are not desirable, while lower ones are. In order to create a system where chromosomes with more desirable fitness values are more likely to be chosen as parents, the percentages that each chromosome has of being picked must first be calculated. One solution is to take the sum of the multiplicative inverses of the fitness values

$$1/9 + 1/41 + 1/22 + 1/24 + 1/11 = 0.3135 \tag{5.11.3}$$

and then calculate the percentages from there.

Chromosome S_i	Likelihood
S_1	1/9/0.3135=35.43%
S_2	1/41/0.3135=7.78%
S_3	1/22/0.3135=14.50%
S_4	1/24/0.3135=13.29%
S_5	1/11/0.3135=29.00%

In order to select five pairs of parents (each of which will have one child, and thus five new solutions sets total), we assume that there is a 10000 sided die, and on 3543 of those sides, chromosome S_1 is labeled, and on 778 of those sides, chromosome S_2 is labeled, and on 1450 of those sides, chromosome S_3 is labeled, and on 1329 of those sides, chromosome S_4 is labeled, and on 2900 of those sides, chromosome S_5 is labeled. To choose our first pair the die is rolled twice, and those chromosomes are taken to be the first two parents. Continuing in this fashion gives the following parents:

Parent 1	Parent 2
S_1	S_5
S_5	S_4
S_4	S_1
S_1	S_3
S_3	S_5

The child of each of these parents contains the genetic information of both parents 1 and 2. How this can be determined is very arbitrary. For this case, an average crossover function is employed to produce the child, which is the result of averaging the corresponding fields of two parents. Thus, five new solution sets are below.

Child Chromosome S_i	(x_1, x_2)
S_1	(2, 1.5)
S_2	(3.5, -0.5)
S_3	(2.5, 0)
S_4	(1.5, 2.5)
S_5	(2.5, 2)

Now the fitness values for the new generation of offspring are calculated.

Chromosome S_i	$x_1^2 + 2x_2^2$
S_1	8.50
S_2	12.75
S_3	6.25
S_4	14.75
S_5	14.25

The average fitness value for the child chromosomes is 11.3, while the average fitness value for the parent chromosomes is 21.4. The next generation (the offspring) are supposed to mutate, that is, for example, some values of each chromosome are disturbed by a small random value, *e.g.,* a value between

0.1 and -0.1. Progressing at this rate, one chromosome should eventually reach a fitness level of 0 eventually, that is when a solution is found.

5.12 Summary

The principal points addressed in this chapter were the following: Genetic algorithms were invented to mimic some features of natural evolution for optimisation. Evaluation functions link genetic algorithms with the problems to be solved. Genetic algorithms use mutation and crossover to create new generations that differs from old generations. Genetic algorithms use selection techniques, which simulate the process of natural selection, so that the fittest individuals tend to reproduce most often. Genetic algorithms manipulate schemata, which builds blocks of good solutions that are combined through crossover and that spread in the population in proportion to the relative fitness of the chromosomes that contain them. The inversion operator is inspired by natural processes that have not yet been widely used.

Chapter 6

Robust Control System Design by Mixed Optimisation

6.1 Introduction

Traditionally, control system designers use design methods based on either analytical optimisation or parameter optimisation. Both methods have advantages and disadvantages; briefly, analytical optimisation techniques (*e.g.* H_∞, LQG) generally:

a) have non-explicit closed-loop performance,
b) are single-objective,
c) are robustly stable,
d) provide high-order controllers,
e) are not very flexible,
f) provide a global optimum,
g) can deal with relatively large multivariable problems;

whereas parameter optimisation based methods (*e.g.* MoI) generally:

a) have explicit closed-loop performance,
b) are often multiobjective,
c) are not implicitly robustly stable,
d) provide simple controllers,
e) are flexible,
f) are often non-convex resulting in local minima,
g) can deal with small problems only,
h) may have difficulty stabilising the system.

A combination of analytical optimisation and parameter search methods may be able to overcome some of the limitations of using just one approach.

The MoI can be combined with analytical optimisation techniques by using the parameters of the weighting functions generally required by such techniques as the design parameters. Thus, for a nominal plant $G(s)$ augmented by a set of n_w weighting functions $W(s) = (W_1(s), W_2(s) \dots W_{n_w})$, a controller $K_{\min}(s, G, W)$, which is optimal in some sense, can generally be synthesised, and a set of closed-loop performance functions ϕ of the optimal control system can be calculated. If the weighting functions are simply parameterised by the design vector p, the problem can be formulated as for (1.1.11), and the MoI used to design the weights for the analytical optimisation problem. The designer thus chooses the optimisation technique and the structure of the weighting functions $W(s, p)$. He/she defines suitable performance functions (*e.g.* rise time, settling time, bandwidth, system norms, etc.) along with the design goals ε_i, and the MoI can be used to search for suitable values of the weighting function parameters p such that (1.1.11) is satisfied.

Details of the above procedure applied to McFarlane and Glover's LSDP are given in the next section. Further details may be found in Whidborne *et al.* (1994), Whidborne *et al.* (1995a,b) and Chipperfield *et al.* (1999). Details of the mixed-optimisation procedure applied to a 2 degree-of-freedom LSDP (Hoyle *et al.*, 1991; Limebeer *et al.*, 1993) may be found in Murad *et al.* (1993) and Whidborne *et al.* (1995a) and to a mixed sensitivity H_∞ approach and an LQG optimisation in Whidborne *et al.* (1995b). An alternative 2 degree-of-freedom LSDP approach with a parameterised pre-compensator is proposed in Whidborne *et al.* (1994b).

Other authors have suggested using a mixed optimisation approach. The first appears to be Baras *et al.* (1984), who suggested using nonlinear optimisation for designing the weights for LQG control design. A similar approach has been independently suggested by Paddison *et al.* (1994), Haessig (1995) and Tych (1994). Use of genetic algorithms to design the weighting functions for the LSDP has also been suggested by White *et al.* (1995). Some efficient methods for mixed optimisation with the LSDP have been proposed by Pantas (1998).

6.2 An H_∞ Loop Shaping Design Procedure

6.2.1 Overview

In general, we are not able to get an absolutely accurate model of the system plant. There are assumptions made about the model, simplifications, and few (if any) real systems are linear. In addition, during the operation of a control system, the plant dynamics change over time, parts wear out, ambient temperature changes, and the operating conditions, such as the load, change. We therefore need to be able to design control systems which can handle this

uncertainty. A control system which can maintain stability and performance in the face of uncertainty is described as *robust*.

The loop-shaping design procedure (LSDP) described in this section is a robust control system design method. It is based on H_∞ robust stabilisation combined with classical loop-shaping. The method was developed by McFarlane and Glover (1992) and is hence sometimes known as McFarlane and Glover's LSDP. Further details of the approach can be found in McFarlane and Glover (1990), Zhou *et al.* (1996), Skogestad and Postlethwaite (1996), Green and Limebeer (1995) and Hyde (1995).

The procedure is essentially a two stage design process. First, the open-loop plant is augmented by pre- and post-plant weighting functions to give a desired shape to the singular values of the open-loop frequency response. Then the resulting shaped plant is robustly stabilised using H_∞ optimisation of a normalised left coprime factorisation description of the plant.

This LSDP is available with the MATLAB Toolbox μ-tools (Balas *et al.*, 1991). A graphical interface for the LSDP, which can also be used with the Robust Control Toolbox (Chiang and Safonov, 1992), is available on the World Wide Web at: `http://www.eee.kcl.ac.uk/mecheng/jfw/lsdptool.html`

6.2.2 Preliminaries

The H_∞-Norm

For a system $G(s)$, the H_∞-norm is defined as

$$\| G \|_\infty = \sup_\omega \{ \bar{\sigma} (G(j\omega)) \} \tag{6.2.1}$$

where $\bar{\sigma}(\cdot)$ denotes the maximum singular value. The H_∞-norm of a system gives the worst case steady-state gain for sinusoidal inputs of any frequency. It is also equal to the worst case energy gain of the system. For further details, see Skogestad and Postlethwaite (1996, pp. 153).

The Small Gain Theorem

The robust stability of a system can be analysed by means of the well-known small gain theorem.

Theorem 6.1 *The system shown in Figure 6.1 is stable if and only if*
a) $\| M \|_\infty < \gamma$ *and for all* Δ *such that* $\| \Delta \|_\infty \leq \frac{1}{\gamma}$,
b) $\| M \|_\infty \leq \gamma$ *and for all* Δ *such that* $\| \Delta \|_\infty < \frac{1}{\gamma}$.

This theorem applies for multi-input multi-output systems. Thus H_∞ theory is used for robust multivariable control system design. From the small

gain theorem, stability conditions for a variety of models of system uncertainty can be obtained. For further details, see, for example, Skogestad and Postlethwaite (1996, pp. 291-309).

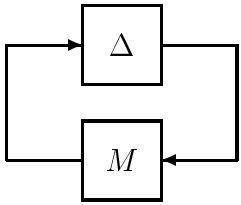

Figure 6.1: General system for the small gain theorem

6.2.3 Normalised Left Coprime Factorisation

A plant model G can be factorised into two stable transfer function matrices (M, N) so that

$$G = M^{-1}N \tag{6.2.2}$$

Such a factorisation is called a left coprime factorisation of G if there exists stable transfer function matrices (V, U) such that

$$MV + NU = I \tag{6.2.3}$$

A left coprime factorisation of a plant model G is normalised if and only if

$$NN^{\sim} + MM^{\sim} = I, \quad \forall\, s \tag{6.2.4}$$

where $N^{\sim}(s) = N^T(-s)$ and $M^{\sim}(s) = M^T(-s)$.

Let

$$G(s) = D + C(sI - A)^{-1}B \stackrel{s}{=} \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \tag{6.2.5}$$

then

$$[N \;\; M] \stackrel{s}{=} \left[\begin{array}{c|cc} A + HC & B + HD & H \\ \hline R^{-1/2}C & R^{-1/2}D & R^{-1/2} \end{array}\right] \tag{6.2.6}$$

is a normalised coprime factorisation of G where $H = -(BD^T + ZC^T)R^{-1}$, $R = I + DD^T$, and the matrix $Z \geq 0$ is the unique stabilising solution to the algebraic Riccati equation (ARE)

$$(A - BS^{-1}D^TC)Z + Z(A - BS^{-1}D^TC)^T - ZC^TR^{-1}CZ + BS^{-1}B^T = 0 \tag{6.2.7}$$

where $S = I + D^TD$.

6.2.4 Coprime Factor Robust H_∞ Stability Problem

The plant uncertainty is represented by stable additive perturbations on each of the factors in a coprime factorisation of the plant. This uncertainty description is difficult to conceptualise and is not as intuitive as, for example, plant multiplicative or plant additive uncertainty. However, it is quite general, and it leads to a very useful H_∞ robust stabilisation problem (Skogestad and Postlethwaite, 1996, pp. 376-377).

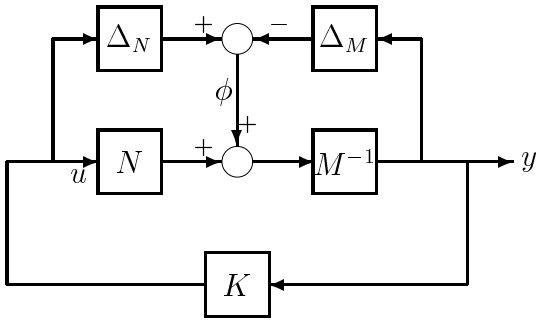

Figure 6.2: Coprime factor robust H_∞ stability problem

The perturbed plant model G_p is

$$G_p = (M + \Delta_M)^{-1}(N + \Delta_N) \tag{6.2.8}$$

where Δ_M, Δ_N are stable unknown transfer functions which represent the uncertainty in the nominal plant model G. The objective of robust stabilisation it to stabilise not only the nominal model G, but also the set of plants defined by

$$\mathcal{G} = \left\{(M + \Delta_M)^{-1}(N + \Delta_N) : \| \begin{bmatrix} \Delta_N & \Delta_M \end{bmatrix} \|_\infty < \epsilon_\gamma \right\} \tag{6.2.9}$$

where $\epsilon_\gamma > 0$ is then the stability margin. To maximise this stability margin is the problem of robust stabilisation of normalised coprime factor plant descriptions (Glover and McFarlane, 1989).

From the small gain theorem (Theorem 6.1), the perturbed feedback system shown in Figure 6.2 is robustly stable for the family of plants $\mathcal{G}$ defined by (6.2.9), if and only if

$$\gamma := \left\| \begin{bmatrix} K \\ I \end{bmatrix} (I - GK)^{-1} M^{-1} \right\|_\infty \leq \frac{1}{\epsilon_\gamma} \tag{6.2.10}$$

Notice that γ is the H_∞-norm from ϕ to $\begin{pmatrix} u \\ y \end{pmatrix}$ and $(I - GK)^{-1}$ is the sensitivity function for this positive feedback arrangement.

If $X \geq 0$ is the unique solution of the algebraic Riccati equation

$$(A - BS^{-1}D^TC)^TX + X(A - BS^{-1}D^TC) - XBS^{-1}B^TX + C^TR^{-1}C = 0 \tag{6.2.11}$$

and Z is the solution to (6.2.7), then the lowest achievable value of γ is given by

$$\gamma_0 = (1 + \rho(XZ))^{1/2} \tag{6.2.12}$$

where ρ denotes the spectral radius (maximum eigenvalue magnitude). Note that for a strictly proper plant, i.e. when $D = 0$, the formulae simplify considerably.

A controller which guarantees that

$$\left\| \begin{bmatrix} K \\ I \end{bmatrix} (I - GK)^{-1}M^{-1} \right\|_\infty \leq \gamma \tag{6.2.13}$$

for a specified $\gamma > \gamma_0$, is given by

$$K \stackrel{s}{=} \left[\begin{array}{c|c} A + BF + \gamma^2(L^T)^{-1}ZC^T(C + DF) & \gamma^2(L^T)^{-1}ZC^T \\ \hline B^TX & -D^T \end{array}\right] \tag{6.2.14}$$

where $F = -S^{-1}(D^TC + B^TX)$, and $L = (1 - \gamma^2)I + XZ$. However, if $\gamma = \gamma_0$, then $L = XZ - \lambda_{\max}(XZ)I$ which is singular, and thus (6.2.14) cannot be implemented. This problem can be resolved using the descriptor system (Safonov *et al.*, 1987).

A descriptor system $\hat{G}(s)$ has equations

$$\hat{E}\dot{x} = \hat{A}x + \hat{B}u \tag{6.2.15}$$

$$y = \hat{C}x + \hat{D}u \tag{6.2.16}$$

which can be written as

$$\hat{G}(s) = \hat{D} + \hat{C}(s\hat{E} - \hat{A})^{-1}\hat{B} \stackrel{s}{=} \left[\begin{array}{c|c} -\hat{E}s + \hat{A} & \hat{B} \\ \hline \hat{C} & \hat{D} \end{array}\right] \tag{6.2.17}$$

The descriptor system allows $\hat{E}$ to be singular, and can be converted to the usual state-space system via the singular value decomposition of $\hat{E}$ (see Chiang and Safonov (1992)). A controller which achieves a $\gamma \geq \gamma_0$ is given in descriptor form by

$$K \stackrel{s}{=} \left[\begin{array}{c|c} -L^Ts + L^T(A + BF) + \gamma^2ZC^T(C + DF) & \gamma^2ZC^T \\ \hline B^TX & -D^T \end{array}\right] \tag{6.2.18}$$

Note that because γ_0 can be computed from (6.2.12), an explicit solution to the problem can be obtained by solving just two Riccati equations and so avoiding the γ-iteration needed to solve general H_∞ problems.

6.2.5 A Loop-Shaping Design Procedure (LSDP)

For practical control system design, performance is required as well as robust stability. To do this, McFarlane and Glover (1992) have suggested weighting the open-loop nominal plant to shape the open-loop singular values prior to robust stabilisation of the "shaped" plant. By means of appropriate choice of weighting function, robust performance and stability trade-off can be made.

If W_1 and W_2 are the pre- and post-plant weighting functions respectively, then the shaped plant G_s is given by

$$G_s = W_2 G W_1 \tag{6.2.19}$$

as shown in Figure 6.3.

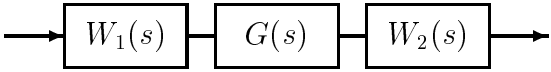

Figure 6.3: Shaped plant

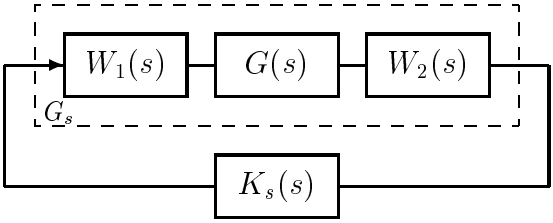

Figure 6.4: Optimal controller

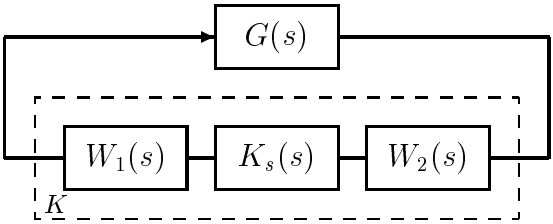

Figure 6.5: Final controller

The controller K_s is synthesised by means of (6.2.14) for the shaped plant G_s with a normalised left coprime factorisation $G_s = M_s^{-1} N_s$ as shown on

Figure 6.4. The feedback controller for the plant G is then

$$K = W_1 K_s W_2 \tag{6.2.20}$$

as shown in Figure 6.5. Skill is required in the selection of the weights W_1 and W_2, but experience on real applications has shown that robust controllers can be fairly easily designed by following some simple rules (Skogestad and Postlethwaite, 1996, pp. 382).

The following systematic procedure for using H_∞ loop-shaping design is drawn from that recommended by Skogestad and Postlethwaite (1996, pp. 382-385):

a) Scale the plant outputs and inputs. This is very important for most design procedures and is sometimes forgotten. In general, scaling improves the conditioning of the design problem, it enables meaningful analysis to be made of the robustness properties of the feedback system in the frequency domain, and for loop-shaping it can simplify the selection of weights. There are a variety of methods available including normalisation with respect to the magnitude of the maximum or average value of the signal in question. However, if one is to go straight to a design the following variation has proved useful in practice:

   i) The outputs are scaled such that equal magnitudes of cross-coupling into each of the outputs is equally undesirable.

   ii) Each input is scaled by a given percentage (say 10%) of its expected range of operation. That is, the inputs are scaled to reflect the relative actuator capabilities.

b) Make the plant as diagonal as possible by ordering the inputs and outputs. The purpose of this pseudo-diagonalisation is to ease the design of the pre- and post-plant weighting functions which, for simplicity, are generally chosen to be diagonal.

c) Select the elements of diagonal pre- and post-plant weighting functions W_1 and W_2 so that the singular values of $W_2 G W_1$ are a desirable shape as shown in Figure 6.6. This would normally mean high gain at low frequencies, roll-off rates of approximately 20 dB/decade at the desired bandwidth(s), with higher rates at high frequencies. The post-plant weighting function, W_2, is usually chosen as a constant, reflecting the relative importance of the outputs to be controlled and the other measurements being fed back to the controller. For example, if there are feedback measurements of two outputs to be controlled and a velocity signal, then W_2 might be chosen to be diag [1, 1, 0.1], where 0.1 is in the velocity signal channel. The pre-plant weighting function, W_1, contains the dynamic shaping. This should usually include integral action, for low frequency performance; phase-advance for reducing the roll-off rates at crossover; and phase-lag to increase the roll-off rates at high

frequencies should all be placed in W_1 if desired. The weights should be chosen so that no unstable hidden modes are created in G_s.

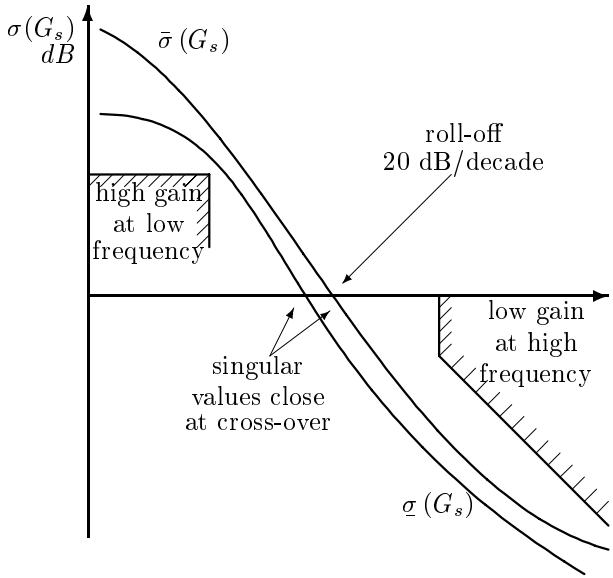

Figure 6.6: Desired shaped plant

d) Robustly stabilise the shaped plant $G_s = W_2 G W_1$, using the formulae of the previous section. First, calculate the maximum permissible uncertainty γ_0. If this is too large ($\gamma_0 > 5$) then go back to step c) and modify the weights. Otherwise synthesise an optimal or sub-optimal controller using (6.2.14) or (6.2.18). There is generally not a great advantage to be gained by using the optimal controller, so select $\gamma \geq \gamma_0$, by up to about 10%. When $\gamma_0 < 5$, the design is usually successful. In this case, at least 20% coprime uncertainty is allowed, and it is also found that the shape of the open-loop singular values do not change much after robust stabilisation. A large value of γ_0 indicates that the chosen singular value loop-shapes are incompatible with robust stability requirements.

e) Analyse the design and if all the specifications are not met make further modifications to the weights.

f) Implement the controller.
For a tracking problem, the reference signal is generally fed between K and $\hat{W}_1$ as shown in Figure 6.7, so that the closed loop transfer function between the reference r and the plant output y becomes

$$Y(s) = (I - G(s)K(s))^{-1} G(s) W_1(s) K_s(0) W_2(0) R(s) \qquad (6.2.21)$$

where the reference $R(s)$ is connected through a gain $K_s(0)W_2(0)$ where

$$K_s(0)W_2(0) = \lim_{s\to 0} K_s(s)W_2(s) \tag{6.2.22}$$

to ensure unity steady-state gain. This is because the references do not directly excite the dynamics of K_s, which can result in large amounts of overshoot (classical derivative kick). The constant pre-compensator ensures a steady-state gain of unity between R and Y, assuming integral action in W_1 or G.

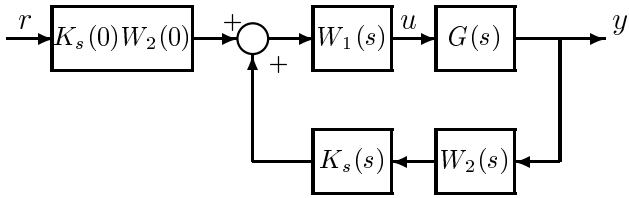

Figure 6.7: A practical implementation of the loop-shaping controller

6.2.6 Example – The Inverted Pendulum

The LSDP is applied to the traditional laboratory control problem of the inverted pendulum, as shown in Figure 6.8. The pendulum is connected to a carriage by a simple pivot. A potentiometer is connected to the pivot so that the angle of the pendulum θ can be measured. The carriage can run along a length of track. The carriage is connected via a drive belt to a motor. The position of the carriage x can be measured via a potentiometer located in one of the pulleys. The position of the top of the pendulum y_c cannot be measured directly, but can be estimated from the formula $y_c = x_c + \ell_p \sin\theta$, where ℓ_p is the pendulum length.

Mathematical Model of System

The basic circuit for a DC motor is shown in Figure 6.9. The torque is expressed by

$$T = K_t i \tag{6.2.23}$$

and the back emf is

$$e = K_e \dot{\theta}_m \tag{6.2.24}$$

The electrical circuit current/voltage relationship is

$$L_a \frac{\mathrm{d}i}{\mathrm{d}t} + R_a i = v - K_e \dot{\theta}_m \tag{6.2.25}$$

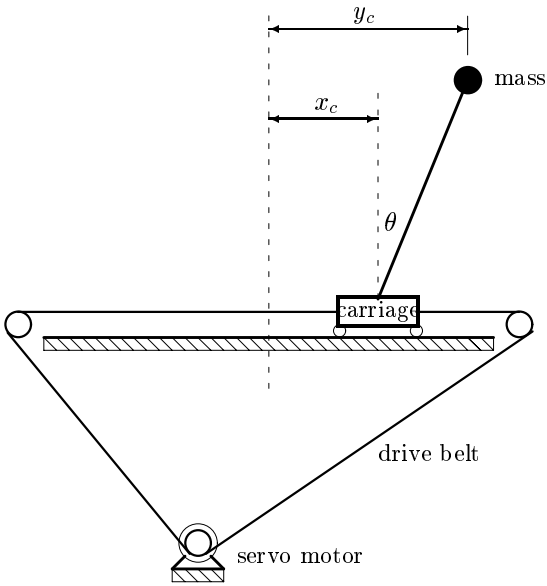

Figure 6.8: Schematic of the inverted pendulum

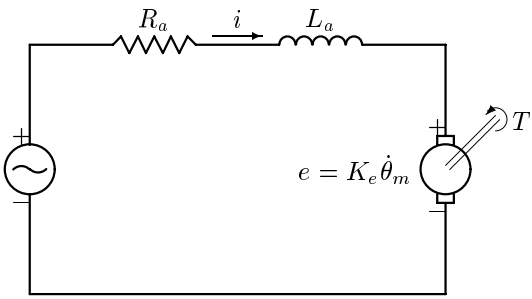

Figure 6.9: Electric circuit of DC motor

From Newton's law $F = ma$, the carriage dynamics are

$$m_t\ddot{x}_c + b\dot{x}_c = u - N \tag{6.2.26}$$

where m_t is the mass of the carriage, b is the coefficient of friction constant, u is the force applied by the belt and N is the horizontal reaction force of the

pendulum. Now assuming that the belt has no slip or elasticity, there is no friction in the pulleys and ignoring the inertia of the pulleys and rotor, then

$$u = \frac{T}{\ell_m} \tag{6.2.27}$$

where ℓ_m is the radius of the motor wheel, and

$$\dot{\theta}_m = \frac{\dot{x}_c}{\ell_m} \tag{6.2.28}$$

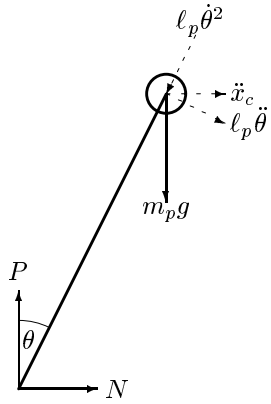

Figure 6.10: Pendulum free body

The dynamics of the pendulum arm free body, shown in Figure 6.10 can be described by resolving the horizontal forces

$$N = m_p\ddot{x}_c + m_p\ell_p\ddot{\theta}\cos\theta - m_p\ell_p\dot{\theta}^2\sin\theta \tag{6.2.29}$$

where m_p is the mass of the pendulum and ℓ_p is the length of the pendulum arm. Taking moments about the pivot point gives

$$m_pg\ell_p\sin\theta = m_p\ell_p^2\ddot{\theta} + m_p\ell_p\ddot{x}_c\cos\theta \tag{6.2.30}$$

which simplifies to

$$g\sin\theta = \ell_p\ddot{\theta} + \ddot{x}_c\cos\theta \tag{6.2.31}$$

Substituting (6.2.29) into (6.2.26) to eliminate N gives

$$u = (m_t + m_p)\ddot{x}_c + b\dot{x}_c + m_p(\ell_p\ddot{\theta}\cos\theta - \ell_p\dot{\theta}^2\sin\theta) \tag{6.2.32}$$

Substituting for $\ddot{\theta}$ from (6.2.31) gives

$$u = (m_t + m_p)\ddot{x}_c + b\dot{x}_c + m_p(g\sin\theta\cos\theta - \ddot{x}_c\cos^2\theta - \ell_p\dot{\theta}^2\sin\theta) \tag{6.2.33}$$

which simplifies to

$$u = (m_t + m_p \sin^2\theta)\ddot{x}_c + b\dot{x}_c + m_p(g\sin\theta\cos\theta - \ell_p\dot{\theta}^2\sin\theta) \tag{6.2.34}$$

Finally, (6.2.34) is rearranged with (6.2.27) and (6.2.23) to give

$$\ddot{x}_c = \frac{1}{m_t + m_p\sin^2\theta}\left(\frac{K_t}{\ell_m}i - b\dot{x}_c + m_p\sin\theta(g\cos\theta - \ell_p\dot{\theta}^2)\right) \tag{6.2.35}$$

The system is simulated in SIMULINK with the following constants :

$$\begin{aligned} m_t &= 0.2 \text{ Kg} && (6.2.36)\\ \ell_m &= 0.015 \text{ m} && (6.2.37)\\ b &= 0.1 \text{ N/m/s} && (6.2.38)\\ K_t &= 0.0984 \text{ Nm/A} && (6.2.39)\\ K_e &= K_t \text{ v/rad/s} && (6.2.40)\\ R_a &= 15\ \Omega && (6.2.41)\\ L_a &= 0.005 \text{ H} && (6.2.42)\\ \ell_p &= 0.25 \text{ m} && (6.2.43)\\ g &= 9.81\text{m/s/s} && (6.2.44) \end{aligned}$$

Controller Design

A linear state-space representation

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) && (6.2.45)\\ y(t) &= Cx(t) + Du(t) && (6.2.46) \end{aligned}$$

is required. Define state variables $x = [\,\dot{x}_c \quad x_c \quad i \quad \dot{y}_c \quad \theta\,]^T$. Then

$$A = \begin{bmatrix} -\frac{b}{m_t} & 0 & \frac{K_t}{\ell_m m_t} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ -\frac{K_e}{L_a\ell_m} & 0 & -\frac{R_a}{L_a} & 0 & 0 \\ 0 & 0 & 0 & 0 & g \\ -\frac{1}{\ell_p} & 0 & 0 & \frac{1}{\ell_p} & 0 \end{bmatrix}, \text{ and } B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L_a} \\ 0 \\ 0 \end{bmatrix} \tag{6.2.47}$$

We have measurements of the position x_c, the velocity $\dot{x}_c$ and the angle θ , so

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.2.48}$$

Using the LSDP Toolbox, the following weighting functions are designed

$$W_1(s) = 20.915\frac{(s + 0.1421)}{s} \tag{6.2.49}$$

$$W_2(s) = \text{diag}\,(1.29,\ 0.5,\ 1.561) \tag{6.2.50}$$

This results in a value of $\gamma_0 = 4.852$. The plant and shaped plant singular values are shown in Figure 6.11. The resulting optimal controller is tested in simulation using the SIMULINK model. For a reference step of 0.1 m, the resulting responses of x_c, y_c, θ and control v are shown in Figures 6.12 and 6.13 respectively.

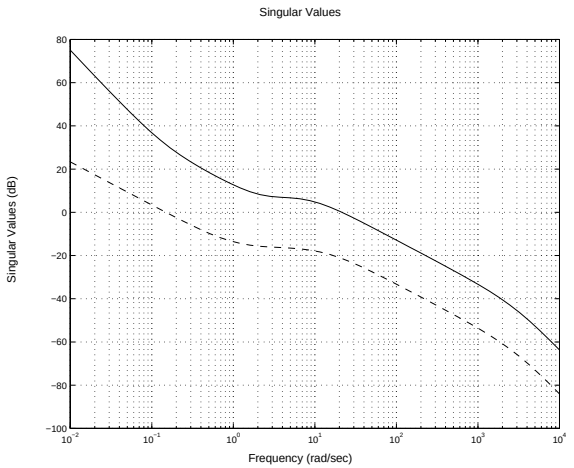

Figure 6.11: Plant singular values (- - -) shaped plant singular values (——)

6.3 Mixed-Optimisation for the LSDP

Essentially, with the LSDP, the weighting functions W_1 and W_2 are the design parameters which are chosen both to give the augmented plant a 'good' open-loop shape and to ensure that γ_0 is not too large. γ_0 is a design indicator of the success of the loop-shaping as well as a measure of the robustness of the system.

The mixed-optimisation problem can thus be expressed as:

Problem 6.1 *For the system of Figure 6.7, find* $\mathbf{p}$ *and hence* (W_1, W_2) *such that*

$$\gamma_0(W_1, W_2) \leq \epsilon_\gamma \tag{6.3.1}$$

and

$$\phi_i(W_1, W_2) \leq \varepsilon_i \qquad \text{for} \qquad i = 1 \ldots n \tag{6.3.2}$$

where

$$\gamma_0(W_1, W_2) = \inf_{K \ \textit{stabilising}} \left\| \begin{bmatrix} W_1^{-1} K \\ W_2 \end{bmatrix} (I - GK)^{-1} \begin{bmatrix} W_2^{-1} & G \end{bmatrix} \right\|_\infty \tag{6.3.3}$$

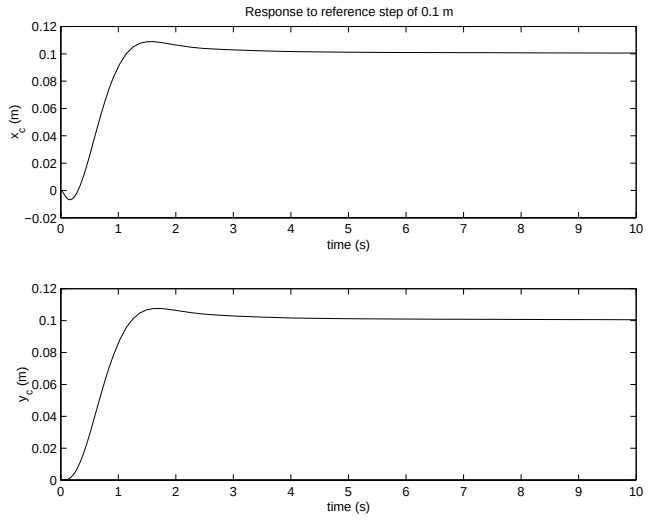

Figure 6.12: Responses of x_c and y_c to a reference step of 0.1 m

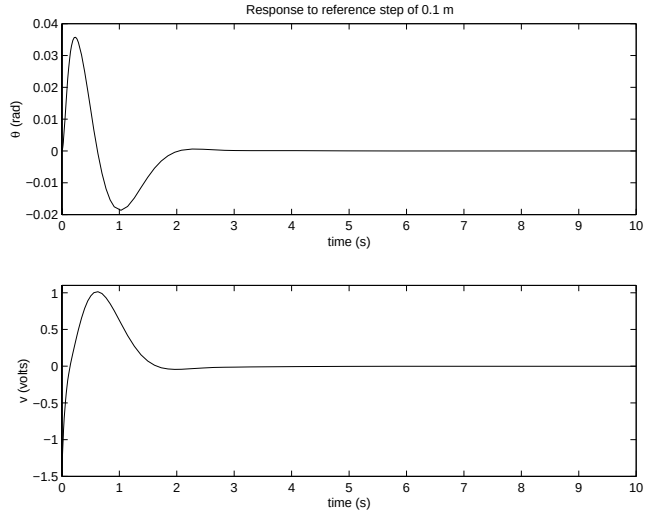

Figure 6.13: Responses of θ and control v to a reference step of 0.1 m

and $\phi_i(W_1, W_2)$ are functionals of the closed-loop system, ϵ_γ, ε_i are real numbers representing desired bounds on γ_0 and ϕ_i respectively, (W_1, W_2) is a pair of fixed order weighting functions with real parameters $\mathbf{p} = (p_1, p_2, \ldots, p_q)$.

Design Procedure

A design procedure to solve the above problem is:

S1 Define the plant G, and define the objective functions ϕ_i.

S2 Define the values of ε_γ and ε_i.

S3 Define the form and order of the weighting functions W_1 and W_2. Bounds should be placed on the values of p_i to ensure that W_1 and W_2 are stable and minimum phase to prevent undesirable pole/zero cancellations. The order of the weighting functions, and hence the value of q, should initially be small.

S4 Define initial values of p_i based on the open-loop frequency response of the plant.

S5 Implement a search algorithm in conjunction with (6.2.14) and (6.3.3) to find a W which satisfies inequalities (6.3.1) and (6.3.2), i.e. locate an admissible point. If a solution is found, the design is satisfactory. If no solution is found, change the initial values of $\mathbf{p}$ by returning to S4; change the structure of the controller by returning to S3, or relax one or more of the design goals ε_γ and ε_i by returning to S2.

S6 With satisfactory weighting functions W_1 and W_2, a satisfactory controller is obtained from (6.2.20).

6.3.1 MATLAB Implementation - The MODCONS Toolbox

A number of MATLAB routines which implement the mixed-optimisation approach have been collected together into the MODCONS Toolbox (Whidborne *et al.*, 1994a). This software is available on the World Wide Web at:

`http://www.eee.kcl.ac.uk/mecheng/jfw/modcons.html`

There are three main components at the core of the toolbox, they are:

a) the search algorithm,

b) the user-interface and

c) the objective functions calculator.

The relationship between these three components and the user is shown in Figure 6.14. The search algorithms comprise the core of the design package. These are generic to most design situations, not just for CACSD, and are the main design machine within the process. The user-interface is a very important component of the process, and this provides the interface between the search algorithms and the designer/user, and enables the designer to make intelligent decisions about the design process. The final component is the objective functions calculator. This is specific to CACSD, in fact, some functions are specific to the particular class of control system (fixed controller, mixed-optimisation, etc.).

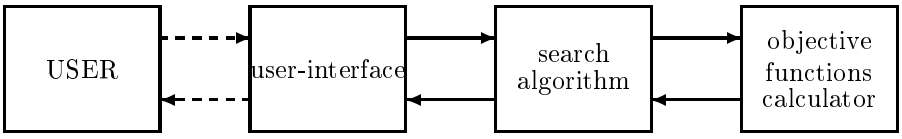

Figure 6.14: Main component relationship

6.4 Example – The Distillation Column

The mixed optimisation design method is used to design a control system for the high purity distillation column described in Limebeer (1991). The column is considered in just one of its configurations, the so-called LV configuration, for which the following model is relevant

$$G_D(s,k_1,k_2,\tau_1,\tau_2) = \frac{1}{75s+1}\begin{bmatrix} 0.878 & -0.864 \\ 1.082 & -1.096 \end{bmatrix}\begin{bmatrix} k_1 e^{-\tau_1 s} & 0 \\ 0 & k_2 e^{-\tau_2 s} \end{bmatrix} \tag{6.4.1}$$

where $0.8 \le k_1, k_2 \le 1.2$ and $0 \le \tau_1, \tau_2 \le 1$, and all time units are in minutes. The time delay and actuator gain values used in the nominal model G are $k_1 = k_2 = 1.0$ and $\tau_1 = \tau_2 = 0.5$. The time delay element is approximated by a first-order Padé approximation.

The design specifications are to design a controller which guarantees for all $0.8 \le k_1, k_2 \le 1.2$ and $0 \le \tau_1, \tau_2 \le 1$:

a) Closed-loop stability.

b) The output response to a step demand $h(t)\begin{bmatrix}1\\0\end{bmatrix}$ satisfies $y_1(t) \le 1.1$ for all t, $y_1(t) \ge 0.9$ for all $t > 30$ and $y_2(t) \le 0.5$ for all t.

c) The output response to a step demand $h(t)\begin{bmatrix}0.4\\0.6\end{bmatrix}$ satisfies $y_1(t) \le 0.5$ for all t, $y_1(t) \ge 0.35$ for all $t > 30$, $y_2(t) \le 0.7$ for all t and $y_2(t) \ge 0.55$ for all $t > 30$.

d) The output response to a step demand $h(t)\begin{bmatrix}0\\1\end{bmatrix}$ satisfies $y_1(t) \le 0.5$ for all t, $y_2(t) \le 1.1$ for all t and $y_2(t) \ge 0.9$ for all $t > 30$.

e) Zero steady-state error.

f) The frequency response of the closed-loop transfer function between demand input and plant input is gain limited to 50 dB and the unity gain cross over frequency of its largest singular value should be less than 150 rad/min.

A set of closed-loop performance functionals $\{\phi_i(G_D, W, K_p),\ i = 1, 2, \ldots, 16\}$, based on these specifications are defined. Functionals ϕ_1 to ϕ_{10} are measures of the step response specifications. Functionals ϕ_1, ϕ_4, ϕ_6 and ϕ_9 are measures of the overshoot; ϕ_2, ϕ_5, ϕ_7 and ϕ_{10} are measures of the rise-time, and ϕ_3 and ϕ_8 are measures of the cross-coupling. Denoting the output response of the closed-loop system with a plant G_D at a time t to a reference step demand

$h(t)\begin{bmatrix}h_1\\h_2\end{bmatrix}$ by $y_i([h_1\ h_2]^T,t)$, $i=1,2$, the step response functionals are

$$\begin{aligned}
\phi_1 &= \max_t y_1([1\ 0]^T,t) && (6.4.2)\\
\phi_2 &= -\min_{t>30} y_1([1\ 0]^T,t) && (6.4.3)\\
\phi_3 &= \max_t y_2([1\ 0]^T,t) && (6.4.4)\\
\phi_4 &= \max_t y_1([0.4\ 0.6]^T,t) && (6.4.5)\\
\phi_5 &= -\min_{t>30} y_1([0.4\ 0.6]^T,t) && (6.4.6)\\
\phi_6 &= \max_t y_2([0.4\ 0.6]^T,t) && (6.4.7)\\
\phi_7 &= -\min_{t>30} y_2([0.4\ 0.6]^T,t) && (6.4.8)\\
\phi_8 &= \max_t y_1([0\ 1]^T,t) && (6.4.9)\\
\phi_9 &= \max_t y_2([0\ 1]^T,t) && (6.4.10)\\
\phi_{10} &= -\min_{t>30} y_2([0\ 1]^T,t) && (6.4.11)
\end{aligned}$$

The steady-state specifications are satisfied automatically by the use of integral action.

From the gain requirement in the design specifications, ϕ_{11} is the H_∞-norm (in dB) of the closed-loop transfer function between the reference and the plant input,

$$\phi_{11} = \sup_\omega \bar{\sigma}\left((I-K(j\omega)G_D(j\omega))^{-1}W_1(j\omega)K_p(j\omega)\right) \qquad (6.4.12)$$

From the bandwidth requirement in the design specifications, ϕ_{12} is defined (in rad/min) as

$$\phi_{12} = \max\{\omega\} \text{ such that } \bar{\sigma}\left((I-K(j\omega)G_D(j\omega))^{-1}W_1(j\omega)K_p(j\omega)\right) \geq 1 \qquad (6.4.13)$$

The first design attempt was to use the MoI to satisfy the performance design specifications for the nominal plant G using the configuration of Figure 6.7. The design criteria were, from (6.3.1) and (6.3.2),

$$\begin{aligned}
\gamma_0(W) &\leq \varepsilon_\gamma && (6.4.14)\\
\phi_i(G,W,K_p) &\leq \varepsilon_i, \quad \text{for} \quad i=1,2,\ldots,12 && (6.4.15)
\end{aligned}$$

where the prescribed bound for γ_0 is not fixed, but for stability robustness, it should not be too large (McFarlane and Glover, 1990), and is here taken as

$$\varepsilon_\gamma = 5.0 \qquad (6.4.16)$$

The respective prescribed bounds are decided from the design specifications and are shown in Table 6.2 (note that ε_{15} is in dB and ε_{16} is in rad/s).

An integrator term was included in W_1, which ensures that the final controller has integral action and the steady-state specifications are satisfied. To ensure the steady-state error specifications are met, the pre-compensator is set to be the gain matrix $K_p = K_s(0)W_2(0)$ where

$$K_s(0)W_2(0) = \lim_{s\to 0} K_s(s)W_2(s) \tag{6.4.17}$$

With weighting functions $W_1 = w_1(s + w_2)/(s(s + w_3))I_2$, and $W_2 = I_2$; the design procedure described in section 6.3 was implemented in MATLAB using the MBP, and a design that successfully satisfied inequalities (6.4.14) and (6.4.15) obtained easily. The performance was then tested with various values of τ_1, τ_2, k_1 and k_2, and the design was found to be not very robust.

To obtain robust performance, the next attempt was to satisfy the performance design specifications for several plant models each at an extreme of the parameter range. The design criteria were hence amended to

$$\gamma_0(W) \leq \varepsilon_\gamma \tag{6.4.18}$$

$$\phi_i(G_j, W, K_p) \leq \varepsilon_i, \text{ for } j = 1,2,3,4, \quad i = 1,2,\ldots,12 \tag{6.4.19}$$

where plants G_j, $j = 1,2,3,4$ have actuator time delays and gains shown in Table 6.1. These extreme plant models were chosen because they were judged to be the most difficult to simultaneously obtain good performance.

Table 6.1: Extreme plants G_j, $j = 1,2,3,4$

	τ_1	τ_2	k_1	k_2
G_1	0	0	0.8	0.8
G_2	1	1	0.8	1.2
G_3	1	1	1.2	0.8
G_4	1	1	1.2	1.2

With weighting functions as above, a satisfactory design was not achieved, so the order of the weighting functions W_1 and W_2 was increased to give

$$W_1 = \frac{s^2 + w_1 s + w_2}{s(s^2 + w_3 s + w_4)} \begin{bmatrix} w_5 & 0 \\ 0 & w_6 \end{bmatrix} \tag{6.4.20}$$

and

$$W_2 = w_7 \frac{s + w_8}{s + w_9} I_2 \tag{6.4.21}$$

To ensure that the weighting functions are stable and minimum phase, the following inequalities were included in the inequality set (6.4.19):

$$\text{Re}\left\{-w_1 + \sqrt{w_1^2 - 4w_2}\right\} < 0 \tag{6.4.22}$$

$$\text{Re}\left\{-w_3 + \sqrt{w_3^2 - 4w_4}\right\} < 0 \tag{6.4.23}$$

$$-w_8 < 0 \tag{6.4.24}$$

$$-w_9 < 0 \tag{6.4.25}$$

To attempt to satisfy all the performance specifications, a 2nd degree-of-freedom is introduced by setting K_p to be a four state dynamic pre-compensator. K_p has a steady-state gain of $K_s(0)W_2(0)$ to ensure that the steady-state error specifications are met. After implementing the MBP, a design that successfully satisfied inequalities (6.4.18) and (6.4.19) was easily obtained. However, inspection of the closed-loop step responses showed a very large amount of undershoot, so four additional inequalities to restrict the minimum undershoot to -0.1 were included. The undershoot functionals are defined as:

$$\phi_{13} = -\min_t y_1([1\ 0]^T, t) \tag{6.4.26}$$

$$\phi_{14} = -\min_t y_2([1\ 0]^T, t) \tag{6.4.27}$$

$$\phi_{15} = -\min_t y_1([0\ 1]^T, t) \tag{6.4.28}$$

$$\phi_{16} = -\min_t y_2([0\ 1]^T, t) \tag{6.4.29}$$

The prescribed bounds for the undershoot functionals are $\varepsilon_i = 0.1$ for $i = 13, \ldots, 16$. The design criteria were hence amended to

$$\gamma_0(W) \leq \varepsilon_\gamma \tag{6.4.30}$$

$$\phi_i(G_j, W, K_p) \leq \varepsilon_i, \text{ for } j = 1,2,3,4, \quad i = 1,2,\ldots,16 \tag{6.4.31}$$

Using the MBP, the performance shown in Table 6.2 was obtained with the weights

$$W_1 = \frac{(s + 0.311 \pm 0.733j)}{s(s + 0.922 \pm 0.714j)} \begin{bmatrix} 114.2 & 0 \\ 0 & 103.9 \end{bmatrix} \tag{6.4.32}$$

and

$$W_2(s) = 2.737\frac{(s + 0.532)}{(s + 1.617)} I_2 \tag{6.4.33}$$

and pre-compensator

$$K_p \overset{s}{=} K_s(0)W_2(0) \times \left[\begin{array}{cccc|cc} -0.4169 & 0 & 0 & 0 & -0.2374 & -0.05372 \\ 0 & -4.153 & 0 & 0 & -2.189 & -4.586 \\ 0 & 0 & -0.8368 & 0 & 0.0293 & -2.038 \\ 0 & 0 & 0 & -2.359 & 0.0269 & 0.0515 \\ \hline 0.2222 & 3.609 & -1.199 & -1.073 & 3.083 & 1.119 \\ -1.792 & -0.838 & 0.3027 & 4.9136 & -1.529 & 0.4734 \end{array}\right] \tag{6.4.34}$$

The resulting optimal compensator K_s had 13 states. The value of the robustness index was $\gamma_0 = 2.739$.

Table 6.2: Performance requirements and final performance

i	ε_i	$\phi_i(G_1)$	$\phi_i(G_2)$	$\phi_i(G_3)$	$\phi_i(G_4)$
1	1.1	1.036	1.036	1.017	0.998
2	−0.9	−0.841	−0.925	−0.853	−0.930
3	0.5	0.372	0.279	0.465	0.343
4	0.5	0.433	0.427	0.425	0.422
5	−0.35	−0.397	−0.398	−0.399	−0.400
6	0.7	0.602	0.601	0.603	0.600
7	−0.55	−0.591	−0.591	−0.597	−0.596
8	0.5	0.394	0.445	0.324	0.364
9	1.1	1.023	1.007	1.032	0.998
10	−0.9	−0.900	−0.899	−0.958	−0.956
11	50.0	50.53	50.60	50.43	49.59
12	150.0	147.3	147.3	147.3	147.3
13	0.1	0.00	0.056	0.032	0.017
14	0.1	0.029	0.085	0.040	0.020
15	0.1	0.029	0.028	0.075	0.016
16	0.1	0.00	0.027	0.081	0.021

All the step response criteria were satisfied except for $\phi_2(G_1)$ and $\phi_2(G_3)$. The 50 dB gain limit was marginally exceeded by $\phi_{11}(G_1)$, $\phi_{11}(G_2)$ and $\phi_{11}(G_3)$. The step responses of the 16 possible extreme plants, using 5th order Padé approximants for the time delays, are shown in Figure 6.15 along with the maximum singular values of $(I - KG)^{-1}W_1K_p$, the demand to plant input transfer function. The prescribed bounds on the responses are also shown in the plots. Over all the extreme plants, the overshoot, rise-time and cross-coupling in the simulations are no worse than for the four extreme plants used for the design, however, this is not the case for the undershoot. To reduce the undershoot, more extreme plants could have been included in the MoI, but this would be at the expense of additional computational effort. The results compare favourably with other designs for the same problem (Hoyle *et al.*, 1991; Yaniv and Horowitz, 1991). It was found that the prescribed gain and bandwidth bounds, ε_{11} and ε_{12}, were the most significant factors in restricting the performance, if these bounds were sufficiently increased, all the performance specifications could be met.

6.5 Example – High Speed EMS Maglev Vehicle

Magnetically levitated (maglev) vehicles have become a practical proposition through recent technological advances. The two most effective suspension

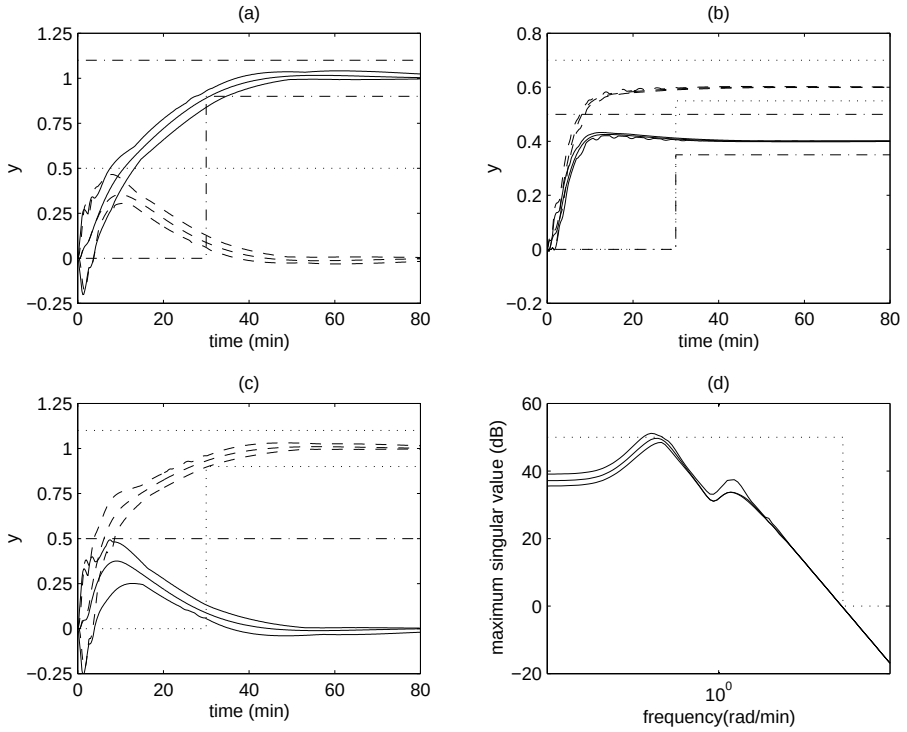

Figure 6.15: Responses of nominal y_1 (——) and y_2 (- - -) envelopes of all extreme plants to (a) input $h(t)\begin{bmatrix}1\\0\end{bmatrix}$, (b) input $h(t)\begin{bmatrix}0.4\\0.6\end{bmatrix}$, (c) input $h(t)\begin{bmatrix}0\\1\end{bmatrix}$ and (d) maximum singular value of nominal and envelopes of all extreme plants for $(I - KG)^{-1}W_1K_p$.

methods are electrodynamic suspension (EDS) and electromagnetic suspension (EMS). EDS requires super-conducting materials in order to produce sufficient repulsive force to propel a vehicle over a conducting sheet. On the other hand, EMS employs the attractive forces of sets of electromagnets acting upwards to levitate the vehicle towards steel tracks. Whilst EDS systems can expect to benefit from technological advancements in super-conducting materials, EMS systems are presently the more successful option. For example, a commercial EMS passenger transport system linking Shanghai city centre with Pudong International Airport is planned.

EMS vehicles are inherently unstable and thus require active control. Moreover, it is necessary to maintain an airgap between the vehicle and supporting guideway in order to avoid undesirable contact between them. Thus the mixed optimisation approach is applied to the problem of designing the suspension controller for a 140 m/s maglev vehicle consisting of a chassis sup-

porting a passenger cabin by means of a secondary suspension of airsprings and hydraulic shock absorbers. The vehicle is shown in cross-section in Figure 6.16. The chassis is levitated by means of DC electro-magnets under active control providing an attractive force to the guideway. The aim of the control is to provide stability to an inherently unstable system, to maintain the airgap, and to ensure the quality of the ride for the passengers.

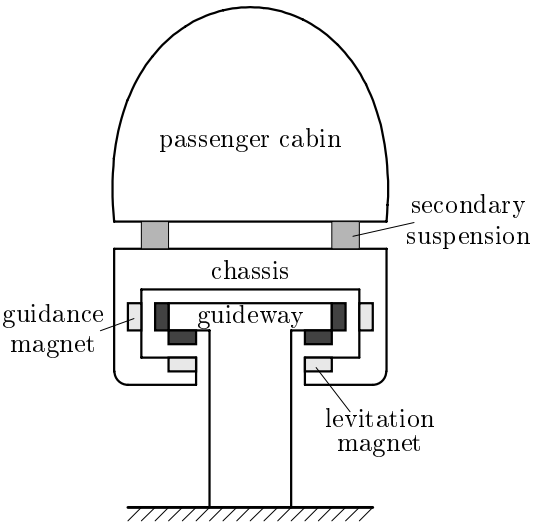

Figure 6.16: Cross-section of maglev vehicle

The design model (Kortum and Utzt, 1984; Sinha, 1987) is for a single electro-magnet considering the vertical movement of the chassis and passenger cabin. The secondary suspension is modelled as a linear spring damper system, and the primary electro-magnetic suspension is described by a nonlinear first order differential equation for the vertical force being derived from the magnet force law and the current/voltage relation. The model configuration is shown in Figure 6.17.

The force, F, exerted by the magnet is

$$F(i,z,t) = \frac{K_m}{2}\left[\frac{i(t)}{z(t)}\right]^2 \tag{6.5.1}$$

where, from steady-state considerations, $K_m = 2mg(z_0/i_0)^2$; and where i is the current (nominal value i_0), z is the gap between magnet and guideway (nominal value z_0), m is the total mass of the vehicle and g is the gravitational constant. If R is the total resistance of the circuit (including the amplifier

output resistance and the magnet winding resistance), then for an instantaneous voltage $v(t)$ across the magnet winding, the excitation current $i(t)$ is controlled by

$$v(t) = Ri(t) + K_m \frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{i(t)}{z(t)}\right] \tag{6.5.2}$$

The chassis has mass m_1 and the passenger cabin has mass m_2. The secondary suspension has a spring constant k and damping constant c. The relationship between the two is assumed linear and satisfies Newton's law:

$$m_1\ddot{x}_1 + c(\dot{x}_1 - \dot{x}_2) + k(x_1 - x_2) = mg - F \tag{6.5.3}$$

$$m_2\ddot{x}_2 + c(\dot{x}_2 - \dot{x}_1) + k(x_2 - x_1) = 0 \tag{6.5.4}$$

where x_1 is the absolute position of the chassis, and x_2 is the absolute position of the passenger cabin.

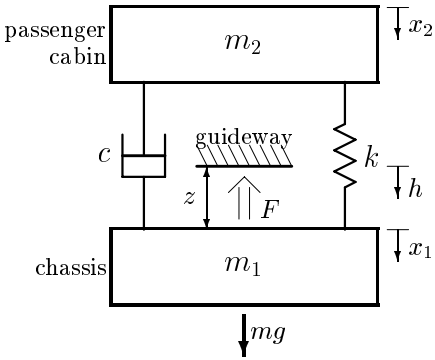

Figure 6.17: Maglev vehicle model

The air gap is related to the absolute chassis position by

$$z(t) = x_1(t) - h_g(t) \tag{6.5.5}$$

where $h_g(t)$ is the disturbance resulting from variations in the guideway profile.

Typical values of the parameters are (Kortum and Utzt, 1984)

$$\begin{array}{lcl} R & = & 1.8 \text{ Ohms} \\ i_0 & = & 50 \text{ A} \\ z_0 & = & 12 \text{ mm} \\ m & = & 5333 \text{ Kg} \\ m_1 & = & 1767 \text{ Kg} \\ m_2 & = & 3550 \text{ Kg} \\ g & = & 9.807 \text{ m/s}^2 \end{array} \tag{6.5.6}$$

The secondary suspension parameters, d and k, are design variables. A linearised State-space nominal model $G(s)$ is given by

$$\begin{bmatrix} \dot{F} \\ \dot{x}_1 \\ \dot{x}_2 \\ \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} = \begin{bmatrix} -(\frac{i_0}{z_0})^2 R & \frac{-R}{L} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \frac{-1}{m_1} & \frac{-k}{m_1} & \frac{k}{m_1} & \frac{-c}{m_1} & \frac{c}{m_1} \\ 0 & \frac{k}{m_2} & \frac{-k}{m_2} & \frac{c}{m_2} & \frac{-c}{m_2} \end{bmatrix} \begin{bmatrix} F \\ x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} + \begin{bmatrix} \frac{i_0}{z_0} & (\frac{i_0}{z_0})^2 R \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v \\ h \end{bmatrix} \tag{6.5.7}$$

where $L = K/z_0$ is the nominal inductance.

The maglev vehicle is inherently unstable and thus the primary goal of the controller is to provide stability. An electro-magnet excited by a constant voltage will either clamp to the rail or fall as the attractive forces decrease with an increasing airgap. However, as well as ensuring the stability of the system, there are three other important considerations that are required when assessing the effectiveness of a control scheme:

a) In order to avoid undesirable contact between the guideway and skids, the allowed air gap should be maintained between 7 and 17 mm with a nominal value of 12 mm. Contact leads to vibration, noise, friction and possible damage to the vehicle and/or guideway and thus the control, or value of this output, is critical as the air gap has a definite bound. The nominal air gap should not be large as the lifting capacity decreases with a growing air gap and feasible power consumption and magnet weights would be unrealisable if the gap were too large. The minimum air gap of 7 mm assures a sufficient safety margin to accommodate the failure of a single magnet (Kortum and Utzt, 1984) and therefore the maximum error between the actual air gap and nominal air gap is 5 mm.

b) Vertical acceleration experienced by passengers may be used as a measure of ride comfort. Typically, this should not exceed 0.5 m/s^2 in either direction (Kortum and Utzt, 1984). However, this is not a rigid requirement and it may be allowed to increase to as much as 1.0 m/s^2.

c) Additionally, the control effort required should be within feasible limits of the electro-magnets. Thus, the required control voltage should be within ± 600V (Kortum and Utzt, 1984).

The performance measures used must reflect the objective of the control, namely the maximum airgap and the quality of the ride. In addition, there is a constraint on the amount of control voltage that can be applied. Hence, performance indices based on the maximum variation in the airgap, z, the maximum variation in control voltage v and the maximum acceleration experienced by the passengers $\ddot{x}_2$ are proposed.

The major disturbance to the system is from variations in the guideway height, and the following bound on the guideway variations has been suggested for a 140 m/s vehicle by Muller (1977):

$$D = \sup\left\{ \left| \dot{h}_g(t) \right| : t \geq 0 \right\} = 30 \text{ mm/s} \tag{6.5.8}$$

Now, from Zakian (1986) and Whidborne and Liu (1993), for all possible $h_g(t)$ such that $\sup\left\{\left|\dot{h}_g(t)\right| : t \geq 0\right\} \leq D$,

$$\sup\{|y_i(t, h_g)| : t \geq 0\} = D \int_0^\infty |y_i(\tau, 1)| \, \mathrm{d}\tau \tag{6.5.9}$$

where $y_i(\tau, 1)$ is the unit step response of the ith output of the linear closed-loop system. Thus, using (6.5.8) and (6.5.9), nominal performance functions for the linear closed-loop system can be defined on the airgap, passenger acceleration and control voltage.

To measure the performance of the closed-loop nonlinear system, the responses of the nonlinear system outputs $y_i'(t, h_{test})$ to a test input, $h_{test}(t)$, can be calculated, and the maximum values evaluated. The test input, $h_{test}(t)$, (shown at the top of Figure 6.18) was chosen to correspond to a severe guideway disturbance [Equation (6.5.8)] and represents a level guideway encountering a constant gradient of 30 mm/s immediately followed by a negative gradient of 30 mm/s.

A maximum power spectral density (PSD) $\Phi_{\max}(\omega)$ of the passenger cabin acceleration has been recommended by the US Department of Transportation as a minimum ride quality standard (Sinha, 1987). A performance functional can be defined based on this recommendation. The PSD of the track variations can be modelled as $\Phi_{h_g h_g}(\omega) = Av/\omega^2$, where A depends on the track quality and v is the speed. The p.s.d. of the passenger cabin is thus

$$\Phi_{\ddot{x}_2 \ddot{x}_2}(\omega) = \left|T_{\ddot{x}_2 h_g}(\omega)\right|^2 \frac{Av}{\omega^2} \tag{6.5.10}$$

where $T_{\ddot{x}_2 h_g}$ represents the transfer function between h_g and $\ddot{x}_2$.

For the design of the maglev EMS control system, the objective functions are defined as

$$\begin{aligned}
\phi_1 &= D \int_0^\infty |z(\tau, 1)| \, \mathrm{d}\tau \\
\phi_2 &= D \int_0^\infty |\ddot{x}_2(\tau, 1)| \, \mathrm{d}\tau \\
\phi_3 &= D \int_0^\infty |v(\tau, 1)| \, \mathrm{d}\tau \\
\phi_4 &= \max_t\{|z'(t, h_{test})|\} \\
\phi_5 &= \max_t\{|\ddot{x}_2'(t, h_{test})|\} \\
\phi_6 &= \max_t\{|v'(t, h_{test})|\} \\
\phi_7 &= \max_\omega\{\Phi_{\ddot{x}_2 \ddot{x}_2}(\omega) - \Phi_{\max}(\omega)\}
\end{aligned}$$

From Whidborne (1993), suitable goals for the objective functions are $\varepsilon_\gamma = 5$, $\varepsilon_1, \varepsilon_4 = 5$mm, $\varepsilon_2, \varepsilon_5 = 500$mm/s, $\varepsilon_3, \varepsilon_6 = 600$ V and $\varepsilon_7 = 0$. The goal ε_γ is set to 5.0.

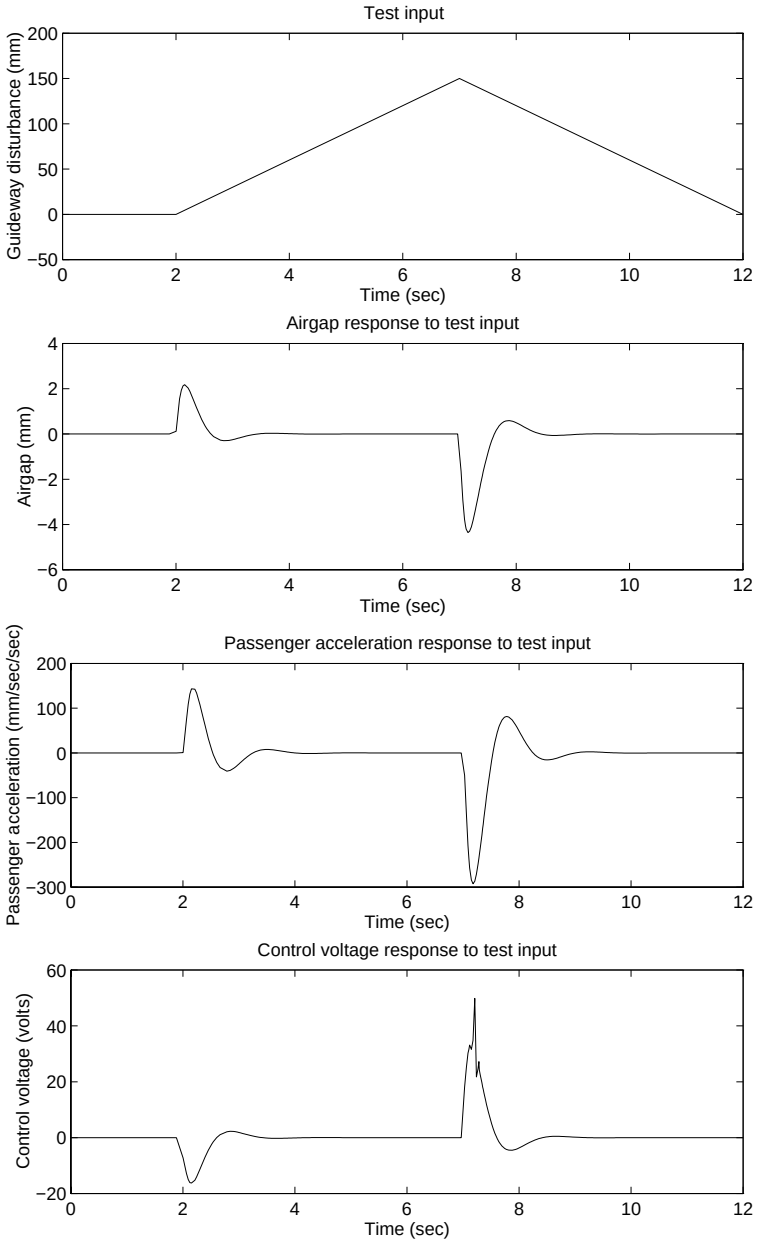

Figure 6.18: Nonlinear system responses to test input h_{test}

Measurements of the air gap, z, and the passenger cabin acceleration, $\ddot{x}_2$, are used for feedback. The weighting function configurations are

$$W_1 = p_1 \frac{(s^2 + p_2 s + p_3)}{(s^2 + p_4 s + p_5)}, \quad W_2 = \text{diag}\left(p_6 \frac{(s + p_7)}{(s + p_8)}, p_9\right) \tag{6.5.11}$$

The secondary suspension stiffness and damping factors c and k are included as design parameters p_{10} and p_{11}.

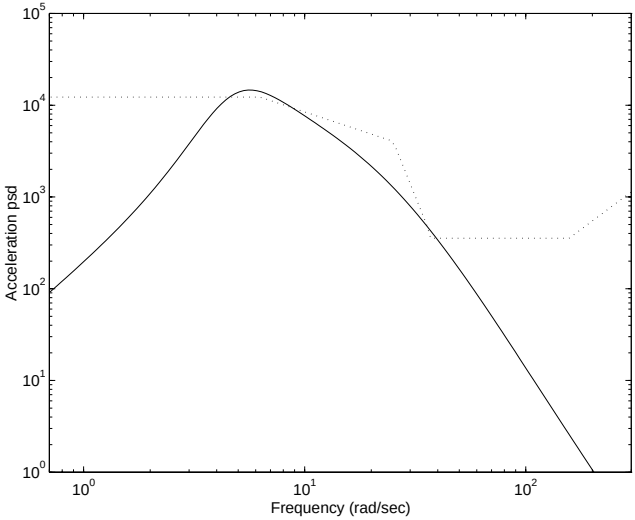

Figure 6.19: Passenger cabin PSD (——) and ride quality standard PSD (- - -)

The multiobjective simulated annealing algorithm (Whidborne *et al.*, 1996a; Chipperfield *et al.*, 1999) was run with the LSDP and a solution found which met all the design goals except the performance functional based on the power spectral density specification, which was only marginally exceeded. The performance objective functions of the design are

$$\begin{aligned}
\gamma_0 &= 2.64 \\
\phi_1 &= 5.06 \text{ mm} \\
\phi_2 &= 391.4 \text{ mm/s}^2 \\
\phi_3 &= 201.9 \text{ V} \\
\phi_4 &= 4.38 \text{ mm} \\
\phi_5 &= 291.1 \text{ mm/s}^2 \\
\phi_6 &= 33.6 \text{ V} \\
\phi_7 &= 0.075
\end{aligned}$$

The designed weighting functions are

$$W_1 = 903.2\frac{(s^2+169.6s+485.4)}{(s^2+69.5s+488.3)} \quad (6.5.12)$$

$$W_2 = \text{diag}\left(426.9\frac{(s+256.2)}{(s+379.2)}, 3.23\right) \quad (6.5.13)$$

and the designed secondary suspension stiffness and damping factors are $c = 90.3$ and $d = 20.0$. The nonlinear simulation response and power spectral density of the final design are shown in Figures 6.18 and 6.19 respectively.

6.6 Summary

The mixed-optimisation approach to multiobjective robust control system design combines the flexibility of numerical optimisation-type techniques with analytical optimisation in an effective and practical manner. The MoI is interactive, thus providing flexibility to the designer in formulating a realistic problem and in determining design trade-offs. Unlike the LSDP, closed-loop performance is *explicitly* considered in the formulation of the design problem, and can include both time and frequency domain performance indices. However, it was found in practice that the initial choice of weighting function parameters is very important in the subsequent progress of the numerical search, and the conventional LSDP approach was seen as useful in choosing the initial parameters and the weighting function structures.

The approach suggested here has some advantages over the usual implementation of the MoI. In the usual implementation, a search is conducted in a set of fixed order controllers to try and find a feasible point. In the approach here, the search is restricted to controllers which are already robustly stable, thus the problem of finding a stability region does not exist. The mixed-optimisation approach is particularly suited to the normalised coprime factor stabilisation problem approach because no γ-iteration is required. The MoI can be combined with H_∞ optimisation methods which require γ-iteration (Whidborne *et al.*, 1995a), but the process is considerably slower.

Chapter 7

Multiobjective Control of Critical Systems

7.1 Introduction

A problem that occurs frequently in control engineering is to control outputs (usually the errors) of a system subjected to random external inputs (reference inputs and/or disturbances) so that the absolute value of the outputs is within prescribed bounds. Any violation of the bounds results in unacceptable, perhaps catastrophic, operation. For example, the electro-magnetic suspension control system for a magnetically levitated (maglev) vehicle (Whidborne, 1993). It is necessary that the airgap between the guideway and the levitating magnets is maintained for the effective operation of the system in spite of disturbance resulting from variations in the guideway profile. Such a system is said to be critical (Zakian, 1989; Whidborne and Liu, 1993). As yet, the design of critical systems has been studied in the continuous-time domain (Zakian, 1986, 1987, 1989; Rutland, 1992), the discrete-time domain (Liu, 1992a,b,c,d; Whidborne, 1992) and the frequency domain (Liu, 1990, 1992a, 1993; Liu *et al.*, 1995). Many important and useful results are presented.

Usually, a control system is probably subjected to two kinds of uncertainties. One is the uncertainty in external inputs which impinge on the system, called the external uncertainty. The other is the uncertainty in the mathematical models used to represent the process, called the internal uncertainty. Though the external uncertainty in critical systems has been considered in many papers, the internal uncertainty in these systems has been paid little attention. Therefore, the robust control, which refers to the maintenance of design specification in the presence of uncertainties, of MIMO critical systems with external and internal uncertainties is a very interesting and important subject.

The robust control design of multivariable critical systems with external

and internal uncertainties is considered in this chapter. The design problem formulated as a set of objectives includes the output performance criteria in the time domain and the robust performance criterion in the frequency domain. Some relationships between an input space, an unmodelling error space, a controller, output performance and robust performance are established for SISO and MIMO cases so that the design problem is largely simplified and can be solved in the frequency domain using multiobjective optimisation methods and H^∞ optimisation techniques.

7.2 Critical Control Systems

A critical system in control engineering is to control an output v (usually the error) of a system subjected to random external inputs w (reference inputs and/or disturbances) so that the absolute value of the output is within a prescribed bound ε, that is,

$$|v(t,w)| \leq \varepsilon, \quad \forall t \in \mathcal{R} \tag{7.2.1}$$

Any violation of the bound results in unacceptable, and perhaps catastrophic, operation. There is a wide variety of critical systems; a small selection is listed here:

a) Motor vehicles are required to travel within their prescribed lanes and any departure from those can have serious consequences.
b) The force on yarns of a loom in textile mills (Kipp, 1989) is required to remain within specified narrow bounds so that the yarn is not broken, despite disturbances such as quality changes of the yarns at random times. Violation of this requirement results in the loom stopping.
c) Air traffic controllers require aircraft to remain within a specified narrow altitude bound (Franklin *et al.*, 1986). This is often achieved by means of an altitude-hold autopilot.
d) The catalytic converter, which is a legal requirement on motor cars to reduce pollution, only works if the engine's air/fuel ratio is within certain narrow bounds (Franklin *et al.*, 1986).
e) To ensure uninterrupted communication, an antenna dish tracking a communications satellite must at all times point accurately at the satellite to within a small tolerance (Franklin and Powell, 1980).

Zakian and Al-Naib (1973) suggested that many feedback control design problems should be posed as the satisfaction of a set of inequalities, rather than the minimisation of some objective functions with inequalities acting as side-constraints, and proposed the method of inequalities (MoI), which makes use of numerical methods.

In essence, the MoI involves expressing the design problem by means of an appropriate set of inequalities and then finding a solution of the inequalities by numerical methods. The design objectives are formulated in a set of

inequalities of the form

$$\phi_i(p) \leq \varepsilon_i, \quad \text{for} \quad i = 1, 2, ..., n \tag{7.2.2}$$

$\phi_i(p)$ is a real scalar and represents an aspect of the dynamic behaviour of the system under design: these aspects include, for example, steady-state error, the rise time, the overshoot, the setting time, the undershoot, the maximal and minimal controller output and so on. p is a real vector representing the parameters of the controller which is being designed. The finite positive real number ε_i represents the numerical bound on the particular aspect of dynamic behaviour described by $\phi_i(p)$.

Since the MoI gives a more accurate formal representation of many design problems, it caused considerable attention in the design of control systems and was developed along several closely related complementary directions (Becker *et al.*, 1979; Polak, 1979; Zakian, 1979; Mayne *et al.*, 1981). Its usefulness has been shown by a number of comparative studies (*e.g.* Taiwo, 1978; Dahshan, 1981) and applications (*e.g.* Bollinger *et al.*, 1979; Coelho, 1979; Crossley and Dahshan, 1982; Taiwo, 1979, 1980; Inooka, 1982). Some aspects of the method of inequalities are explained in Patel and Munro (1982), Maciejowski (1989) and Ng (1989).

However, it soon became clear that the development of control system design was no longer hampered by the lack of efficient mathematical programming methods but by the inadequacy of design criteria for expressing the real aims of control. The research work in critical control systems (Zakian, 1979, 1983) led to a design framework, called the critical control framework (Zakian, 1986, 1987, 1989, 1991). Based on an explicit definition of control, the critical control framework provides control system designers with a set of appropriate design criteria.

It is well known that a system can be described in many ways. There are two main descriptions. One is a larger description, in which a concrete situation can be represented or modelled by an input function space describing the external environment to which the system is subjected and an input-output rule mapping the input space into an output function space. The other is a more conventional and also more restricted description which does not take into account the input space and which considers the system to be simply that part of the concrete situation which is represented by the input-output rule with which, given a suitable input, the corresponding output can be obtained.

Much of control theory is based on the restricted description of a system. Consequently, the concepts of performance (for example, IAE concept, *i.e.* integral of absolute error), of controllability (see, for example, Rosenbrock, 1970) and of stability (see, for example, Willems, 1970) do not involve an input space. In contrast, the critical control framework is based on the larger description. By providing means of accommodating a model of system's environment or inputs which can be transient and/or persistent, and facilities

for explicit statement of the control objectives, it overcomes the two important disadvantages of conventional control approaches. One is that most of the criteria in conventional control approaches specify performance measures with respect to certain test-input functions, *e.g.* a step, a sinusoid, white noise, which are often not a realistic representation of the environment of the system. The other is that the design objectives are often not stated or formulated explicitly. Since the critical control framework involves an input space, the ensuing concepts of performance, controllability and stability are related to the input space (Zakian, 1986). Further, by laying down a clear relation between the input space, the input-output mapping and the output performance, the critical control framework provides a sound theoretical framework for assessing the appropriateness and relative merits of any design criterion (Zakian, 1986, 1987).

The critical control framework has been applied to the design of many practical systems, *e.g.* the brake control of heavy duty trucks (Bada, 1987a), a maglev vehicle suspension control system (Whidborne, 1993), a hydraulic turbine control system (Liu *et al.*, 1990), the control of motor vehicles (Rutland, 1992). In addition, it has been extended to multivariable systems (Zakian, 1987, 1989).

7.3 Critical System Descriptions

First, several definitions are introduced, which are used in this chapter.

Hardy space $\mathcal{H}^p$ ($p \in [1, \infty]$) consists of all complex-valued functions $H(s)$ of a complex variable s which are analytic in the open right half-plane (RHP), Re $s > 0$ and satisfy the condition

$$\| H \|_p := \sup \left\{ \left[\frac{1}{2\pi} \int_{-\infty}^{+\infty} |H(\sigma + j\omega)|^p d\omega \right]^{1/p} : \sigma > 0 \right\} < \infty, \quad p \in [1, \infty) \tag{7.3.1}$$

$$\| H \|_\infty := \sup\{|H(s)| : Re \quad s > 0\} < \infty \tag{7.3.2}$$

In the time domain, define

$$\| h \|_m = \left(\int_0^{+\infty} |h(t)|^m dt \right)^{1/m}, \quad 1 \le m < \infty \tag{7.3.3}$$

$$\| h \|_\infty = \sup\{|h(t)| : t \ge 0\} \tag{7.3.4}$$

where $h(t)$ is a piecewise-continuous function $h : [0, \infty) \to \mathcal{R}$.

Note that upper-case letters are used to represent the complex-valued functions and lower-case letters represent the real-valued functions in this chapter. In addition, there exist the following relations between the complex-valued and real-valued functions: $H = \mathcal{L}[h]$ and $h = \mathcal{L}^{-1}[H]$, where $\mathcal{L}[\cdot]$ is the Laplace

transform and $\mathcal{L}^{-1}[\cdot]$ is the inverse Laplace transform. The same norm symbol $\|\cdot\|$ in the frequency and time domains are used. Context determines which is intended.

In the frequency domain, the standard feedback continuous system (G, K) is shown in Figure 7.1, where G is a plant, K is a controller, R, F, E, U and Y are the reference input, the disturbance, the error, the control and the output of the system, respectively. Let W denote the externally applied input which is either the reference input R or the disturbance F and let V denote the generalised output which is either the output Y or the error E.

Note that if there is not a reference input, then the design of the systems (G, K) is a regulation problem, otherwise the design of the system (G, K) is a control problem. In fact, the regulation problem is the special case of the control problem. A useful preliminary concept is that of well posedness.

Definition 7.1 *The system (G, K) is well posed if the transfer functions from the external inputs to the generalised outputs exist and are proper.*

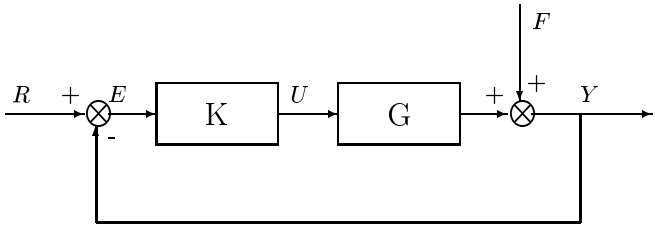

Figure 7.1: Standard feedback system (G, K)

It is assumed from now on that the systems (G, K) are well posed. A control system can be represented by a pair $(S, \mathcal{E})$, where $\mathcal{E}$ is an external input function space and S is an input-output rule that maps the external input function space $\mathcal{E}$ into an output function space.

In the time domain the input-output rule S is expressed by

$$S : w \mapsto v(w, K) \tag{7.3.5}$$

where w, v and K are the external input, the generalised output and the controller, respectively. The external input function:

$$w : t \mapsto w(t) \tag{7.3.6}$$

maps $\mathcal{R}$ into $\mathcal{R}$ in the continuous-time domain. The generalised output function:

$$v(w, K) : t \mapsto v(t, w, K) \tag{7.3.7}$$

maps $\mathcal{R}$ into $\mathcal{R}$ in the continuous-time domain and depends on the external input w and the controller K. Let $v(t, \delta, K)$ denote the impulse response of the system. The input-output relation in the continuous-time domain is

$$v(t, w, K) = \int_0^t v(t - \tau, \delta, K) w(\tau) d\tau, \quad t \in \mathcal{R}^+ \tag{7.3.8}$$

In the frequency domain, the input-output rule S is expressed by

$$S : W \mapsto V(W, K) \tag{7.3.9}$$

where W and V are the Laplace transforms of the external input w and the generalised output v, respectively. The input-output relation is defined by

$$V(s, W, K) = V(s, \delta, K) W(s) \tag{7.3.10}$$

where $V(s, W, K)$, $V(s, \delta, K)$ and $W(s)$ are the Laplace transforms of $v(t, W, K)$, $v(t, \delta, K)$ and $w(t)$.

It is well known that the measure of performance in control system design is quite important. Some performance functions for critical systems are thus defined, based on their corresponding input spaces. A critical control system is said to be well behaved if it ensures that during all time and all external inputs, the generalised output remains tolerably small. Therefore, the performance measure for critical, continuous-time systems is defined as

$$\phi(K) = \sup \left\{ |v(t, w, K)| : t \in \mathcal{R}^+, w \in \mathcal{F} \right\} \tag{7.3.11}$$

where the output $v(t, w, p)$ is the generalised output, usually the error, but it may be the control effort or other outputs, t is time, $w \in \mathcal{F}$ is the system inputs, which belong to the known function space $\mathcal{F}$ of inputs, and K is the controller. ϕ is the greatest output as time ranges over the half line $[0, \infty)$ and the input ranges over the input space $\mathcal{F}$. The reasons for its preference are:

a) it is a natural measure of performance and it has an obvious physical meaning;
b) the system inputs are explicitly considered;
c) it is the ideal performance index for critical systems.

In critical control systems, the main objective is to ensure that during all time t, the generalised output remains tolerably small. A control system is said to be well behaved if it satisfies a set of performance criteria that include, among other conditions, mathematical statement of how small the performance function has to be. An appropriate mathematical statement for critical continuous time systems is expressed in the form of the following criterion:

$$\phi(K) \leq \varepsilon \tag{7.3.12}$$

where ε is a finite positive number and the largest tolerable value of $\phi(K)$, and the design parameter is the controller K.

7.4 Input Spaces of Systems

The definition of input space plays an important role in critical control systems since it forms an integral part of the control system representation. The choice of input space is crucial to the definition of performance and directly affects the design method of a system. In the literature, the following kinds of external input models have been investigated. First, the external inputs are assumed to be some known standard functions, *e.g.* a unit step, a pulse, sine, etc., which are considered in much of classical control theory (see, for example, Ogata, 1975). Second, it is assumed that the external inputs are a white Gaussian process with zero mean and covariance not greater than unity, which are considered in the well-known LQG optimisation problem (Anderson and Moore, 1971), H^2-optimisation problem (see, *e.g.* Youla *et al.*, 1976) and much of stochastic control theory (Astrom, 1970; Goodwin and Sin, 1984; Clarke *et al.*, 1987). Third, it is assumed that the external inputs are a square-integrable signal of unity energy, which are considered in H^∞-optimisation problem (see, Zames, 1981; Francis and Zames, 1984; Vidyasagar, 1985; Francis, 1987; Doyle *et al.*, 1989). Finally, it is assumed that the external inputs are a persistent bounded casual signal, which is considered in the set theoretic approach (Glover and Schweppe, 1971; Parlos *et al.*, 1988) and in l^1/L^1-optimisation problems (Vidyasagar, 1986, 1991; Dahleh and Pearson, 1987a,b).

In order to design continuous systems, here we introduce an input space in the frequency domain (Liu, 1993).

Definition 7.2 *Suppose that $F_1 \in \mathcal{H}^2(\sigma)$ and $F_2 \in \mathcal{H}^2(1)$, then $F = F_1F_2$ is said to be an input in the frequency domain. The set of all functions F is known as an input space denoted by $\mathcal{F}(\sigma)$, i.e.*

$$\mathcal{F}(\sigma) = \{F_1F_2 : F_1 \in \mathcal{H}^2(\sigma), F_2 \in \mathcal{H}^2(1)\} \tag{7.4.1}$$

where the space $\mathcal{H}^2(\sigma_i)$ $(\sigma_i = 1, \sigma)$ is the set of $\mathcal{H}^2$ Hardy space (Francis, 1987) consisting of functions H(s) satisfying that $H \in \mathcal{H}^2$ and $\| H \|_2 \leq \sigma_i$, for $\sigma_i \geq 0$.

Here one can think of the input W as being generated by the given input F_1 through the transfer function F_2. Similarly, one can think of the input space $\mathcal{F}(\sigma)$ as being generated by a set of given inputs F_1 with H^2-norm no larger than σ through a class of variable transfer functions F_2 with H^2-norm no larger than 1. Thus, the input space $\mathcal{F}(\sigma)$ can be shown as Figure 7.2.

From a practical viewpoint, one can also think of F_1 as a set of signals with finite power and F_2 as the transfer function of a subsystem which might be uncertain but is characterised by $\| F_2 \|_2 \leq 1$. For example, assume that a set of given signals is $f_1 = \exp(-at)$, for $a \in [0.32, \infty)$ and $t \in \mathcal{R}^+$, and a transfer function of the subsystem with uncertainty is $F_2 = 1/(s + b)$, for

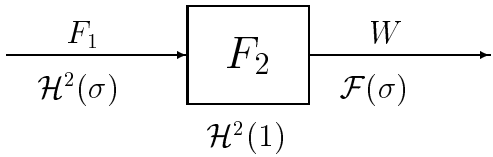

Figure 7.2: Input space $\mathcal{F}(\sigma)$

$b \in [0.5, \infty)$. Then, obviously,

$$\| F_1 \|_2 = \frac{1}{\sqrt{2a}} \leq 1.25 \tag{7.4.2}$$

$$\| F_2 \|_2 = \frac{1}{\sqrt{2b}} \leq 1 \tag{7.4.3}$$

Thus, in this case, the output of the subsystem F_1 can be modelled by the space $\mathcal{F}(1.25)$.

Notice that the inputs w in examples above are transient and it is conjectured that the input space $\mathcal{F}(\sigma)$ consists of transients. The external inputs which the space $\mathcal{F}(\sigma)$ covers are finite-energy signals. It can be used to model the uncertainty of the reference input R and the disturbance F.

For the sake of simplicity, we assume that the reference input R is set to zero and the disturbance F is modelled by the space $\mathcal{F}(\sigma)$. Of course, if we assume that the reference input R is also modelled by the space $\mathcal{F}(\sigma)$, then the results for this case are also similar to the results to be given in this chapter but the output Y (or y) should be replaced by the error E (or e) in the following discussion.

7.5 Multiobjective Critical Control

In the robust control design of critical systems, one should consider four aspects of: external input space (or external uncertainty), internal uncertainty (*e.g.*, modelling error), output performance and robust stability.

a) External input space: This is concerned with environmental conditions which the system is subjected to, including reference inputs and disturbances.

b) Internal uncertainty: This is concerned with modelling errors of the plant, including parameter variations and unmodelled dynamics.

c) Output performance: This is concerned with the ability of the system to reach satisfactory outputs.

d) Robust stability: This is concerned with stability of the system in the case of internal uncertainty.

Based on the definition of the critical systems and the four aspects above, a possible definition of output performance criteria for robust control of MIMO critical systems may be formulated in a set of inequalities of the form:

$$\phi_i(K) \leq \varepsilon_i, \quad \text{for} \quad i = 1, 2, ..., n \tag{7.5.1}$$

where $\phi_i(K)$ is a real scalar and represents the supremum of the absolute value of the i-th output for all time, and for all external inputs and all internal uncertainties to which the system is subjected. It also maybe represents some physical constraints, such as actuator or sensor saturation. K is a stabilising controller which is being designed. The finite positive real number ε_i represents the numerical bound on the aspect of dynamic behaviour described by $\phi_i(K)$.

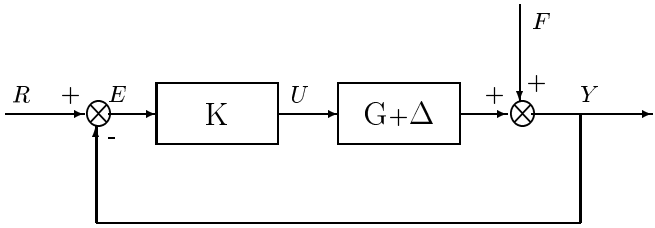

Figure 7.3: Feedback system with uncertainties

Several robust control methods for MIMO systems are studied on the basis of the use of singular values (Lehtomaki, 1981; Francis, 1987; Doyle *et al.*, 1989). It was recognised that singular values are good indicators of matrix size. Using the Small Gain Theorem (Zames, 1966), conditions for guaranteed stability can be obtained in terms of the maximum singular values of perturbing transfer functions. Now, we consider the feedback system shown in Figure 7.1, where $K(s)$ is the controller transfer function and $G(s) + \Delta(s)$ the plant transfer function. $G(s)$ is a known nominal model and $\Delta(s)$ an unknown perturbation representing parameter variations and unmodelled dynamics. Thus, the following bound may be established for the maximum singular value of the perturbation of the system in order to guarantee robust stability:

$$\| \Delta \|_\infty < \left\| K(I + GK)^{-1} \right\|_\infty^{-1} \tag{7.5.2}$$

where $\| . \|_\infty$ is the H^∞-norm of a matrix, *i.e.* the maximal singular value of a matrix.

It is well-known that the robust stability of MIMO systems with internal uncertainty can be guaranteed by (7.5.2). Let us define a dynamical modelling error space which models the internal uncertainty in the frequency domain (Liu *et al.*, 1995).

Definition 7.3 *The modelling error space $\mathcal{D}(\delta)$ is a set of functions $\Delta(s)$ satisfying that*

$$\| \Delta \|_\infty < 1/\delta, \quad \text{for} \quad \delta > 0 \tag{7.5.3}$$

Then from the stability formula (7.5.2) the robust performance criterion of MIMO systems can be formulated as

$$\psi(K) \leq \delta \tag{7.5.4}$$

where

$$\psi(K) = \left\| K(I + GK)^{-1} \right\|_\infty \tag{7.5.5}$$

Therefore, after combining the output performance criteria (7.5.1) and the robust performance criterion (7.5.4), a formulation of the robust control design of MIMO critical systems will be expressed by the following form:

$$\begin{cases} \psi(K) \leq \delta \\ \phi_i(K) \leq \varepsilon_i, \qquad \text{for} \quad i = 1, 2, ..., n \end{cases} \tag{7.5.6}$$

Obviously, the design problem is to find a controller such that (7.5.6) is satisfied. For this design problem, firstly, simplify the output performance functions $\phi_i(K)$ by some analytical methods; secondly, find a controller K satisfying (7.5.6) employing some numerical optimisation methods.

Sometimes, there does not exist a solution to the stabilising controller such that (7.5.6) is satisfied. Therefore, in this case, the design problem is not realistically formulated and must be reformulated. This means that δ and/or ε_i $(i = 1, 2, ..., n)$ in (7.5.6) must be readjusted.

7.6 Control Design of SISO Critical Systems

In this section we consider the robust control design of SISO continuous critical systems in the frequency domain.

It assumes that the disturbance $F \in \mathcal{F}(\sigma)$ and the modelling error $\Delta \in \mathcal{D}(\delta)$ and the reference input $R = 0$ in Figure 7.1. Then, from section 7.3, it is readily appreciated that a measure of output performance of the system in Figure 7.1 can be defined by

$$\phi(K) = \sup\{|y(t, F, \Delta, K)| : t \in \mathcal{R}, F \in \mathcal{F}(\sigma), \Delta \in \mathcal{D}(\delta)\} \tag{7.6.1}$$

where $y(t, F, \Delta, K)$ is the output of the system and is the function of the disturbance F, the modelling error Δ and the controller K.

Roughly speaking, the output performance function $\phi(K)$ is the least upper bound of the absolute output that occurs over all time $t \in \mathcal{R}$, all disturbances $F \in \mathcal{F}(\sigma)$ and all dynamical modelling errors $\Delta \in \mathcal{D}(\delta)$. Thus, from (7.6.1), the output performance criterion is given by

$$\phi(K) \leq \varepsilon \tag{7.6.2}$$

where ε is a real positive number.

Let $\phi^*(K)$ denote the output performance function of the system without internal uncertainty ($\Delta = 0$), that is,

$$\phi^*(K) = \sup\{|y^*(t, F, K)| : t \in \mathcal{R}, F \in \mathcal{F}(\sigma)\} \tag{7.6.3}$$

where $y^*(t, F, K)$ is the output of the system without internal uncertainty.

In addition, here let

$$\psi(K) = \left\| K(1 + GK)^{-1} \right\|_\infty \tag{7.6.4}$$

The robust performance criterion is still given by (7.5.4), *i.e.*

$$\psi(K) \leq \delta \tag{7.6.5}$$

where δ is the positive number given by the modelling error space $\mathcal{D}(\delta)$.

Now, we consider the evaluation of output performance for the system (G, K) with only disturbances in the input space $\mathcal{F}(\sigma)$. Before a main result for this case is given , we need the following lemmas.

Lemma 7.1 *If $Q \in \mathcal{H}^2$, then $\| Q \|_2 = \| q \|_2$, i.e.*

$$\left[\frac{1}{2\pi}\int_{-\infty}^{+\infty} |Q(j\omega)|^2 d\omega\right]^{1/2} = \left[\int_0^{+\infty} q^2(t)dt\right]^{1/2} \tag{7.6.6}$$

Proof: Refer to Plancherel's theorem (Francis, 1987).

Lemma 7.2 *If $P \in \mathcal{H}^\infty$, then*

$$\| P \|_\infty = \sup\{\| PQ \|_2 : Q \in \mathcal{H}^2(\delta)\}/\delta \tag{7.6.7}$$

Proof: If $P \in \mathcal{H}^\infty$, then (Francis, 1987)

$$\| P \|_\infty = \sup\{\| PX \|_2 : X \in \mathcal{H}^2, \| X \|_2 \leq 1\} \tag{7.6.8}$$

Since $Q \in \mathcal{H}^2(\delta), \| Q/\delta \|_2 \leq 1$. Hence,

$$\begin{aligned} \| P \|_\infty &= \sup\{\| PQ/\delta \|_2 : Q/\delta \in \mathcal{H}^2, \| Q/\delta \|_2 \leq 1\} \\ &= \sup\{\| PQ \|_2/\delta : Q \in \mathcal{H}^2(\delta)\} \\ &= \sup\{\| PQ \|_2 : Q \in \mathcal{H}^2(\delta)\}/\delta \end{aligned} \tag{7.6.9}$$

A relation between the input space $\mathcal{F}(\sigma)$, the controller K and the output performance $\phi(K)$ is given by the following theorem (Liu, 1993).

Theorem 7.1 *Assume that* $(1+GK)^{-1} \in \mathcal{RH}^\infty$ *and* $F \in \mathcal{F}(\sigma)$, *then*

$$\phi^*(K) = \left\| (1+GK)^{-1} \right\|_\infty \sigma \tag{7.6.10}$$

Proof: By Definition 7.2, $F = F_1 F_2$, for $F_1 \in \mathcal{H}^2(\sigma)$ and $F_2 \in \mathcal{H}^2(1)$. Let $H = (1+GK)^{-1}F_1$. By Lemma 7.1, then $H \in \mathcal{H}^2$. By the convolution, then the output of the system is

$$y(t,R,K) = \int_0^t h(t-\tau) f_2(\tau) d\tau \tag{7.6.11}$$

Using the Cauchy-Schwarz's inequality (Luenberger, 1969) gives

$$\begin{aligned} |y(t,R,K)| &\leq \left[\int_0^t h^2(\tau)d\tau\right]^{1/2} \left[\int_0^t f_2^2(\tau)d\tau\right]^{1/2} \\ &\leq \left[\int_0^\infty h^2(\tau)d\tau\right]^{1/2} \left[\int_0^\infty f_2^2(\tau)d\tau\right]^{1/2} \\ &= \| h \|_2 \| f_2 \|_2 \end{aligned} \tag{7.6.12}$$

By Lemma 7.1, then

$$|y(t,R,K)| \leq \left\| (1+GK)^{-1}F_1 \right\|_2 \| F_2 \|_2 \tag{7.6.13}$$

Since $F \in \mathcal{F}(\sigma)$,

$$|y(t,R,K)| \leq \left\| (1+GK)^{-1}F_1 \right\|_2 \tag{7.6.14}$$

According to the definition of the performance function $\phi(K)$, it can be deduced from (7.6.1) and (7.6.14) that

$$\phi(K) \leq \gamma \tag{7.6.15}$$

where

$$\gamma = \sup\{ \left\| (1+GK)^{-1}F_1 \right\|_2 : F_1 \in \mathcal{H}^2(\sigma) \} \tag{7.6.16}$$

For $\xi \in (0,\gamma)$ and $\alpha > 1$, let

$$H_n^+ = (1+GK)^{-1}F_{1n}^+, \quad n \in \mathbb{N}^+ \tag{7.6.17}$$

where $F_{1n}^+ \in \mathcal{H}^2(\sigma)$ and makes H_n^+ satisfy

$$\left\| H_n^+ \right\|_2 = \gamma - \xi\alpha^{-n} \tag{7.6.18}$$

Since γ is the supremum of $\| H \|_2$ for $F_1 \in \mathcal{H}^2(\sigma)$, according to the definition of the supremum there certainly exists a sequence $F_{1n}^+ \in \mathcal{H}^2(\sigma)$ $(n \in \mathcal{N}^+)$ which makes (7.6.18) hold. Choose a particular input sequence $F_n^* = F_{1n}^* F_{2n}^*$ $(n \in \mathcal{N}^+)$ defined by $F_{1n}^* = F_{1n}^+$ and $F_{2n}^* = \mathcal{L}\left[f_{2n}^+\right]$, where

$$f_{2n}^+(\tau) = \begin{cases} h_n^+(t-\tau) \| H_n^+ \|_2^{-1} & \text{for} \quad 0 \le \tau \le t \\ 0 & \text{for} \quad \tau > t \quad \text{and} \quad \tau < 0 \end{cases} \tag{7.6.19}$$

and $h_n^+ = \mathcal{L}^{-1}[H_n^+]$. It is obvious that $F_{2n}^+ \in \mathcal{H}^2(1)$. So, $F_n^* \in \mathcal{F}(\sigma)$ for $n \in \mathcal{N}^+$.

Substituting (7.6.19) into (7.6.11) gives

$$y(t, R_n^*, K) = \int_0^t \delta_2 (h_n^+(\tau))^2 \left\| H_n^+ \right\|_2^{-1} d\tau \tag{7.6.20}$$

As $t \to \infty$, then

$$\begin{aligned} y(\infty, R_n^*, K) &= \int_0^\infty (h_n^+(\tau))^2 \left\| H_n^+ \right\|_2^{-1} d\tau \\ &= \left\| h_n^+ \right\|_2^2 \left\| H_n^+ \right\|_2^{-1} \end{aligned} \tag{7.6.21}$$

By Lemma 7.1, then

$$\begin{aligned} y(\infty, R_n^*, K) &= \left\| H_n^+ \right\|_2 \\ &= (\gamma - \xi \alpha^{-n}) \end{aligned} \tag{7.6.22}$$

As $n \to \infty$, then

$$y(\infty, R_\infty^*, K) = \gamma \tag{7.6.23}$$

With (7.6.1), it follows from the above that

$$\phi^*(K) \ge \gamma \sigma \tag{7.6.24}$$

From (7.6.15) and (7.6.23), it can be concluded that

$$\phi^*(K) = \sup\{ \left\| (1 + GK)^{-1} F_1 \right\|_2 : F_1 \in \mathcal{H}^2(\sigma) \} \tag{7.6.25}$$

Using Lemma 7.2 gives

$$\phi^*(K) = \left\| (1 + GK)^{-1} \right\|_\infty \sigma \tag{7.6.26}$$

Therefore, the theorem is established.

Theorem 7.1 shows that the design of the controller K that minimises $\phi(K)$ for the system (G, K), subjected to the input space $\mathcal{F}(\sigma)$, is converted to the problem of minimising $\left\| (1 + GK)^{-1} \right\|_\infty$.

Theorem 7.2 *If all $F \in \mathcal{F}(\sigma)$, all $\Delta \in \mathcal{D}(\delta)$ and $\psi(K) \leq \delta$, then*

$$\frac{\delta}{\delta + \psi(K)}\phi^*(K) \leq \phi(K) \leq \frac{\delta}{\delta - \psi(K)}\phi^*(K) \tag{7.6.27}$$

where $\phi(K)$, $\phi^*(K)$ and $\psi(K)$ are given by (7.6.1), (7.6.3) and (7.6.4), respectively.

Proof: Since $F \in \mathcal{F}(\sigma)$, it is known from Theorem 7.1 that the output performance functions $\phi(K)$ and $\phi^*(K)$ defined in the time domain can be calculated in the frequency domain by, respectively,

$$\phi(K) = \left\| (1 + (G + \Delta)K)^{-1} \right\|_\infty \sigma \tag{7.6.28}$$

$$\phi^*(K) = \left\| (1 + GK)^{-1} \right\|_\infty \sigma \tag{7.6.29}$$

Since

$$(1 + (G + \Delta)K)^{-1} = (1 + GK)^{-1}\left[1 + \frac{\Delta K}{1 + GK}\right]^{-1} \tag{7.6.30}$$

it is from (7.6.28) that

$$\begin{aligned} \phi(K) &= \left\| (1 + GK)^{-1}\left[1 + \frac{\Delta K}{1 + GK}\right]^{-1} \right\|_\infty \sigma \\ &\leq \left\| \left[1 + \frac{\Delta K}{1 + GK}\right]^{-1} \right\|_\infty \left\| (1 + GK)^{-1} \right\|_\infty \sigma \end{aligned} \tag{7.6.31}$$

It is well known that if $\| A \|_\infty < 1$, for $A \in \| A \|_\infty$, then

$$\left\| (I + A)^{-1} \right\|_\infty \leq (1 - \| A \|_\infty)^{-1} \tag{7.6.32}$$

Thus, using the above, $\Delta \in \mathcal{D}(\delta)$ and $\psi(K) \leq \delta$ gives

$$\begin{aligned} \phi(K) &\leq \left[1 - \left\| \frac{\Delta K}{1 + GK} \right\|_\infty\right]^{-1} \phi^*(K) \\ &\leq [1 - \| \Delta K \|_\infty \psi(K)]^{-1} \phi^*(K) \\ &\leq \frac{\delta}{\delta - \psi(K)}\phi^*(K) \end{aligned} \tag{7.6.33}$$

On the other hand, since

$$(1 + GK)^{-1} = (1 + (G + \Delta)K)^{-1}\left[1 + \frac{\Delta K}{1 + GK}\right] \tag{7.6.34}$$

it is from (7.6.29) that

$$\begin{aligned}\phi^*(K) &= \left\| (1+(G+\Delta)K)^{-1}\left[1+\frac{\Delta K}{1+GK}\right]\right\|_\infty \sigma \\ &\le \left\| 1+\frac{\Delta K}{1+GK}\right\|_\infty \left\| (1+(G+\Delta)K)^{-1}\right\|_\infty \sigma \\ &\le \frac{\delta+\psi(K)}{\delta}\phi(K) \qquad (7.6.35)\end{aligned}$$

which gives

$$\phi(K) \ge \frac{\delta}{\delta+\psi(K)}\phi^*(K) \qquad (7.6.36)$$

Therefore, the theorem follows from (7.6.33) and (7.6.35).

Theorem 7.2 shows a relationship between the output performance functions of the system with and without internal uncertainty Δ. According to the result of Theorem 7.2, the robust control design problem (7.5.6) for SISO critical systems can be simplified as

$$\begin{cases} \psi(K) \le \delta \\ \phi^*(K) \le \varepsilon^* \end{cases} \qquad (7.6.37)$$

where

$$\psi(K) = \left\| K(1+GK)^{-1}\right\|_\infty \qquad (7.6.38)$$

$$\phi^*(K) = \left\| (1+GK)^{-1}\right\|_\infty \sigma \qquad (7.6.39)$$

$$\varepsilon^* = \frac{\delta-\psi(K)}{\delta}\varepsilon \qquad (7.6.40)$$

Obviously, it overcomes the difficulty of calculating the output performance function expressed by (7.6.1) in the time domain or (7.6.28) in the frequency domain. And it means that the robust control design of critical systems can be solved by the combination of multiobjective optimisation methods and the H^∞-optimisation method (Zames, 1981; Francis, 1987; Doyle *et al.*, 1989). It is also clear that the controller satisfying (7.6.37) can guarantee that the system satisfies the output performance criterion (7.6.2) and the robust performance criterion (7.6.5).

7.7 Control Design of MIMO Critical Systems

Now, we consider multivariable systems with disturbance

$$F = [F_1, F_2, ..., F_n]^T \qquad (7.7.1)$$

and the output

$$Y = [Y_1, Y_2, ..., Y_n]^T \tag{7.7.2}$$

Let each disturbance F_i belong to a corresponding space $\mathcal{F}(\sigma_i)$, and the modelling error $\Delta \in \mathcal{D}(\delta)$. Also, let us define the following Cartesian spaces:

$$\mathcal{F}(\sigma) = \mathcal{F}(\sigma_1) \times \mathcal{F}(\sigma_2) \times ... \times \mathcal{F}(\sigma_n) \tag{7.7.3}$$

where $\sigma = [\sigma_1, \sigma_2, ..., \sigma_n]$.

Following the scalar case, the output performance of the system is defined by

$$\phi_i(K) = \sup\{|y_i(t, F, \Delta, K)| : t \in \mathcal{R}, F \in \mathcal{F}(\sigma), \Delta \in \mathcal{D}(\delta)\},$$

$$\text{for} \quad i = 1, 2, ..., n \tag{7.7.4}$$

where $y_i(t, F, \Delta, K)$ is the ith output of the system and the function of the disturbance F, the modelling error Δ and the controller K. Thus the output performance criteria are defined by

$$\phi_i(K) \leq \varepsilon_i, \quad \text{for} \quad i = 1, 2, ..., n \tag{7.7.5}$$

where ε_i is a real positive number.

The robust performance criterion is still defined by

$$\psi(K) \leq \delta \tag{7.7.6}$$

where

$$\psi(K) = \left\| K(I + GK)^{-1} \right\|_\infty \tag{7.7.7}$$

and δ is given by the external uncertainty space $\mathcal{D}(\delta)$.

A relationship between the input space $\mathcal{F}(\sigma)$ and the controller K in the frequency domain, and the output performance in the time domain is given by the following theorem (Liu *et al.*, 1995).

Theorem 7.3 *Assume that* $(I + (G + \Delta)K)^{-1} \in \mathcal{H}^\infty$ *and* $F \in \mathcal{F}(\sigma)$, *then*

$$\phi_i(K) = \sum_{j=1}^{n} \left\| e_i^T (I + (G + \Delta)K)^{-1} e_j \right\|_\infty \sigma_j \tag{7.7.8}$$

where $e_i \in \mathcal{R}^{n \times 1}$ *is an identity vector, e.g.* $e_2 = [0, 1, 0, ..., 0]^T$.

Proof: From Figure 7.3, the relation between the output and the disturbance is

$$Y = (I + (G + \Delta)K)^{-1} F \tag{7.7.9}$$

It is known from the above that for $i = 1, 2, ..., n$,

$$Y_i = \sum_{j=1}^{n} e_i^T (I + (G + \Delta)K)^{-1} e_j F_j \tag{7.7.10}$$

where $F_j = F_{j1} F_{j2}$. Let $H_{ij} = e_i^T (1 + (G + \Delta)K)^{-1} e_j F_{j1}$. Since $e_i^T (I + (G + \Delta)K)^{-1} e_j \in \mathcal{H}^\infty$ and $F_{j1} \in \mathcal{H}^2$, by the definition of $\mathcal{H}^2$, it is clear that $H_{ij} \in \mathcal{H}^2$. In terms of the convolution, then the output of the system is

$$y_i(t, F, \Delta, K) = \sum_{j=1}^{n} \int_0^t h_{ij}(t - \tau) f_{j2}(\tau) d\tau \tag{7.7.11}$$

Using the Cauchy-Schwarz inequality (Luenberger, 1969) gives

$$\begin{aligned} |y_i(t, F, \Delta, K)| &\leq \sum_{j=1}^{n} \left[\int_0^t h_{ij}^2(\tau) d\tau \right]^{\frac{1}{2}} \left[\int_0^t f_{j2}^2(\tau) d\tau \right]^{\frac{1}{2}} \\ &\leq \sum_{j=1}^{n} \left[\int_0^\infty h_{ij}^2(\tau) d\tau \right]^{\frac{1}{2}} \left[\int_0^\infty f_{j2}^2(\tau) d\tau \right]^{\frac{1}{2}} \\ &= \sum_{j=1}^{n} \| h_{ij} \|_2 \| f_{j2} \|_2 \end{aligned} \tag{7.7.12}$$

Using Lemma 7.1, then, $\| h_{ij} \|_2 = \| H_{ij} \|_2$ and $\| f_{j2} \|_2 = \| F_{j2} \|_2$. Thus, the above leads to

$$|y_i(t, F, \Delta, K)| \leq \sum_{j=1}^{n} \left\| e_i^T (1 + (G + \Delta)K)^{-1} e_j F_{j1} \right\|_2 \| F_{j2} \|_2 \tag{7.7.13}$$

Since $F_{j2} \in \mathcal{H}^2(1)$, for $j = 1, 2, ..., n$,

$$|y_i(t, F, \Delta, K)| \leq \sum_{j=1}^{n} \left\| e_i^T (1 + (G + \Delta)K)^{-1} e_j F_{j1} \right\|_2 \tag{7.7.14}$$

From the above it can be seen that the item on the right hand side is not the function of the time t. According to the definition of the output performance $\phi_i(K)$, it can be deduced that

$$\phi_i(K) \leq \sum_{j=1}^{n} \gamma_{ij} \tag{7.7.15}$$

where

$$\gamma_{ij} = \sup\{ \left\| e_i^T (1 + (G + \Delta)K)^{-1} e_j F_{j1} \right\|_2 : F_{j1} \in \mathcal{H}^2(\sigma_j) \} \tag{7.7.16}$$

For $\delta_{ij} \in (0, \gamma_{ij})$ and $\alpha_{ij} > 1$, let

$$H^+_{ij,m} = e_i^T (1 + (G + \Delta)K)^{-1} e_j F^+_{j1,m}, \quad m \in \mathcal{N}^+ = \{0, 1, 2, ...\} \tag{7.7.17}$$

where $F^+_{j1,m} \in \mathcal{H}^2(\sigma_j)$ and makes $H^+_{ij,m}$ satisfy

$$\| H^+_{ij,m} \|_2 = \gamma_{ij} - \delta_{ij} \alpha_{ij}^{-m} \tag{7.7.18}$$

According to the definition of the supremum there certainly exists a sequence $F^+_{j1,m} \in \mathcal{H}^2(\sigma_j)$ $(m \in \mathcal{N}^+)$ that makes (7.7.18) hold. Choose a particular input sequence $F^*_m = [F^*_{1,m}, F^*_{2,m}, ..., F^*_{n,m}]$ which is defined as follows: for $j = 1, 2, ..., n$, $F^*_{j,m} = F^*_{j1,m} F^*_{j2,m}$, $F^*_{j1,m} = F^+_{j1,m}$ and $F^*_{j2,m} = \mathcal{L}\left[f^+_{j2,m}\right]$, where

$$f^+_{j2,m}(\tau) = \begin{cases} h^+_{ij,m}(t - \tau) \| H^+_{ij,m} \|_2^{-1} & \text{for} \quad 0 \le \tau \le t \\ 0 & \text{otherwise} \end{cases} \tag{7.7.19}$$

and $h^+_{ij,m} = \mathcal{L}^{-1}\left[H^+_{ij,m}\right]$. It is obvious that $F^*_{j2,m} \in \mathcal{H}^2(1)$. So, $F^*_{j,m} \in \mathcal{F}(\sigma_j)$. Substituting (7.7.19) into (7.7.11) gives

$$y_i(t, F^*_m, \Delta, K) = \sum_{j=1}^{n} \int_0^t (h^+_{ij,m}(\tau))^2 \| H^+_{ij,m} \|_2^{-1} d\tau \tag{7.7.20}$$

As $t \to \infty$, then

$$\begin{aligned} y_i(\infty, F^*_m, \Delta, K) &= \sum_{j=1}^{n} \int_0^{\infty} (h^+_{ij,m}(\tau))^2 \| H^+_{ij,m} \|_2^{-1} d\tau \\ &= \sum_{j=1}^{n} \| h^+_{ij,m} \|_2^2 \| H^+_{ij,m} \|_2^{-1} \end{aligned} \tag{7.7.21}$$

Using Plancherel's theorem gives

$$y_i(\infty, F^*_m, \Delta, K) = \sum_{j=1}^{n} \| H^+_{ij,m} \|_2 = \sum_{j=1}^{n} (\gamma_{ij} - \delta_{ij} \alpha_{ij}^{-m}) \tag{7.7.22}$$

As $m \to \infty$, then $\alpha_{ij}^{-m} \to 0$. Thus

$$y_i(\infty, F^*_\infty, \Delta, K) = \sum_{j=1}^{n} \gamma_{ij} \tag{7.7.23}$$

From the definition of the output performance $\phi_i(K)$ and the result above, it is easy to know that $\phi_i(K) \ge y_i(\infty, F^*_\infty, \Delta, K)$, that is

$$\phi_i(K) \ge \sum_{j=1}^{n} \gamma_{ij} \tag{7.7.24}$$

From (7.7.15) and (7.7.24), it can be concluded that

$$\phi_i(K) = \sum_{j=1}^{n} \sup\{\| e_i^T(1 + (G + \Delta)K)^{-1} e_j F_{j1} \|_2 : F_{j1} \in \mathcal{H}^2(\sigma_j)\} \quad (7.7.25)$$

It is well known that (Francis, 1987)

$$\begin{aligned} &\sup\{\| e_i^T(1 + (G + \Delta)K)^{-1} e_j F_{j1} \|_2 : F_{j1} \in \mathcal{H}^2(\sigma_j)\} \\ &\qquad = \| e_i^T(1 + (G + \Delta)K)^{-1} e_j \|_\infty \sigma_j \end{aligned} \quad (7.7.26)$$

Thus,

$$\phi_i(K) = \sum_{j=1}^{n} \| e_i^T(1 + (G + \Delta)K)^{-1} e_j \|_\infty \sigma_j \quad (7.7.27)$$

Therefore, the theorem is proved.

If there is not internal uncertainty, *i.e.* $\Delta = 0$, it can be known from Theorem 7.3 that the relationship between the input space $\mathcal{F}(\sigma)$, the controller K and the output performance $\phi_i(K)$ is

$$\phi_i^*(K) = \sum_{j=1}^{n} \| e_i^T(1 + GK)^{-1} e_j \|_\infty \sigma_j \quad (7.7.28)$$

for $i = 1, 2, ..., n$, where $\phi_i^*(K)$ is the i-th output performance function of the system without internal uncertainty. For the multivariable case, the following theorem is obtained.

Theorem 7.4 *If all $F \in \mathcal{F}(\sigma)$, all $\Delta \in \mathcal{D}(\delta)$ and $\psi(K) \leq \delta$, then for $i = 1, 2, ..., n$*

$$\phi_i(K) \leq \frac{\delta\gamma}{\delta - \psi(K)} \phi^+(K) \quad (7.7.29)$$

where

$$\gamma = \sum_{j=1}^{n} \sigma_j \quad (7.7.30)$$

$$\phi^+(K) = \| (I + GK)^{-1} \|_\infty \quad (7.7.31)$$

and $\phi_i(K)$ is defined in (7.7.4).

Proof: As a result of Theorem 7.3, it is known that

$$\phi_i(K) = \sum_{j=1}^{n} \| e_i^T(I + (G + \Delta)K)^{-1} e_j \|_\infty \sigma_j$$

$$\leq \sum_{j=1}^{n} \left\| (I + (G+\Delta)K)^{-1} \right\|_\infty \sigma_j$$

$$\leq \left\| (I + (G+\Delta)K)^{-1} \right\|_\infty \gamma \tag{7.7.32}$$

Since

$$(I + (G+\Delta)K)^{-1} = (I + GK)^{-1}[I + \Delta K(I + GK)^{-1}]^{-1} \tag{7.7.33}$$

from (7.7.32), we have

$$\begin{aligned} \phi_i(K) &\leq \left\| (I + GK)^{-1}[I + \Delta K(I + GK)^{-1}]^{-1} \right\|_\infty \gamma \\ &\leq \left\| [I + \Delta K(I + GK)^{-1}]^{-1} \right\|_\infty \left\| (I + GK)^{-1} \right\|_\infty \gamma \end{aligned} \tag{7.7.34}$$

It is shown from the robust stability condition that $\left\| \Delta K(I + GK)^{-1} \right\|_\infty < 1$. Thus, using (7.6.32) and $\psi(K) \leq \delta$ gives

$$\begin{aligned} \phi_i(K) &\leq \left[1 - \left\| \Delta K(I + GK)^{-1} \right\|_\infty\right]^{-1} \phi^+(K)\gamma \\ &\leq \frac{\delta\gamma}{\delta - \psi(K)}\phi^+(K) \end{aligned} \tag{7.7.35}$$

which, therefore, proves the theorem.

Theorem 7.4 shows that the robust control design problem of MIMO critical systems can be simplified as

$$\begin{cases} \psi(K) \leq \delta \\ \phi^+(K) \leq \varepsilon^+ \end{cases} \tag{7.7.36}$$

where

$$\psi(K) = \left\| K(I + GK)^{-1} \right\|_\infty \tag{7.7.37}$$

$$\phi^+(K) = \left\| (I + GK)^{-1} \right\|_\infty \tag{7.7.38}$$

$$\varepsilon^+ = \frac{\delta - \psi(K)}{\delta\gamma} \max_{i=1,2,\ldots,n} \{\varepsilon_i\} \tag{7.7.39}$$

Similar to the scalar case, (7.7.36) can be solved using multiobjective optimisation methods and H^∞ optimisation method. It is also clear that the output performance criteria (7.7.5) and the robust performance criterion (7.7.6) can be satisfied by the solution of the inequalities (7.7.36). Therefore, the robust control design of critical systems is largely simplified.

7.8 An Example

The following example illustrates the robust control design of multivariable critical systems. Here we consider a two-input two-output plant given by

$$G(s) = \begin{bmatrix} \dfrac{-45s+1}{s^2+5s+6} & \dfrac{24s}{s^2+5s+6} \\ \dfrac{-35s}{s^2+5s+6} & \dfrac{58s+1}{s^2+5s+6} \end{bmatrix} \tag{7.8.1}$$

The external uncertainty (the disturbance) F and the internal uncertainty (the modelling error) Δ which the system is subjected to are assumed to be

$$F \in \mathcal{F}(\sigma), \quad \sigma = [\,0.01 \quad 0.01\,]$$

$$\Delta \in \mathcal{D}(2.5)$$

The output performance functions $\phi_1(K)$ and $\phi_2(K)$ in time domain are required to satisfy

$$\phi_1(K) \le 0.13$$
$$\phi_2(K) \le 0.13$$

It is easy to know that $\gamma = \sum_{i=1}^{2} \sigma_i = 0.02$, $\delta = 2.5$ and $\max_{i=1,2}\{\varepsilon_i\} = 0.13$. According to the result (7.7.36) of Section 7.7, the robust control design problem of the system (7.8.1) can be simplified to find a stabilising controller such that

$$\psi(K) = \left\| K(I+GK)^{-1} \right\|_\infty \le 2.5 \tag{7.8.2}$$

$$\phi^+(K) = \left\| (I+GK)^{-1} \right\|_\infty \le \varepsilon^+ = \frac{0.13(2.5-\psi(K))}{0.05} \tag{7.8.3}$$

are satisfied. Combining the method of inequalities (Whidborne and Liu, 1993) and the H^∞ optimisation techniques (Chiang and Safonov, 1988), a stabilising controller that satisfies both the output and the robust performance requirements (7.8.2) and (7.8.3) is found to be

$$K(s) = \begin{bmatrix} \dfrac{-0.0386s^2-0.0056s-0.0001}{s^3+0.1918s^2+0.0104s+0.0002} & \dfrac{-0.0108s^2+0.0022s+0.0001}{s^3+0.1918s^2+0.0104s+0.0002} \\ \dfrac{-0.0182s^2-0.0012s+0.0001}{s^3+0.1918s^2+0.0104s+0.0002} & \dfrac{-0.0179s^2+0.0107s+0.0005}{s^3+0.1918s^2+0.0104s+0.0002} \end{bmatrix}$$

which gives the following values of the output and robust performance functions

$$\psi(K) = 1.9574 \tag{7.8.4}$$

$$\phi^+(K) = 1.3309 \tag{7.8.5}$$

It is easily obtained from (7.8.3) and (7.8.4) that $\varepsilon^+ = 1.4108$ which is greater than 1.3309. From (7.7.29), we can calculate that $\phi_1(K) \leq 0.1226$ and $\phi_2(K) \leq 0.1226$, which are less than 0.13. Therefore, both the output and the robust performance requirements are satisfied.

7.9 Summary

This chapter has given an introduction to critical control systems and discussed the robust control design of critical systems with external and internal uncertainties. The formulation of the robust control design of these systems is expressed by a set of inequalities which includes output performance criteria in the time domain and robust performance criterion in the frequency domain of the system. Several relationships between an input space, a modelling error space, a controller, output performance and robust performance are established for SISO and MIMO critical systems so that the robust control design problem of these systems is largely simplified.

Chapter 8

Multiobjective Control Using Eigenstructure Assignment

8.1 Introduction

In the 1960s Wonham presented the fundamental result on eigenvalue assignment in linear time-invariant multivariable controllable systems (Wonham, 1967). This states that the closed-loop eigenvalues of any controllable system may be arbitrarily assigned by state feedback control. Later, Moore found that degrees of freedom are available over and above eigenvalue assignment using state feedback control for linear time-invariant multi-input multi-output (MIMO) systems (Moore, 1976). Since then, numerous methods and algorithms involving both state and output feedback control have been developed to exercise those degrees of freedom to give the systems some good performance characteristics (Kautsky *et al.*, 1985; Liu and Patton, 1995, 1996b, 1998a,d,e; Liu and Daley, 1998).

A number of traditional approaches to control system design objectives make use of scalar summation of all weighted objectives in one cost function. Though this method simplifies the approach to optimisation, it is not clear how each objective is affected by the controller. On the other hand, if all of the objectives are considered through the use of the individual cost functions, then the action of each objective on the structure of the controller can be determined.

Most eigenstructure assignment techniques have only paid attention to optimal solutions for one special performance index, *e.g.*, the eigenvalue sensitivity function or the linear quadratic index. However, many practical control systems are required to have the ability to fit simultaneously different

and often conflicting performance objectives as best as possible, for instance, closed-loop stability, low feedback gains and insensitivity to model parameter variations.

This chapter is concerned with multiobjective control system design using eigenstructure assignment. The multiobjective performance indices include the individual eigenvalue sensitivities, low feedback gains, and the sensitivity functions in the frequency domain. The robust performance criteria are expressed by a set of inequalities on the basis of the multiobjective performance indices. In order to obtain an approximate global optimisation for the multiobjective control system design, some numerical algorithms are outlined using eigenstructure assignment, the method of inequalities and genetic algorithms.

8.2 What is Eigenstructure Assignment

Eigenstructure assignment is a design technique which may be used to assign the entire eigenstructure (eigenvalues, and right or left eigenvectors) of a closed-loop linear system via a constant gain full state or output feedback control law. It consists, essentially, of the following steps:

a) Choose a set (or sets) of possible closed-loop eigenvalues (or poles).

b) Compute the associated so-called allowable eigenvector subspaces, which describe the freedom available for closed-loop eigenvector assignment.

c) Select specific eigenvectors from the allowable eigenvector subspaces according to some design strategies.

d) Calculate a control law, appropriate to the chosen eigenstructure.

Consider the following linear time-invariant system

$$\delta x = Ax + Bu \tag{8.2.1}$$

$$y = Cx \tag{8.2.2}$$

where $x \in \mathcal{R}^{n\times 1}$ is the state vector, δx represents $\dot{x}(t)$ for continuous systems and $x(t+1)$ for discrete systems, $u \in \mathcal{R}^{r\times 1}$ is the control input vector, $y \in \mathcal{R}^{m\times 1}$ is the output vector, $A \in \mathcal{R}^{n\times n}$, $B \in \mathcal{R}^{n\times r}$ and $C \in \mathcal{R}^{m\times n}$.

A linear output feedback control law applied to the system above is

$$u = Ky \tag{8.2.3}$$

where the matrix $K \in \mathcal{R}^{r\times m}$ is the output feedback controller gain. The output feedback system is shown in Figure 8.1.

It is well known that $\max\{r, m\}$ eigenvalues are assignable arbitrarily by output feedback. This restricts the choice of eigenvalue and eigenvector pairs. However, if $m + r > n$, the whole spectrum can be assigned, with some restrictions on the eigenvector selection (Kimura, 1975). This also implies that an eigenvector associated with an eigenvalue can be assigned in either the right eigenvector set or the left eigenvector set but not both. In addition,

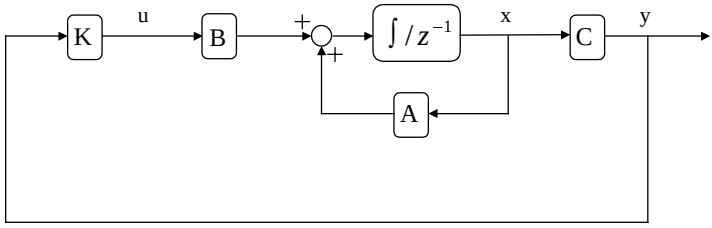

Figure 8.1: Output feedback control system

whatever the selection of eigenvectors is, it may be very difficult to assign some spectrum if the system to be designed suffers from pathological structures. For this case or $m + r \leq n$, the following dynamical compensator will be helpful (Han, 1989; Duan, 1993):

$$\delta z = A_d z + B_d y \tag{8.2.4}$$

$$u = C_d z + D_d y \tag{8.2.5}$$

where $z \in \mathcal{R}^{p\times 1}$ is the state vector of the controller, and A_d, B_d, C_d and D_d are matrices. Combining (8.2.1), (8.2.2), (8.2.4) and (8.2.5) yields the following equivalent system:

$$\delta \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} B & 0 \\ 0 & I \end{bmatrix} \bar{u} \tag{8.2.6}$$

$$\bar{y} = \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} \tag{8.2.7}$$

$$\bar{u} = \begin{bmatrix} D_d & C_d \\ B_d & A_d \end{bmatrix} \bar{y} \tag{8.2.8}$$

This clearly shows that the dynamical compensator can be designed as a standard output feedback controller. The number of closed-loop eigenvalues is equal to the sum of the orders of the system (A, B) and the dynamical controller (A_d, B_d, C_d, D_d), *i.e.*, $n + p$. It is clear that the system matrices change and the number of the system states will be less than the sum of the plant inputs and outputs by appropriate choice of the order of the dynamical compensator. But, for this augmented system, it may be inconvenient to choose separated spectra: one for the plant and the other for the controller.

Thus, without loss of generality, the following assumptions are made for the system (8.2.1) and (8.2.2): rank $(B) = r$, rank$(C) = m$ and $m + r > n$. Then the closed-loop system representation is given by

$$\delta x = (A + BKC)x \tag{8.2.9}$$

Now, let us define a closed-loop self-conjugate eigenvalue set $\Lambda = \{\lambda_i : \lambda_i \in \mathcal{C}, i = 1, 2, ..., \tilde{n}\}$, *i.e.*, the set of the eigenvalues of the closed-loop matrix

$A + BKC$, where $\tilde{n}$ is the number of distinct eigenvalues. For an uncontrollable and/or unobservable system, the uncontrollable and/or unobservable open-loop eigenvalues should be included in the closed-loop self-conjugate eigenvalue set Λ. Though output feedback eigenstructure assignment can not change those eigenvalues, their corresponding eigenvectors may properly be chosen to improve the insensitivity of the closed-loop matrix and the robustness of the closed-loop system.

Denote the algebraic and geometric multiplicity of the i-th eigenvalue λ_i by q_i and r_i, respectively, then, in the Jordan canonical form of the matrix $A + BKC$, there are r_i Jordan blocks, associated with the i-th eigenvalue λ_i, of orders p_{ij}, $j = 1, 2, ..., r_i$, and the following relations:

$$\sum_{j=1}^{r_i} p_{ij} = q_i, \quad \sum_{i=1}^{\bar{n}} q_i = n \tag{8.2.10}$$

In the case of $p_{ij} = 1$, $j = 1, 2, ..., r_i$, $i = 1, 2, ..., \tilde{n}$, the Jordan canonical form of the matrix $A + BKC$ is diagonal. Now, let the right eigenvectors and generalised eigenvectors of the matrix $A + BKC$ corresponding to the eigenvalue λ_i be $R_{ij,k} \in \mathcal{C}^{n\times 1}$, $k = 1, 2, ..., p_{ij}$, $j = 1, 2, ..., r_i$. According to the definition of the right eigenvector and generalised eigenvector for a multiple eigenvalue, then we have

$$(\lambda_i I - A - BKC)R_{ij,k} = -R_{ij,k-1}, \quad R_{ij,0} = 0 \tag{8.2.11}$$

for $k = 1, 2, ..., p_{ij}$, $j = 1, 2, ..., r_i$ and $i = 1, 2, ..., \tilde{n}$. Thus, the right eigenvector and generalised eigenvector matrix is given by

$$R = [R_1, R_2, ..., R_{\bar{n}}] \in \mathcal{C}^{n\times n} \tag{8.2.12}$$

$$R_i = [R_{i1}, R_{i2}, ..., R_{ir_i}] \in \mathcal{C}^{n\times q_i} \tag{8.2.13}$$

$$R_{ij} = [R_{ij,1}, R_{ij,2}, ..., R_{ij,p_{ij}}] \in \mathcal{C}^{n\times p_{ij}} \tag{8.2.14}$$

for $j = 1, 2, ..., r_i$, $i = 1, 2, ..., \tilde{n}$, where, in fact, R_i contains all right eigenvectors and generalised eigenvectors associated with the eigenvalue λ_i.

Similarly, the left eigenvectors and generalised eigenvectors for multiple eigenvalues are defined by

$$L_{ij,k}^T(\lambda_i I - A - BKC) = -L_{ij,k-1}^T, \quad L_{ij,0} = 0 \tag{8.2.15}$$

for $k = 1, 2, ..., p_{ij}$, $j = 1, 2, ..., r_i$ and $i = 1, 2, ..., \tilde{n}$. Then, the left eigenvector and generalised eigenvector matrix is given by

$$L = [L_1, L_2, ..., L_{\bar{n}}] \in \mathcal{C}^{n\times n} \tag{8.2.16}$$

$$L_i = [L_{i1}, L_{i2}, ..., L_{ir_i}] \in \mathcal{C}^{n\times q_i} \tag{8.2.17}$$

$$L_{ij} = [L_{ij,1}, L_{ij,2}, ..., L_{ij,p_{ij}}] \in \mathcal{C}^{n\times p_{ij}} \tag{8.2.18}$$

for $j = 1, 2, ..., r_i$, $i = 1, 2, ..., \tilde{n}$, where the matrix L_i consists of all left eigenvectors and generalised eigenvectors associated with the eigenvalue λ_i.

Therefore, the problem of eigenstructure assignment via output feedback for the system (8.2.1)-(8.2.3) can be stated as follows: Given the closed-loop self-conjugate eigenvalue set Λ, and the integers r_i, p_{ij}, q_i, for $j = 1, 2, ..., r_i$, $i = 1, 2, ..., \tilde{n}$ satisfying (8.2.10), find a real controller K such that the eigenvalues of the matrix $A + BKC$ are in the set Λ by choosing both the right and left eigenvector and generalised eigenvector matrices V and F properly.

8.3 Allowable Eigenvector Subspaces

The allowable eigenvector subspace is, in general, contained (in the context of set theory) in the full state-space ($\mathcal{R}^n$). This subspace consists of a function of the system input, output and state matrices and the choice of closed-loop eigenvalues. Two cases for calculation of these allowable eigenvector subspaces will be outlined in this section. One is that the sets of open- and closed-loop eigenvalues have no elements in common, *i.e.*, $|\lambda_i I - A| \neq 0$, for $i = 1, 2, ..., n$. The other is that the sets of open- and closed-loop eigenvalues intersect, *i.e.*, $|\lambda_i I - A| = 0$, for some i. For example, the uncontrollable open-loop eigenvalues must be in the closed-loop eigenvalue set.

In the case of complex eigenvector assignment each of these methods may be modified so that a real representation of the allowable eigenvector subspace can be computed using real arithmetic only. This real representation is useful when it comes to assigning only the real (or complex) part of a specific element of a desired eigenvector. It is not clear how this may be performed using the complex representation of the allowable subspace.

For the sake of simplicity, it is assumed that all eigenvalues are real and distinct. Denote the i-th eigenvalue and corresponding right and left eigenvectors of the system described by λ_i, R_i and L_i, respectively. The spectral and modal matrices can then be defined as

$$R = [\, R_1 \quad R_2 \quad ... \quad R_n \,] \in \mathcal{R}^{n \times n} \tag{8.3.1}$$

$$L = [\, L_1 \quad L_2 \quad ... \quad L_n \,] \in \mathcal{R}^{n \times n} \tag{8.3.2}$$

and

$$D_\Lambda = \mathrm{diag}[\lambda_1, \lambda_2, ..., \lambda_n] \in \mathcal{R}^{n \times n} \tag{8.3.3}$$

For the general case where the modal matrix D_Λ is of a Jordan form, the following results are also similar.

Case 1: $|\lambda_i I - A| \neq 0$

Consider the eigenstructure of the closed-loop system corresponding to real eigenvalues described by (8.3.1)-(8.3.3). By definition, we may write

$$(A + BKC)R_i = \lambda_i R_i \tag{8.3.4}$$

$$L_i^T(A + BKC) = \lambda_i L_i^T \tag{8.3.5}$$

Since it is assumed that $|\lambda_i I - A| \neq 0$, rearranging these equations results in

$$R_i = (\lambda_i I - A)^{-1} B W_i \tag{8.3.6}$$
$$L_i = (\lambda_i I - A^T)^{-1} C^T V_i \tag{8.3.7}$$

where

$$W_i = KCR_i \tag{8.3.8}$$
$$V_i = K^T B^T L_i \tag{8.3.9}$$

are called the right and left parameter vectors, respectively. It is clear from (8.3.6) and (8.3.7) that the i-th allowable right eigenvector may be chosen from a linear combination of the columns of

$$P_{R,i} = (\lambda_i I - A)^{-1} B \tag{8.3.10}$$

and the i-th allowable left eigenvector may be chosen as a linear combination of the columns of

$$P_{L,i} = (\lambda_i I - A^T)^{-1} C^T \tag{8.3.11}$$

The dimension of each subspace is given by

$$\dim(P_{R,i}) = m \tag{8.3.12}$$
$$\dim(P_{L,i}) = r \tag{8.3.13}$$

The latter two equations show that an allowable i-th right (left) closed-loop eigenvector may be chosen with m (r) degrees of freedom, *i.e.*, that the number of entries of a right (left) eigenvector that may be exactly assigned is m (r).

The above analysis may also be applied to the calculation of allowable eigenvector subspaces corresponding to complex eigenvalues. However, a real representation of this subspace facilitates the assignment of a desired eigenvector, in the case where only the real or imaginary part of an element of a desired eigenvector is specified (the other element is left unconstrained). The calculation of the allowable right eigenvector subspaces corresponding to complex eigenvalues, using real arithmetic only, is performed next. A similar analysis is applied to the calculation of allowable left eigenvector subspaces. Define

$$\lambda_i = \lambda_i^{re} + j\lambda_i^{im} \tag{8.3.14}$$
$$R_i = R_i^{re} + jR_i^{im} \tag{8.3.15}$$

where, from now on, re and im denote the real and the imaginary parts, respectively, and j indicates the imaginary part of a complex number. Hence, continuing from (8.3.4) gives

$$(A + BKC)(R_i^{re} + jR_i^{im}) = (R_i^{re} + jR_i^{im})(\lambda_i^{re} + j\lambda_i^{im}) \tag{8.3.16}$$

Equating real and imaginary parts yields

$$\begin{bmatrix} R_i^{re} \\ R_i^{im} \end{bmatrix} = \begin{bmatrix} \lambda_i^{re} I - A & -\lambda_i^{im} I \\ \lambda_i^{im} I & \lambda_i^{re} I - A \end{bmatrix}^{-1} \begin{bmatrix} B & 0 \\ 0 & B \end{bmatrix} \begin{bmatrix} W_i^{re} \\ W_i^{im} \end{bmatrix} \tag{8.3.17}$$

where

$$W_i^{re} = KCR_i^{re} \tag{8.3.18}$$

$$W_i^{im} = KCR_i^{im} \tag{8.3.19}$$

$W_i = W_i^{re} + jW_i^{im}$ is the right parameter vector. Let

$$\tilde{A} = \begin{bmatrix} \lambda_i^{re} I - A & -\lambda_i^{im} I \\ \lambda_i^{im} I & \lambda_i^{re} I - A \end{bmatrix} \tag{8.3.20}$$

$$\tilde{B} = \begin{bmatrix} B & 0 \\ 0 & B \end{bmatrix} \tag{8.3.21}$$

Since $|\lambda_i I - A| \neq 0$, the matrix $\tilde{A}$ is nonsingular as well. Thus, the real and imaginary parts of the allowable right eigenvector corresponding to the i-th complex eigenvalue may be chosen from a linear combination of the columns of

$$P_{R,i}^c = \tilde{A}^{-1}\tilde{B} \tag{8.3.22}$$

The above technique of expressing the allowable eigenvector subspace is useful for the analytically-derived partial derivatives of the allowable eigenvector subspaces with respect to some parameters of interest. This is important when considering eigenstructure control laws which have optimal low sensitivity performance and stability robustness (Patton and Liu, 1994; Liu and Patton, 1998b).

Case 2: $|\lambda_i I - A| = 0$

For the case of $|\lambda_i I - A| = 0$, the calculation of allowable right eigenvector subspaces is now given. The left eigenvector subspaces may be found using a similar analysis. Starting with (8.3.6), and considering first only the assignment of real eigenvalues, an alternative rearrangement to that performed in the above analysis produces

$$[\, A - \lambda_i I \quad B \,] \begin{bmatrix} R_i \\ W_i \end{bmatrix} = 0 \tag{8.3.23}$$

which is equivalent to

$$\begin{bmatrix} R_i \\ W_i \end{bmatrix} \in \{ S_i : [\, A - \lambda_i I \quad B \,] S_i = 0 \} \tag{8.3.24}$$

Equation (8.3.24) is interpreted as meaning that the vector $[R_i^T, W_i^T]^T$ belongs to the null space S_i (or kernel) of the $n \times (n+m)$ matrix $[\, A - \lambda_i I \quad B \,]$.

If complex eigenvalues are to be assigned then, in order to avoid the use of complex arithmetic, proceed as follows. Equation (8.3.16) is rewritten as one equation:

$$\begin{bmatrix} \lambda_i^{re} I - A & -\lambda_i^{im} I & B & 0 \\ \lambda_i^{im} I & \lambda_i^{re} I - A & 0 & B \end{bmatrix} \begin{bmatrix} R_i^{re} \\ R_i^{im} \\ W_i^{re} \\ W_i^{im} \end{bmatrix} = 0 \tag{8.3.25}$$

Equation (8.3.25) describes another null space problem.

Roughly speaking, the null space S_i forms an allowable eigenvector subspace. This space may be calculated by performing the singular value decomposition or orthogonal triangular decomposition method.

8.4 Parametric Eigenstructure Assignment

In this section, the (ij)-th Jordan block is taken to show how to obtain the parametric expression of the eigenvector and generalised eigenvectors. The other Jordan blocks can also follow the same procedure developed below.

From (8.2.11) and (8.2.15), the right and left eigenvectors and generalised eigenvectors corresponding to the real eigenvalue λ_i in the (ij)-th Jordan block can also be written as

$$(\lambda_i I - A) R_{ij,k} = BKCR_{ij,k} - R_{ij,k-1} \tag{8.4.1}$$

$$L_{ij,k}^T (\lambda_i I - A) = L_{ij,k}^T BKC - L_{ij,k-1}^T \tag{8.4.2}$$

$R_{ij,0} = 0$, $L_{ij,0} = 0$, for $k = 1, 2, ..., p_{ij}$. Here introduce the right and left auxiliary vectors defined by, respectively

$$W_{ij,k} = KCR_{ij,k} \in \mathcal{R}^{r \times 1} \tag{8.4.3}$$

$$V_{ij,k}^T = L_{ij,k}^T BK \in \mathcal{R}^{1 \times m} \tag{8.4.4}$$

for $k = 1, 2, ..., p_{ij}$. Substituting the right and left auxiliary vectors into equations (8.4.1) and (8.4.2), respectively gives

$$(\lambda_i I - A) R_{ij,k} + R_{ij,k-1} = BW_{ij,k} \tag{8.4.5}$$

$$L_{ij,k}^T (\lambda_i I - A) + L_{ij,k-1}^T = V_{ij,k}^T C \tag{8.4.6}$$

for $k = 1, 2, ..., p_{ij}$. The relationship between the right eigenvector and generalised eigenvectors and the right auxiliary vectors can also compactly be written in the following matrix form:

$$A_j^R(\lambda_i) \bar{R}_{ij} = B_{ij} \bar{W}_{ij} \tag{8.4.7}$$

where

$$A_j^R(\lambda_i) = \begin{bmatrix} \lambda_i I - A & & & \\ I & \lambda_i I - A & & \\ & \ddots & \ddots & \\ & & I & \lambda_i I - A \end{bmatrix} \tag{8.4.8}$$

$$B_{ij} = \begin{bmatrix} B & & & \\ & B & & \\ & & \ddots & \\ & & & B \end{bmatrix} \tag{8.4.9}$$

$$\bar{R}_{ij} = \begin{bmatrix} R_{ij,1} \\ R_{ij,2} \\ \vdots \\ R_{ij,p_{ij}} \end{bmatrix} \tag{8.4.10}$$

$$\bar{W}_{ij} = \begin{bmatrix} W_{ij,1} \\ W_{ij,2} \\ \vdots \\ W_{ij,p_{ij}} \end{bmatrix} \tag{8.4.11}$$

Equation (8.4.7) provides a good formulation for the parametric expression of the eigenvector and generalised eigenvectors. Similarly, the relationship between the left eigenvector and generalised eigenvectors and the left auxiliary vectors is of the following compact matrix form:

$$A_j^L(\lambda_i)\bar{L}_{ij} = C_{ij}\bar{V}_{ij} \tag{8.4.12}$$

where

$$A_j^L(\lambda_i) = \begin{bmatrix} \lambda_i I - A^T & & & \\ I & \lambda_i I - A^T & & \\ & \ddots & \ddots & \\ & & I & \lambda_i I - A^T \end{bmatrix} \tag{8.4.13}$$

$$C_{ij} = \begin{bmatrix} C^T & & & \\ & C^T & & \\ & & \ddots & \\ & & & C^T \end{bmatrix} \tag{8.4.14}$$

$$\bar{L}_{ij} = \begin{bmatrix} L_{ij,1} \\ L_{ij,2} \\ \vdots \\ L_{ij,p_{ij}} \end{bmatrix} \tag{8.4.15}$$

$$\bar{V}_{ij} = \begin{bmatrix} V_{ij,1} \\ V_{ij,2} \\ \vdots \\ V_{ij,p_{ij}} \end{bmatrix} \tag{8.4.16}$$

The matrices $A_j^R(\lambda_i)$ and $A_j^L(\lambda_i)$ are singular if the closed- and open-loop eigenvalue sets intersect. Thus, given the vectors $\bar{W}_{ij}$ and $\bar{V}_{ij}$, the vectors $\bar{R}_{ij}$ and $\bar{L}_{ij}$ cannot simply be calculated by premultiplying the matrices $(A_j^R(\lambda_i))^{-1}$ in (8.4.7) and $(A_j^L(\lambda_i))^{-1}$ in (8.4.12), respectively. In order to develop a generic parametric expression, we firstly rewrite equations (8.4.7) and (8.4.12) as

$$[\, A_j^R(\lambda_i) \quad -B_{ij} \,] \begin{bmatrix} \bar{R}_{ij} \\ \bar{W}_{ij} \end{bmatrix} = 0 \tag{8.4.17}$$

$$[\, A_j^L(\lambda_i) \quad -C_{ij} \,] \begin{bmatrix} \bar{L}_{ij} \\ \bar{V}_{ij} \end{bmatrix} = 0 \tag{8.4.18}$$

To find a solution, define

$$\Phi_{ij} = [\, A_j^R(\lambda_i) \quad -B_{ij} \,] \tag{8.4.19}$$

$$\Psi_{ij} = [\, A_j^L(\lambda_i) \quad -C_{ij} \,] \tag{8.4.20}$$

and two compatibly-dimensioned matrices

$$P_{ij} = \begin{bmatrix} P_{ij,1} \\ P_{ij,2} \end{bmatrix} \tag{8.4.21}$$

$$Q_{ij} = \begin{bmatrix} Q_{ij,1} \\ Q_{ij,2} \end{bmatrix} \tag{8.4.22}$$

so that the columns of the matrix P_{ij} form the basis for the nullspace of Φ_{ij} and the columns of the matrix Q_{ij} form the basis for the nullspace of Ψ_{ij}. Thus

$$[\, A_j^R(\lambda_i) \quad -B_{ij} \,] \begin{bmatrix} P_{ij,1} \\ P_{ij,2} \end{bmatrix} D_{ij} = 0 \tag{8.4.23}$$

$$[\, A_j^L(\lambda_i) \quad -C_{ij} \,] \begin{bmatrix} Q_{ij,1} \\ Q_{ij,2} \end{bmatrix} E_{ij} = 0 \tag{8.4.24}$$

where D_{ij} and E_{ij} are called the right and left parameter vectors, respectively. Comparing (8.4.17) and (8.4.23) gives

$$\bar{R}_{ij} = P_{ij,1} D_{ij} \tag{8.4.25}$$

$$\bar{W}_{ij} = P_{ij,2} D_{ij} \tag{8.4.26}$$

The parametric expressions of the eigenvector and generalised eigenvectors and the right auxiliary vectors can be given by, respectively,

$$R_{ij,k} = P_{ij,1}^{(k)} D_{ij} \tag{8.4.27}$$

$$W_{ij,k} = P_{ij,2}^{(k)} D_{ij} \tag{8.4.28}$$

where

$$\begin{bmatrix} P_{ij,1}^{(1)} \\ P_{ij,1}^{(2)} \\ \vdots \\ P_{ij,1}^{(p_{ij})} \end{bmatrix} = P_{ij,1}, \quad \begin{bmatrix} P_{ij,2}^{(1)} \\ P_{ij,2}^{(2)} \\ \vdots \\ P_{ij,2}^{(p_{ij})} \end{bmatrix} = P_{ij,2} \tag{8.4.29}$$

Clearly, $R_{ij,k}$ and $W_{ij,k}$ are the linear functions of the right parameter vector D_{ij}. Hence the right eigenvector and generalised eigenvector matrix and the right auxiliary-vector matrix for (ij)-th Jordan block are

$$R_{ij} = \begin{bmatrix} P_{ij,1}^{(1)} D_{ij} & P_{ij,1}^{(2)} D_{ij} & \dots & P_{ij,1}^{(p_{ij})} D_{ij} \end{bmatrix} \tag{8.4.30}$$

$$W_{ij} = \begin{bmatrix} P_{ij,2}^{(1)} D_{ij} & P_{ij,2}^{(2)} D_{ij} & \dots & P_{ij,2}^{(p_{ij})} D_{ij} \end{bmatrix} \tag{8.4.31}$$

As can be seen from the above, the matrices R_{ij} and W_{ij} are determined by the right parameter vector D_{ij} which can arbitrarily be chosen to ensure the vectors $R_{ij,k}$ are independent.

Next, let us consider the parametric expression of the left eigenvectors and generalised eigenvectors. Similarly, comparing (8.4.18) and (8.4.24) gives

$$\bar{L}_{ij} = Q_{ij,1} E_{ij} \tag{8.4.32}$$

$$\bar{V}_{ij} = Q_{ij,2} E_{ij} \tag{8.4.33}$$

The left eigenvector and generalised eigenvectors and the left auxiliary vectors can be given by, respectively,

$$L_{ij,k} = Q_{ij,1}^{(k)} E_{ij} \tag{8.4.34}$$

$$V_{ij,k} = Q_{ij,2}^{(k)} E_{ij} \tag{8.4.35}$$

where

$$\begin{bmatrix} Q_{ij,1}^{(1)} \\ Q_{ij,1}^{(2)} \\ \vdots \\ Q_{ij,1}^{(p_{ij})} \end{bmatrix} = Q_{ij,1}, \quad \begin{bmatrix} Q_{ij,2}^{(1)} \\ Q_{ij,2}^{(2)} \\ \vdots \\ Q_{ij,2}^{(p_{ij})} \end{bmatrix} = Q_{ij,2} \tag{8.4.36}$$

It is clear that $L_{ij,k}$ and $V_{ij,k}$ are the linear functions of the left parameter vector E_{ij}. So, the left eigenvector and generalised eigenvector matrix and the left auxiliary-vector matrix for (ij)-th Jordan block are

$$L_{ij} = \begin{bmatrix} Q_{ij,1}^{(1)} E_{ij} & Q_{ij,1}^{(2)} E_{ij} & \dots & Q_{ij,1}^{(p_{ij})} E_{ij} \end{bmatrix} \tag{8.4.37}$$

$$V_{ij} = \begin{bmatrix} Q_{ij,2}^{(1)} E_{ij} & Q_{ij,2}^{(2)} E_{ij} & \dots & Q_{ij,2}^{(p_{ij})} E_{ij} \end{bmatrix} \tag{8.4.38}$$

The above shows that the matrices L_{ij} and V_{ij} are determined by the left parameter vector E_{ij} which can arbitrarily be chosen to ensure the vectors $L_{ij,k}$ are independent.

It is known that there exists a real matrix K such that all closed-loop eigenvalues are in the set Λ if and only if the right and left eigenvectors and generalised eigenvectors satisfy the following constraints: a) $R_{ij,k} = R^*_{lj,k}$, $L_{ij,k} = L^*_{lj,k}$, for $\lambda_i = \lambda^*_l$, $j = 1, 2, ..., r_i$, $k = 1, 2, ..., p_{ij}$, b) $L^T R = I$. The constraint a) guarantees the controller K is a real matrix. If constraints a) and b) are satisfied, the controller K is given by either (Kwon and Youn, 1987; Liu and Patton, 1998a)

$$K = W(CR)^T(CR(CR)^T)^{-1} \tag{8.4.39}$$

or

$$K^T = V(B^T L)^T(B^T L(B^T L)^T)^{-1} \tag{8.4.40}$$

It has been shown that the right and left eigenvectors and generalised eigenvectors are determined by the non-null right and left parameter vectors D_{ij} and E_{ij}, for $j = 1, 2, ..., r_i$ and $i = 1, 2, ..., \tilde{n}$. Thus, using equations (8.4.30) and (8.4.37), the constraint b) can be written by

$$E^T_{xy}(Q^{(z)}_{xy,1})^T P^{(k)}_{ij,1} D_{ij} = \begin{cases} 1, & \text{if } (x, y, z) = (i, j, k) \\ 0, & \text{otherwise} \end{cases} \tag{8.4.41}$$

for $k = 1, 2, ..., p_{ij}$, $j = 1, 2, ..., r_i$, $z = 1, 2, ..., p_{xy}$, $y = 1, 2, ..., r_x$, $i, x = 1, 2, ..., \tilde{n}$.

Since the output feedback eigenstructure assignment usually provides some freedom beyond satisfying (8.4.41), next we discuss how to make full use of the freedom to obtain an optimal solution. Some performance functions which measure sensitivity of the closed-loop matrix and robustness performance of the closed-loop systems are introduced.

8.5 Multiobjective Eigenstructure Assignment

In most parameter insensitive design methods using eigenstructure assignment the performance indices are given on the basis of the right eigenvector matrix. For example, a very common performance index is the overall eigenvalue sensitivity:

$$\phi(R) = \| R \|_2 \| L \|_2 \tag{8.5.1}$$

which gives an overall measure of conditioning of the eigenproblem.

It has been shown that the individual eigenvalue sensitivity ϕ_i of a matrix $A + BK$ to perturbations for the i-th eigenvalues λ_i is

$$\phi_i(R, L) = \frac{\| L_i \|_2 \| R_i \|_2}{|L_i^T R_i|} \tag{8.5.2}$$

where L_i and R_i are the i-th left and right eigenvectors of $A + BKC$.

Though it has been shown that a minimisation of $\phi(R)$ reduces a bound on the individual eigenvalue sensitivities and the actual values of $\phi_i(R,L)$ themselves will become small so that the conditioning of the eigenproblem is improved, it is often conservative because the $\phi(R)$ measures the upper bound of all individual eigenvalue sensitivities, *i.e.*,

$$\phi(R) \geq \max_{i=1,2,\ldots,n} \phi_i(R,L) \tag{8.5.3}$$

Hence, in order to reduce the conservatism we consider the following formulation:

$$\phi_i(R) \leq \varepsilon_i, \quad \text{for} \quad i = 1,2,\ldots,n \tag{8.5.4}$$

where ε_i is a real positive number. It is clear that if we appropriately choose

$$\varepsilon_i \in [\min_{R,L} \phi_i(R,L), \min_{R} \phi(R)], \quad \text{for} \quad i = 1,2,\ldots,n \tag{8.5.5}$$

then the conservatism can be significantly reduced.

There are also a number of robust performance indices which are considered in optimisation design of control systems in the frequency domain. First, let us define the following norms:

$$\| H \|_{\infty} = \sup\{|H(j\omega)| : \quad \omega \in \mathcal{R}\} \tag{8.5.6}$$

$$\| H \|_{\infty,\mathcal{F}} = \sup\{|H(j\omega)| : \quad \omega \in \mathcal{F}\} \tag{8.5.7}$$

where H is a transfer function and $\mathcal{F}$ denotes a frequency range.

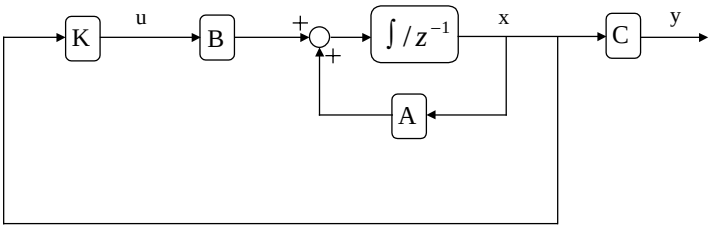

Figure 8.2: State feedback control system

In robust control using H^{∞} for state feedback systems, as shown in Figure 8.2, the objectives are expressed in terms of the H^{∞}-norm of transfer functions. One of the objectives of the system with $C = I$ is the following:

$$\min_{K} \left\| \begin{matrix} S \\ C_S \end{matrix} \right\|_{\infty} \tag{8.5.8}$$

where

$$S = (I - (sI - A)^{-1}BK)^{-1} \tag{8.5.9}$$

$$C_S = K((sI - A)^{-1}BK - I)^{-1} \tag{8.5.10}$$

S is the sensitivity function, C_S is the complementary sensitivity function, and the minimisation is over the whole set of stabilising controllers K.

As the singular value techniques are used to evaluate control system stability and robustness characteristics it becomes apparent that a singular value matrix norm is often conservative in its ability to predict near instability (Safonov *et al.*, 1981). Though the robust performance formulation (8.5.8) is widely used in H^∞ control, it is often conservative because of the following relations:

$$\max\{\| S \|_\infty, \| C_S \|_\infty\} \leq \left\| \begin{array}{c} S \\ C_S \end{array} \right\|_\infty \tag{8.5.11}$$

This means that the maximum singular value of the matrix $[S^T, (C_S)^T]^T$ gives a measurement of the upper bound of the maximum singular values of the sensitivity function S and the function C_S. Thus, in order to reduce the conservatism above, let us consider the following robust control formulation:

$$\| S \|_\infty \leq \varepsilon_1 \tag{8.5.12}$$

$$\| C_S \|_\infty \leq \varepsilon_2 \tag{8.5.13}$$

where ε_i is a positive real number. If we let

$$\varepsilon_i = \min_K \left\| \begin{array}{c} S \\ C_S \end{array} \right\|_\infty \tag{8.5.14}$$

for $i = 1, 2$, the robust control problem above covers a common H^∞ control problem described by (8.5.8). *i.e.*, the H^∞-norm of the transfer function $\left\| \begin{array}{c} S \\ C_S \end{array} \right\|_\infty$ is minimised by choosing an appropriate stabilising controller K. If we let

$$\varepsilon_1 \in \left[\min_K \| S \|_\infty, \min_K \left\| \begin{array}{c} S \\ C_S \end{array} \right\|_\infty \right] \tag{8.5.15}$$

$$\varepsilon_2 \in \left[\min_K \| C_S \|_\infty, \min_K \left\| \begin{array}{c} S \\ C_S \end{array} \right\|_\infty \right] \tag{8.5.16}$$

it is clear that the robust control problem described by (8.5.15) and (8.5.16) is less conservative than that described by (8.5.8). According to practical requirements ε_i can be appropriately adjusted so that the conservatism of (8.5.8) is probably reduced (Liu and Patton, 1996a).

Nowadays, weighting matrices $\tilde{W}_0$, $\tilde{W}_1$ and $\tilde{W}_2$ are introduced to give

$$\min_K \left\| \begin{array}{c} \tilde{W}_1 S \tilde{W}_0 \\ \tilde{W}_2 C_S \tilde{W}_0 \end{array} \right\|_\infty \tag{8.5.17}$$

The weights are selected so that $\tilde{W}_1 S \tilde{W}_0$ dominates the cost at low frequencies, and $\tilde{W}_2 C_S \tilde{W}_0$ dominates the cost at high frequencies. The aims of the

optimisation can be interpreted as improvement in performance, *i.e.*, disturbance rejection at low frequencies and enhancement in robust stabilities to safeguard against modelling errors at high frequencies.

If the low frequency range and the high frequency range are denoted by $\mathcal{F}_L$ and $\mathcal{F}_H$, respectively, we similarly have the following robust control formulation:

$$\| S \|_{\infty,\mathcal{F}_L} \leq \varepsilon_1 \tag{8.5.18}$$

$$\| C_S \|_{\infty,\mathcal{F}_H} \leq \varepsilon_2 \tag{8.5.19}$$

Thus, the multiobjective performance functions for multivariable control systems may be

$$\phi_i(K) = \phi_i(R, L), \quad \text{for} \quad i = 1, 2, \ldots, n \tag{8.5.20}$$

$$\phi_{n+1}(K) = \| S \|_{\infty,\mathcal{F}_L} \tag{8.5.21}$$

$$\phi_{n+2}(K) = \| C_S \|_{\infty,\mathcal{F}_H} \tag{8.5.22}$$

Furthermore, we often need to consider some constraints on the controller gain matrix K. A scalar measure which quantifies a structurally constrained gain matrix may be defined as follows:

$$\phi_{n+3}(K) = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \beta_{ij} K_{ij}^2 \right)^{1/2} \tag{8.5.23}$$

where K_{ij} is the (ij)th element of the full state gain matrix K and β_{ij} is a positive weighting factor which may be used to force certain elements of the gain matrix to become small. For the output feedback case, some elements of the gain matrix K are zero-valued and the corresponding weighting parameters β_{ij} can be selected to have large values to force this structure (Burrows and Patton, 1991). There is then no loss of generality in using this formulation for either state or output feedback control problems.

In practice, it is usually intended to locate the eigenvalue vector λ in a well-defined set to meet the requirements of the practical control system (*e.g.*, stability, speed of response, etc.). This leads to eigenvalue constraints, for example of the form $\underline{\lambda}_i \leq \lambda_i \leq \bar{\lambda}_i$, where $\underline{\lambda}_i \in \mathcal{R}$ and $\bar{\lambda}_i \in \mathcal{R}$ are the lower and the upper bound vectors, respectively. These constraints may be removed by considering the change of variables given by

$$\lambda_i(z_i) = \underline{\lambda}_i + (\bar{\lambda}_i - \underline{\lambda}_i) \sin^2(z_i) \tag{8.5.24}$$

with $z_i \in \mathcal{R}$.

Since it has been shown that the right and left eigenvector matrices are determined by closed-loop eigenvalues and two parameter matrices (D, E), clearly the controller matrix K is a function of $Z = [z_1, z_2, ..., z_n]$, D and E.

Thus, the performance functions $\phi_i(K)$ $(i = 1, 2, ..., n+3)$ can be described by $\phi_i(Z, D, E)$.

If one of the performance functions $\phi_i(Z, D, E)$ $(i = 1, 2, ..., n+3)$ is minimised individually (single-objective approach), then unacceptably large values may result for other performance functions $\phi_j(Z, D, E)$ $(j \neq i,\ j = 1, 2, ..., n+3)$. The single (or mixed) objective approach has been considered in Roppenecker (1983) and Burrows and Patton (1991). Generally, there does not exist a solution for all performance functions $\phi_i(Z, D, E)$, for $i = 1, 2, ..., n+3$ to be minimised by the same controller K.

We can therefore reformulate the optimisation into a multiobjective problem as

$$\begin{cases} \min\limits_{Z,D,E} \sum\limits_{i=1}^{n+3} \phi_i(Z, D, E)/\varepsilon_i \\ \phi_i(Z, D, E) \leq \varepsilon_i, \quad \text{for} \quad i = 1, 2, \ldots, n+3 \end{cases} \tag{8.5.25}$$

where the positive real number ε_i represents the numerical bound on the performance function $\phi_i(Z, D, E)$ and is determined by the designer.

8.6 Controller Design Using the Method of Inequalities

In order that the closed-loop system satisfy a set of required performance criteria with less conservatism, eigenstructure assignment techniques and the method of inequalities are applied to the multiobjective optimisation control problem (8.5.25).

Following the method of inequalities (Zakian and Al-Naib, 1973; Maciejowski, 1989; Whidborne and Liu, 1993), a numerical solution to the multiobjective optimisation control problem (8.5.25) is discussed below.

Let Θ_i be the set of parameters (Z, D, E) for which the ith performance criterion is satisfied:

$$\Theta_i = \{(Z, D, E) : \phi_i(Z, D, E) \leq \varepsilon_i\} \tag{8.6.1}$$

Then the admissible or feasible set of parameters for which all the performance criteria hold is the intersection

$$\Theta = \bigcap_{i=1}^{n+3} \Theta_i \tag{8.6.2}$$

Clearly, (Z, D, E) are admissible parameters if and only if

$$\max_{i=1,2,...,n+3} \{\phi_i(Z, D, E)/\varepsilon_i\} \leq 1 \tag{8.6.3}$$

which shows that the search for an admissible pair (Z, D, E) can be pursued by optimisation, in particular by solving

$$\min_{Z,D,E}\{\max_{i=1,2,\ldots,n+3}\{\phi_i(Z,D,E)/\varepsilon_i\}\} \leq 1 \tag{8.6.4}$$

Now, let (Z^k, D^k, E^k) be the values of the parameters at the kth step, and define

$$\Theta_i^k = \{(Z,D,E) : \phi_i(Z,D,E)/\varepsilon_i \leq \Delta^k\}, \text{ for } i = 1,2,\ldots,n+3 \tag{8.6.5}$$

where

$$\Delta^k = \max_{1,2,\ldots,n+3}\{\phi_i(Z^k, D^k, E^k)/\varepsilon_i\} \tag{8.6.6}$$

and also define

$$\Theta^k = \bigcap_{i=1}^{n+3} \Theta_i^k \tag{8.6.7}$$

$$\Phi^k = \sum_{i=1}^{n+3} \phi_i(Z^k, D^k, E^k)/\varepsilon_i \tag{8.6.8}$$

Θ^k is the kth set of parameters for which all performance functions satisfy

$$\phi_i(Z,D,E)/\varepsilon_i \leq \Delta^k, \quad \text{for} \quad i = 1,2,\ldots,n+3 \tag{8.6.9}$$

It is clear that Θ^k contains both (Z^k, D^k, E^k) and the admissible set Θ. Φ^k is a combined measurement of all performance functions. If we find new parameters $(\bar{Z}^k, \bar{D}^k, \bar{E}^k)$, such that

$$\bar{\Delta}^k < \Delta^k \tag{8.6.10}$$

or

$$\bar{\Delta}^k = \Delta^k \quad \text{and} \quad \bar{\Phi}^k < \Phi^k \tag{8.6.11}$$

where $\bar{\Delta}^k$ and $\bar{\Phi}^k$ are defined similarly to Δ^k and Φ^k, then we accept $(\bar{Z}^k, \bar{D}^k, \bar{E}^k)$ as the next value of the parameters. One of the methods to find $(\bar{Z}^k, \bar{D}^k, \bar{E}^k)$ is the moving boundary process (MBP) (Zakian and Al-Naib, 1973), which uses an optimisation algorithm introduced in Rosenbrock (1960). Then, we set $(Z^{k+1}, D^{k+1}, E^{k+1}) = (\bar{Z}^k, \bar{D}^k, \bar{E}^k)$. We have

$$\phi_i(Z^{k+1}, D^{k+1}, E^{k+1}) \leq \phi_i(\bar{Z}^k, \bar{D}^k, \bar{E}^k), \text{ for } i = 1,2,\ldots,n+3 \tag{8.6.12}$$

and

$$\Theta \subset \Theta^{k+1} \subset \Theta^k \tag{8.6.13}$$

So, the boundary of the set in which the parameters are located has been moved towards the admissible set, or, rarely, has remained unaltered. The process of finding the optimisation solution is terminated when both Δ_k and Φ^k cannot be reduced any further. But the process of finding an admissible parameter pair (Z, D, E) is terminated when

$$\Delta^k \leq 1 \tag{8.6.14}$$

i.e., when the boundaries of Θ^k have converged to the boundaries of Θ. If the Δ^k persists in being larger than 1, this may be taken as an indication that the performance criteria may be inconsistent, whilst their magnitude gives some measure of how closely it is possible to approach the objectives. In this case, some of the performance criteria should be relaxed until they are satisfied. From a practical viewpoint, the approximate optimal solution is also useful if the optimal solution is not achievable.

Example 8.1 Consider the linear equation of motion of lateral dynamics of the L-1011 aircraft corresponding to a certain cruise flight condition (Andry *et al.*, 1983). The control system for this example is described by

$$\begin{aligned} \dot{x} &= Ax + Bu \\ u &= Kx \end{aligned}$$

where

$$A = \begin{bmatrix} 0 & 1.0000 & 0 & 0 \\ 0 & -1.8900 & 0.3900 & -5.5550 \\ 0 & -0.0340 & -2.9800 & 2.4300 \\ 0.0350 & -0.0011 & -0.9900 & -0.2100 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0.760 & -1.600 \\ -0.950 & -0.032 \\ 0.030 & 0 \end{bmatrix}$$

The closed-loop eigenvalues are required to lie in ranges given by

$$\begin{aligned} \lambda_1 &\in [-1.5, -0.5] \\ \lambda_2 &\in [-2.5, -1.5] \\ \lambda_3 &\in [-3.5, -2.5] \\ \lambda_4 &\in [-4.5, -3.5] \end{aligned}$$

A state feedback controller K is required to assign the eigenvalues in the above

ranges to satisfy the following performance criteria:

$$\begin{bmatrix} \phi_1(Z,D,E) \\ \phi_2(Z,D,E) \\ \phi_3(Z,D,E) \\ \phi_4(Z,D,E) \\ \phi_5(Z,D,E) \end{bmatrix} \leq \begin{bmatrix} 2.3 \\ 4.1 \\ 2.3 \\ 4.1 \\ 9.0 \end{bmatrix}$$

The performance function ϕ_i is the sensitivity of each eigenvalue λ_i, for $i = 1,2,3,4$, respectively. The performance function ϕ_5 is the gain matrix measure, which is defined as (8.5.23) with $\beta_{ij} = 1$, for $i = 1,2,3,4$ and $j = 1,2$. Using the method considered in this section, this has led to the following controller gain matrix, closed-loop eigenvalue set and the control performance:

$$K = \begin{bmatrix} 1.6569 & 0.6663 & 2.4237 & -3.5491 \\ 3.2205 & 1.8746 & 1.8304 & -6.1889 \end{bmatrix}$$

$$\Lambda = [-1.0184 \quad -2.0165 \quad -3.0148 \quad -3.9908]$$

$$\begin{bmatrix} \phi_1(Z,D,E) \\ \phi_2(Z,D,E) \\ \phi_3(Z,D,E) \\ \phi_4(Z,D,E) \\ \phi_5(Z,D,E) \end{bmatrix} = \begin{bmatrix} 2.2378 \\ 4.0807 \\ 2.2326 \\ 4.0893 \\ 8.7863 \end{bmatrix}$$

It has been found that the overall eigenvalue sensitivity $\phi(R) = 8.0840$. Clearly, it is much larger than the maximum (4.0893) of the individual eigenvalue sensitivities $\phi_i(Z,D,E)$ $(i = 1,2,3,4)$ which have met the design requirements. This indicates that the individual eigenvalue sensitivity functions are more accurate and better to describe the system insensitivity property to perturbations than the performance function $\phi(R)$. If the gain matrix elements were considered to be too large, a further reduction would be possible by sacrificing parameter insensitivity (and vice versa), *i.e.*, the current level of one or more individual eigenvalue sensitivities, by appropriate adjustments to ε_i $(i = 1,2,3,4)$. In this way, we can design the controller to meet the specific needs.

8.7 Controller Design Using Genetic Algorithms

As we are concerned with several types of objectives (or cost functions) for control systems, this section introduces another numerical algorithm for the multiobjective optimisation control problem (8.5.25) using genetic algorithms (Liu and Patton, 1996a), based on the formulation in the previous section. The steps to be executed for the GA implementation are as follows:

S1 Each solution in the population is represented as a real number string. As the eigenvalues $Z \in \mathcal{R}^{1\times n}$, the right and left parameter matrices $D = [D_1, D_2, ..., D_n]$ and $E = [E_1, E_2, ..., E_n]$ then the chromosomal representation may be expressed as an array

$$P = [Z, D_1^T, D_2^T, ..., D_n^T, E_1^T, E_2^T, ..., E_n^T] \tag{8.7.1}$$

S2 The N (an odd number) sets of parameter strings P for the initial population are randomly generated.

S3 Evaluate the performance functions $\phi_i(P_j)$ $(i = 1, 2, ..., n+3)$ for all N sets of the parameters P_j and

$$\Delta_j = \max_{i=1,2,...,n+3} \phi_i(P_j)/\varepsilon_i \tag{8.7.2}$$

$$\Phi_j = \sum_{i=1}^{n+3} \phi_i(P_j) \tag{8.7.3}$$

for $j = 1, 2, \ldots, N$.

S4 According to the fitness of the performance functions for each set of parameters, remove the $(N-1)/2$ weaker members of the population and reorder the sets of parameters. The fitness of the performance functions is measured by

$$F_j = \Delta_j^{-1}, \quad \text{for} \quad j = 1, 2, \ldots, N \tag{8.7.4}$$

S5 Perform the crossover using an average crossover function to produce the $(N-1)/2$ offspring. The average crossover operator takes two parents which are selected in S4 and produces one child that is the result of averaging the corresponding fields of two parents. Thus, the average crossover function is given by

$$P_{Cj} = (P_{j+1} + P_j)/2, \quad \text{for} \quad j = 1, 2, \ldots, (N-1)/2 \tag{8.7.5}$$

S6 A real number mutation operator is used. The maximum amount that this operator can alter the value of a field is a parameter of the operator. The mutation operation is defined as

$$P_{Mj} = P_{Cj} + d_m \xi_j, \quad \text{for} \quad j = 1, 2, \ldots, (N-1)/2 \tag{8.7.6}$$

where d_m is the maximum to be altered and $\xi_j \in [-1, 1]$ is a random variable with zero mean.

S7 To prevent the best parameter set from loss in the succeeding parameter sets, the elitist strategy is used to copy the best parameter set into the succeeding parameter sets. The best parameter set P_b is defined as one satisfying

$$\Phi_b = \min_l \{\Phi_l : \Phi_l \leq \Phi_m - \alpha(\Delta_l - \Delta_m) \text{ and } \Delta_l \leq \Delta_m + \delta\} \tag{8.7.7}$$

where

$$\Delta_m = \min_{j=1,2,\ldots,n+3}\{\Delta_j\} \tag{8.7.8}$$

E_m and E_l correspond to Δ_m and Δ_l, $\alpha > 1$ and δ is a positive number, which are given by the designer (*e.g.*, $\alpha = 1.1$ and $\delta = 0.1$).

S8 Insert the $(N-1)/2$ new offspring to the population which are generated in a random fashion. Actually, the new offspring are formed by mutating the best parameter set P_b with a probability, *i.e.*,

$$P_{Nj} = P_b + d_n\xi_j, \quad \text{for} \quad j = 1, 2, \ldots, (N-1)/2 \tag{8.7.9}$$

where d_n is the maximum to be altered and $\xi_j \in [-1, 1]$ is a random variable with zero mean. Thus, the next population is formed by the parameter sets P_{Mj} $(j = 1, 2, \ldots, (N-1)/2)$, P_{Nj} $(j = 1, 2, \ldots, (N-1)/2)$ and P_b.

S9 Continue the cycle initiated in S3 until convergence is achieved. The population is considered to have converged when Φ_b cannot be reduced any further, subject to

$$\Delta_j - \Delta_b \leq \varepsilon, \quad \text{for} \quad j = 1, 2, \ldots, N \tag{8.7.10}$$

where ε is a positive number.

Take the best solution in the converged generation and place it in a second 'initial generation'. Generate the other $N-1$ parameter sets in this second initial generation at random and begin the cycle again until a satisfactory solution is obtained or Δ_b and Φ_b cannot be improved any further.

Example 8.2 Consider a linear representation of a distillation column (Kautsky *et al.*, 1985) with a state feedback controller, which is of the form:

$$\dot{x} = Ax + Bu$$
$$u = Kx$$

where

$$A = \begin{bmatrix} -0.1094 & 0.0628 & 0 & 0 & 0 \\ 1.3060 & -2.1320 & 0.9807 & 0 & 0 \\ 0 & 1.5950 & -3.1490 & 1.5470 & 0 \\ 0 & 0.0355 & 2.6320 & -4.2570 & 1.8550 \\ 0 & 0.0023 & 0 & 0.1636 & -0.1625 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0.0638 & 0.0838 & 0.1004 & 0.0063 \\ 0 & 0 & -0.1396 & -0.2060 & -0.0128 \end{bmatrix}^T$$

The closed-loop eigenvalues are chosen such that

$$\begin{aligned} -10 \le \lambda_1 \le -0.01 \\ -10 \le \lambda_{2,re} \le -0.01 \\ -10 \le \lambda_{2,im} \le -0.01 \\ -10 \le \lambda_i \le -0.01 \qquad (i = 4, 5) \end{aligned}$$

Note that $\lambda_3 = \lambda_2^*$. A state feedback controller is required to assign the eigenvalues in the above regions to satisfy the following performance criteria:

$$\begin{aligned} &\phi_i(Z, D, E) \le 3.0, \quad \text{for} \quad i = 1, 2, 3, 4, 5 \\ &\phi_6(Z, D, E) \le 8.5 \\ &\phi_7(Z, D, E) \le 20.0 \end{aligned}$$

The performance functions ϕ_i (i = 1, 2, ..., 5) are the individual eigenvalue sensitivity functions. The performance functions ϕ_6 and ϕ_7 are the sensitivity function S and the function C_S of the closed-loop system in the frequency domain. The parameters for the numerical algorithm are

parameter length	15
population size N	21
d_m (mutation)	0.1
d_n (new population)	0.2
α (elitism)	1.1
δ (elitism)	0.1
frequency range $\mathcal{F}$	$[0.01, 100]$

Using the eigenstructure assignment toolbox (Liu and Patton, 1999), after 150 generations the optimal results have been found approximately by the numerical algorithm. The performance functions are

$$\begin{aligned} \phi_1(Z, D, E) &= 1.4968 \\ \phi_2(Z, D, E) &= 1.1875 \\ \phi_3(Z, D, E) &= 1.1875 \\ \phi_4(Z, D, E) &= 1.5144 \\ \phi_5(Z, D, E) &= 1.0757 \\ \phi_6(Z, D, E) &= 1.3056 \\ \phi_7(Z, D, E) &= 12.7281 \end{aligned}$$

and the eigenvalue set Λ of the closed-loop system is

$$\Lambda = [-0.1676 \quad -2.6299 - 0.9682j \quad -2.6299 + 0.9682j \quad -0.0950 \quad -5.9625]$$

The optimal state feedback controller is

$$K = \begin{bmatrix} -6.7708 & -4.7671 & 1.2392 & 2.1961 & 2.8328 \\ -2.8498 & 8.6277 & 7.2037 & 3.1976 & 3.8011 \end{bmatrix}$$

8.8 Summary

The robust control design using eigenstructure assignment and multiobjective optimisation has been presented for multivariable systems. The individual eigenvalue sensitivities, and the sensitivity functions of the closed-loop system in the frequency domain, and controller gain constraint have all been considered as the multiobjective performance functions for the robust control design. The multiobjective performance criteria are expressed by a set of inequalities. The criteria describe the practical control problem more accurately and less conservatively than would be possible using scalar performance criterion approaches. The multiobjective optimisation problem was also solved numerically using the method of inequalities and genetic algorithms.

Chapter 9

Multiobjective PI Controller Design for a Gasifier

9.1 Introduction

To provide environmentally clean and efficient power generation from coal, a gasifier plays an important role in integrated gasification combined cycle (IGCC) power plants. The gasifier is based on the spouted fluidised bed gasification concept (originally developed by the Coal Technology Development Division (CTDD) of the British Coal Corporation) and can be considered as a reactor where coal is gasified with air and steam. In the ALSTOM benchmark challenge on gasifier control (Dixon *et al.*, 2001), the gasifier is a nonlinear, multivariable system, having five inputs (coal, limestone, air, steam and char extraction) and four outputs (pressure, temperature, bed mass and gas quality) with a high degree of cross coupling between them (Donne *et al.*, 1998). The aim of the benchmark challenge is to design a controller to satisfy a set of specifications on the system outputs, inputs and input rates when a step pressure disturbance is applied to the system, based on a linearised model of the gasifier at the 100% load operating point. In addition, the challenge requires that the controller also be evaluated on models representing the gasifier at 50% and 0% load operating points and on rejection to a sine- and step-wave pressure disturbances applied to the gasifier at the three operating points.

There are a wide variety of control techniques which can be used to design a controller for the gasifier. It is well known that the PID controller is the most popular approach for industrial process control and many design techniques have been developed (see, for example, Ziegler and Nichols, 1942; Astrom and Hagglund, 1984; Hang *et al.* 1991; Zhuang and Atherton, 1993;

McCormack and Godfrey, 1998; Liu and Daley, 1999, 2000, 2001; Daley and Liu, 1999). This chapter introduces a PI controller design approach using multiobjective optimisation which addresses the ALSTOM benchmark challenge on gasifier control. The nonlinear gasifier has been linearised about three operating points. The design aim is to satisfy a set of specifications on the system outputs, inputs and input rates when a step disturbance is applied to the system. The parameters of the multi-input multi-output (MIMO) PI controller are selected to satisfy a set of multiobjective performance criteria which are formulated from the system specifications. Simulation results are presented which demonstrate the performance of the controller at the three operating points.

9.2 Modelling of the Gasifier

Integrated gasification combined cycle power plants are being developed around the world to provide environmentally clean and efficient power generation from coal. A detailed feasibility study on the development of a small scale Prototype Integrated Plant (PIP) has undertaken, based on the Air Blown Gasification Cycle (ABGC). In pursuit of this goal, a comprehensive dynamic model and control philosophy for the PIP has produced (Donne *et al.*, 1998). The gasifier is one component of the model which, being a highly coupled multivariable system with five inputs (coal, limestone, air, steam and char extraction) and four outputs (pressure, temperature, bed mass and gas quality), has been found to be particularly difficult to control. For this reason the gasifier together with its associated control specification, constraints, nonlinearity (with operating point) and various disturbance characteristics, has been selected for control system design. The motivation for gasifier control is that the gasifier is a real industrial problem facing the providers of power plant and poses a serious challenge even for advanced control techniques.

The programme undertaken by the UK's Clean Coal Power Generation Group (CCPGG) addressed development of the key components of an 87 MW Prototype Integrated Plant (PIP) based upon the ABGC shown in Figure 9.1. This programme also involved undertaking technical and economic assessments as required prior to the detailed design and specification of a fully integrated commercial-scale plant.

One aspect in the CCPGG programme was the development of a dynamic simulation model and a control philosophy for the PIP, more details of which may be found in Donne *et al.* (1998). In fulfilling this, the design and dynamic modelling requirements of specific components of the system, such as the gasifier, boost compressor, steam turbine and gas turbine were analysed. Furthermore, physical based models of all the gas and steam cycle components which compose the plant were developed, tested and validated as appropriate. These models were then integrated, together with the relevant control systems, to form the overall model of the PIP.

One of the reasons for modelling the PIP was to aid the development of a suitable control philosophy. Here, emphasis was placed on those aspects of the PIP that are not normally encountered in conventional power plants. The key components from this point of view are the gasifier, the gas turbine (running on low and variable calorific fuel-gas), the Circulating Fluidised Bed Combustor (CFBC) and the global plant control itself. There were two objectives for the control system analysis: primarily, to verify that the plant can be safely and adequately controlled; and secondly, to examine some of the more complex components of the PIP with a view to proposing safer, more economical, higher performance controllers using control techniques.

The control scheme developed was shown to be capable of controlling the complete plant even in the event of a full-load rejection, which represents the most severe "trip" (or fault) condition likely to be encountered. Also adequate control of load acceptance and load reduction was demonstrated with the demanded power being followed accurately in both cases.

The gasification plant for the PIP is based on the British Coal experimental gasifier, making use of the spouted fluidised bed gasification concept, and can be considered as a reactor where coal is gasified with air and steam. In simple terms the gasification process works as described below.

Pulverised coal mixed with limestone, which captures sulphur originating in the coal, is conveyed by pressurised air into the gasifier. The air and injected steam not only fluidise the solids in the gasifier, but also react with the carbon and volatiles from the coal, producing a low calorific value fuel gas (approximately 4.5 MJ/kg or 12% that of natural gas). The remaining char (ash from coal, limestone and unreacted carbon) is removed as bed material from the base of the gasifier or carried out of the top of the gasifier as fines with the product gas. Under certain circumstances, this elutriation can be as much as 70% of the total char off-take.

The gasifier model has been developed using the Advanced Continuous Simulation Language (ACSL), and is compatible with Framatome's Modular Modelling System (MMS). The different processes in the model include:

a) Drying process: the moisture in the coal and limestone is removed and added to the steam flow, dry limestone and ash are separated from the coal and fed to the desulphurisation process. The resulting dry ash-free coal is an input to the pyrolysis process.

b) Desulphurisation process: the sulphur in the ash is captured by the dry limestone resulting in a flow of inerts which is added to the bed mass.

c) Pyrolysis process: dry ash-free coal is devolatilised; the gases evolved and the resulting fixed carbon are available to the gasification process.

d) Gasification process: the chemical reactions modelled here are two exothermic combustion reactions and two endothermic gasification reactions, each with its own reaction rate. The unreacted carbon is added to the bed mass.

e) Mass Balance: a separate mass balance on the inerts and carbon to

obtain solids composition and hence the total mass of solids. The mass flow rate of elutriated char is calculated and added to the fuel gas.

At the global level, there is an overall heat balance for gases and solids to obtain the specific enthalpy of the gas and the total enthalpy of the solids. This model has been validated using measured time histories from the British Coal CTDD experimental test facility and it was shown that the model predicts the main trends in fuel gas quality.

9.3 System Specifications of the Gasifier

This section describes the control system requirements for the gasifier. A functional layout of the gasifier is shown in Figure 9.1. It is a nonlinear, multivariable component, having five inputs (coal, limestone, air, steam and char extraction) and four outputs (pressure, temperature, bed mass and gas quality) with a high degree of cross coupling between them. In addition, there is a disturbance input (PSINK) representing pressure disturbances induced as the gas turbine fuel inlet valve is opened and closed.

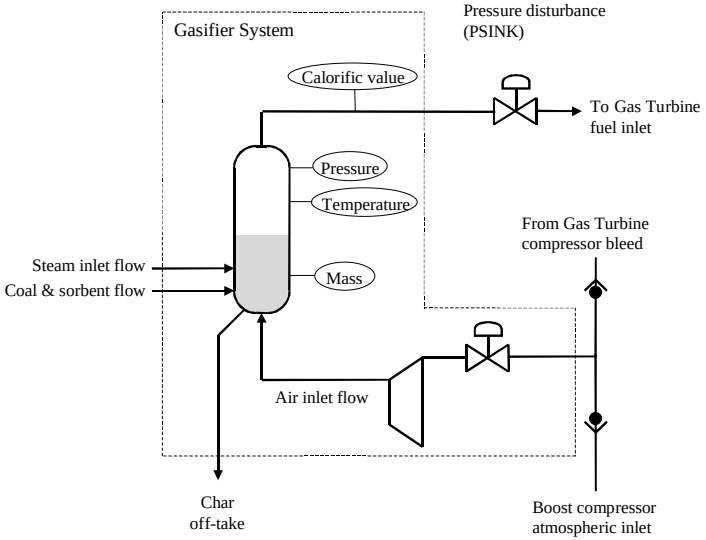

Figure 9.1: Gasifier plant functional diagram

The controllable inputs are:

a) Char extraction flow - WCHR (kg/s)
b) Air mass flow - WAIR (kg/s)
c) Coal flow - WCOL (kg/s)
d) Steam mass flow - WSTM (kg/s)

e) Limestone mass flow - WLS (kg/s-1)

The disturbance input is:

a) Sink pressure - PSINK (N/m^2)

The outputs are:

a) fuel gas calorific value - CVGAS (J/kg)
b) bed mass - MASS (kg)
c) fuel gas pressure - PGAS (N/m^2)
d) fuel gas temperature - TGAS (K)

Note that: Limestone absorbs sulphur in the coal so WLS should be set to a fixed ratio of WCOL, nominally 1:10 limestone to coal. This leaves effectively 4 degrees of freedom for the control design.

Three continuous time, state-space, linear models have been obtained from the nonlinear ACSL model at operating points of 100%, 50% and 0% load. These models were validated against the nonlinear model for a series of (10% of nominal) step inputs. For the validation, a PI controller manipulates the char off-take in order to maintain the bed mass - this is required because the bed mass level is marginally stable and over the run time (7200s or 2h) the nonlinear plant would have moved far from the operating point of the linearisation.

The controller should regulate the outputs, bearing in mind that there are the following input limits:

a) the input flow limits must not be exceeded;
b) the input rate of change limits must not be exceeded.

Table 9.1: Input limits

Input	Name	max (kg/s)	min (kg/s)	rate (kg/s^2)
Coal Inlet Flow	WCOL	10	0	0.2
Air Inlet flow	WAIR	20	0	1.0
Steam Inlet Flow	WSTM	6.0	0	1.0
Char Extraction	WCHR	3.5	0	0.2

The input limits are shown in Table 9.1. Note that the above input limits are absolute; *e.g.*, at 100% load the maximum increase in coal flow is $10 - 8.55 = 1.45$ kg/s.

There are also the following output limits:

a) the calorific value (CV) fluctuation should be minimised, but must always within 10KJ/kg.
b) the pressure fluctuation should be minimised, but must always be within 0.1bar.
c) bed mass should fluctuate by less than 5% of the nominal.

d) temperature fluctuation should be kept to a minimum, but always within 1^oC.

Note: the input and output limits are the best estimates of those which will prevail on the actual system. It is not certain that these limits are physically attainable.

The primary design is undertaken for the 100% load operating point. The controller is then evaluated on the other models in order to obtain an indication of robustness.

The following tests are then undertaken:

a) Apply a pressure step disturbance of -0.2bar to the system (at t = 30 seconds). The above constraint criteria must not be violated. Run the simulation for 5minutes (t = 300 seconds) and calculate the integral of absolute error for the CV and pressure outputs over the run.

b) Apply a sine wave pressure disturbance of amplitude 0.2bar and frequency of 0.04Hz. Over a 300 second run calculate the integral of absolute error (IAE) as above.

c) Repeat tests a) and b) at the 50% and 0% load operating points, again calculating the integral of absolute error performance criterion.

9.4 Multiobjective PI Control Formulation

The gasifier (or gasification plant) is a nonlinear system. In the ALSTOM benchmark challenge, the gasifier is linearised about the three operating points: 100%, 50% and 0% loads. For the three cases, the gasifier is assumed to be of the following continuous state-space form:

$$\dot{x} = Ax + Bu + Ed \tag{9.4.1}$$

$$y = Cx + Du \tag{9.4.2}$$

where $x \in \mathcal{R}^n$ is the state vector, $u \in \mathcal{R}^m$ the input vector, $y \in \mathcal{R}^r$ the output vector, $d \in \mathcal{R}^1$ the disturbance and the system matrices are $A \in \mathcal{R}^{n\times n}$, $B \in \mathcal{R}^{n\times m}$, $C \in \mathcal{R}^{r\times n}$, $D \in \mathcal{R}^{r\times m}$ and $E \in \mathcal{R}^{n\times 1}$.

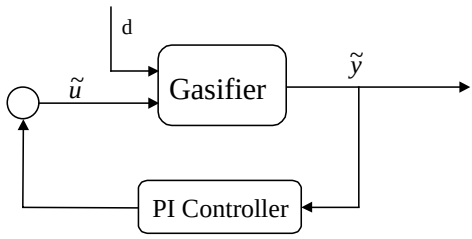

Figure 9.2: Gasifier control with PI controller

A MIMO PI controller is to be applied to the gasifier. The closed-loop control structure with the MIMO PI controller is shown in Figure 9.2. Here, the MIMO PI controller is of the following state-space form:

$$\dot{z} = A_c z + B_c \tilde{y} \tag{9.4.3}$$

$$y = K_I x + K_P \tilde{u} \tag{9.4.4}$$

where $z \in \mathcal{R}^r$ represents the controller state vector, $\tilde{u} \in \mathcal{R}^m$ and $\tilde{y} \in \mathcal{R}^r$ represent the input and output fluctuation from steady-state values, respectively, and the matrices $A_c \in \mathcal{R}^{r\times r}$, $B_c \in \mathcal{R}^{r\times r}$, $K_I \in \mathcal{R}^{m\times r}$, $K_P \in \mathcal{R}^{m\times r}$ have the following form (Liu *et al.*, 1998, 2001):

$$A_c = \begin{bmatrix} -\alpha & & & \\ & -\alpha & & \\ & & \ddots & \\ & & & -\alpha \end{bmatrix} \tag{9.4.5}$$

$$B_c = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \tag{9.4.6}$$

$$K_I = \begin{bmatrix} K^I_{11} & K^I_{12} & \cdots & K^I_{1r} \\ K^I_{21} & K^I_{22} & \cdots & K^I_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ K^I_{m1} & K^I_{m2} & \cdots & K^I_{mr} \end{bmatrix} \tag{9.4.7}$$

$$K_I = \begin{bmatrix} K^I_{11} & K^I_{12} & \cdots & K^I_{1r} \\ K^I_{21} & K^I_{22} & \cdots & K^I_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ K^I_{m1} & K^I_{m2} & \cdots & K^I_{mr} \end{bmatrix} \tag{9.4.8}$$

It is clear that K_I and K_P are the integral and proportional matrices of the MIMO PI controller. α in the matrix A_c must not be less than zero. If $\alpha = 0$, the controller has a pure integral part. In practice, α is always chosen to be a small positive number to ensure the PI controller itself is stable. Since the gasifier outputs, inputs and input rates are constrained, a set of performance functions should be considered during the design of the controller. For this purpose, the following performance functions are introduced:

$$\phi_i(K) = \frac{|\tilde{y}_i|}{y_i^d}, \quad \text{for} \quad i = 1, 2, ..., r \tag{9.4.9}$$

$$\phi_{r+j}(K) = \frac{|\tilde{u}_j|}{u_j^d}, \quad \text{for} \quad j = 1, 2, ..., m \tag{9.4.10}$$

$$\phi_{r+m+k}(K) = \frac{|\dot{u}_k|}{\delta u_k^d}, \quad \text{for} \quad k = 1, 2, ..., m \tag{9.4.11}$$

$$\phi_{r+2m+1}(K) = 1 + \max_{j=1,2,...,n}\{Re(\lambda_j)\} + \varepsilon \tag{9.4.12}$$

where $\tilde{y}_i$ is the i-th output fluctuation and $\tilde{u}_i$ the j-th input fluctuation, K the parameter vector denotes the elements of the matrices K_I and K_P of the PI controller, λ_j is the j-th pole of the closed-loop system, $Re(.)$ denotes the real part of a number, ε represents the requirements on the closed-loop poles, y_i^d, u_j^d and δu_k^d are the upper bounds on the output fluctuation, input fluctuation and input rates, respectively.

The evaluation of the above performance functions for each PI controller is carried out in the offline closed-loop system, which consists of the gasifier linearised model and the MIMO PI controller. In practice, when the gasifier changes the operating point, its model should be updated using an estimation algorithm, *e.g.*, the recursive least squares algorithm. Then, based on the updated model, the performance functions can be re-evaluated. In (9.4.12), ε should be chosen to be a number so that the closed-loop system is stable. The larger the ε, the more stable the system. For this benchmark problem, there is not any specific requirement on the closed-loop system. Thus, during the design of the PI controller, ε can simply be chosen to be a positive number, i.e., $\varepsilon > 0$.

According to the requirements on the gasifier, the following multiobjective performance criteria should be satisfied:

$$\phi_i(K) \leq 1, \quad \text{for} \quad i = 1, 2, ..., r + 2m + 1 \tag{9.4.13}$$

If the above inequalities are satisfied, then the problem is solved. Thus, the design problem is to find a PI controller to make (9.4.13) hold.

9.5 Multiobjective Optimal-Tuning PI Control

There are a number of methods to solve the multiobjective performance criteria problem (9.4.13). Here, two methods are briefly introduced. One is the minimax optimisation method and the other is the method of inequalities. Using the minimax optimisation method (Gill, 1981), the multiobjective performance criteria can be satisfied if

$$\min_{K} \max_{i=1,2,...,r+2m+1}\{\phi_i(K)\} \leq 1 \tag{9.5.1}$$

Clearly, the above minimises the worst case values of the performance functions. The method of inequalities (Zakian and Al-Naib, 1973; Whidborne and Liu, 1993) uses optimisation algorithms (*e.g.*, moving boundaries algorithm) to find the admissible or feasible set of parameter vectors, for which all the performance inequalities hold. The admissible set is defined as

$$\Omega = \bigcap_{i=1,2,...,r+2m+1} \Omega_i \tag{9.5.2}$$

where Ω_i is the set of parameter vectors for which the i-th functional inequality is satisfied, that is

$$\Omega_i = \{K, \phi_i(K) \leq 1\} \tag{9.5.3}$$

Similar algorithms for solving the problem defined in (9.4.13) now exist in standard libraries of optimisation software, *e.g.* the optimisation toolbox for use with MATLAB (Grace, 1994).

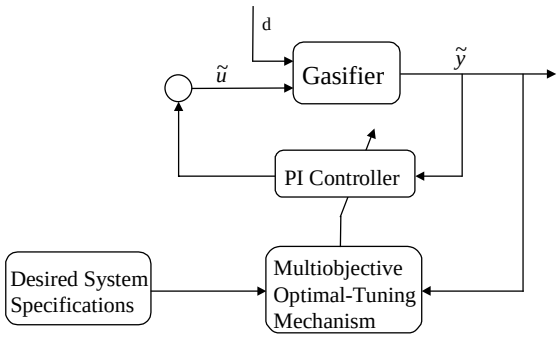

Figure 9.3: Multiobjective optimal-tuning PI control

If a system has different operating points with widely differing dynamic properties, it is not always possible to control with a fixed parameter controller, even if this is a highly robust controller. For this case, the multiobjective optimal-tuning PI control is proposed as shown in Figure 9.3 (Liu *et al.*, 1998; Liu and Daley, 2001). When the system changes its operating point, the multiobjective optimal-tuning mechanism will be switched on to find the optimal parameters for the PI controller to satisfy the multiobjective performance criteria. In this way, the PI controller may cope with all operating points of the gasifier and the closed-loop system will have optimal control performance. But, compared with fixed parameter control, the disadvantage of this strategy is that it needs more computation time to search for the optimal parameters. In addition, there may be some undesired transient response when the new parameters are applied.

In the ALSTOM gasifier benchmark challenge problem, the gasifier has three operating points: 100%, 50% and 0% loads. To have a consistent control performance for all three different loads, the multiobjective optimal-tuning PI control was applied and the expected results were obtained.

9.6 Simulation Results and Discussions

The design was undertaken using the linearised model of the gasifier at the 100% load operating point. The four control inputs and the four outputs of

the gasifier are

$$u = \begin{bmatrix} WCHR & \text{(Char extraction flow)} \\ WAIR & \text{(air mass flow)} \\ WCOL & \text{(coal flow)} \\ WSTM & \text{(steam mass flow)} \end{bmatrix} \tag{9.6.1}$$

$$y = \begin{bmatrix} CVGAS & \text{(fuel gas calorific value)} \\ MASS & \text{(bed mass)} \\ PGAS & \text{(fuel gas pressure)} \\ TGAS & \text{(fuel gas temperature)} \end{bmatrix} \tag{9.6.2}$$

There are twenty-five state variables, *i.e.* $n = 25$. The disturbance on the gasifier is the sink pressure which represents the pressure upstream of the gas turbine. Thus, the MIMO PI controller is a 4-by-4 PI controller, which consists of 16 P-parameters and 16 I-parameters. Also, there are 13 performance function criteria which need to be met during the controller design. Based on the linearised model of the gasifier at the 100% load operating points with a step pressure disturbance, all 13 performance function criteria were successfully satisfied by the 4-by-4 PI controller (denoted by K_{op}) using the multiobjective optimisation algorithms provided by the optimisation toolbox for use with MATLAB (Grace, 1994). Three simulations were carried out for the optimal PI controller, which are given below.

Table 9.2: 100% load operating point case

Input/Output	Max & Min Absolute Values				Peak Rate		IAE	
	Step		Sine		Step		Sine	
	Max	Min	Max	Min				
WCHR (kg/s)	1.00	0.28	1.05	0.75	0.02	0.02		
WAIR (kg/s)	18.49	17.26	18.15	16.65	0.31	0.18		
WCOL (kg/s)	9.18	8.55	8.73	8.18	0.14	0.08		
WSTM (kg/s)	3.47	2.70	3.52	1.89	0.63	0.21		
CVGAS (J/kg)	4.36e6	4.35e6	4.37e6	4.36e6	178.8	108.8	2.06e6	4.95e5
MASS (kg)	1.00e4	9.97e3	1.00e4	1.00e3	0.05	0.04		
PGAS (N/m2)	2.00e6	1.99e6	2.01e6	1.99e6	726.2	208.4	2.00e5	1.73e6
TGAS (K)	1224	1223	1224	1223	0.04	0.01		

Firstly, the controller K_{op} was used in the closed-loop gasifier simulation system with the model obtained at the 100% load. The simulation results for the 100% load case are briefly given in Table 9.2. The output and control input responses of the closed-loop systems to the step pressure disturbances are shown in Figures 9.4 and 9.5. The results for the sine-wave pressure disturbance responses are given in Figures 9.6 and 9.7. Figure 9.4 shows that in the first 300s the CVGAS variable converges to its steady state very slowly and the MASS variable is slowly drifting from the reference, which causes

poor IAE performance. But, these two variables eventually converge to the desired references, which was observed over a longer time-scale and was also guaranteed by the stability performance function (9.4.12).

Secondly, the same controller K_{op} was applied to the closed-loop gasifier simulation system with the linearised models at the 50% operating point. The simulation results are summarised in Table 9.3 . Table 9.3 shows that all performance requirements of the gasifier at 50% load are satisfied. But, the values of all performance functions are slightly worse than those for the 100% load. This is because the controller was designed at the 100% load operating point rather than at the 50% load.

Thirdly, the same controller K_{op} was also employed in the closed-loop gasifier simulation system with the linearised models at the 0% load operating point. The simulation results are listed in Table 9.4. It shows that some performance requirements at 0% load are violated. For example, the outputs CVGAS and PGAS, and the rates of the control inputs WCOL and WSTM are slightly beyond the desired bounds. The output and control input responses to the step and sine-wave disturbances for 0% load are shown in Figures 9.8-9.11. Though the CVGAS and MASS variables in Figure 9.8 are not settled down in the first 300s, they finally enter into the stable steady state in a longer time-scale.

Table 9.3: 50% load operating point case

Input/Output	Max & Min Absolute Values				Peak Rate		IAE	
	Step		Sine		Step		Sine	
	Max	Min	Max	Min				
WCHR (kg/s)	1.05	0.46	1.03	0.76	0.02	0.02		
WAIR (kg/s)	12.23	10.70	11.68	10.03	0.38	0.20		
WCOL (kg/s)	6.12	5.34	5.62	4.84	0.20	0.12		
WSTM (kg/s)	2.61	1.69	2.63	0.75	0.76	0.24		
CVGAS (J/kg)	4.49e6	4.48e6	4.50e6	4.49e6	263.6	158.8	2.51e6	7.84e5
MASS (kg)	1.00e4	9.96e3	1.00e4	10.0e3	0.05	0.03		
PGAS (N/m2)	1.55e6	1.54e6	1.56e6	1.54e6	881.7	226.0	2.58e5	1.88e6
TGAS (K)	1181	1181	1181	1181	0.05	0.01		

Table 9.4: 0% load operating point case

Input/Output	Max & Min Absolute Values				Peak Rate		IAE	
	Step		Sine		Step		Sine	
	Max	Min	Max	Min				
WCHR (kg/s)	0.89	0.00	0.71	0.30	0.03	0.03		
WAIR (kg/s)	6.87	4.06	5.31	3.13	0.66	0.26		
WCOL (kg/s)	3.37	2.14	2.74	1.21	0.39	0.26		
WSTM (kg/s)	2.07	0.68	12.0	0.00	1.33	0.34		
CVGAS (J/kg)	4.71e6	4.69e6	4.72e6	4.70e6	518.6	350.0	4.07e6	1.65e6
MASS (kg)	1.00e4	9.95e3	1.00e4	10.0e3	0.06	0.06		
PGAS (N/m2)	1.12e6	1.11e6	1.13e6	1.11e6	1.5e3	290.5	4.97e5	2.34e6
TGAS (K)	1116	1115	1116	1115	0.10	0.01		

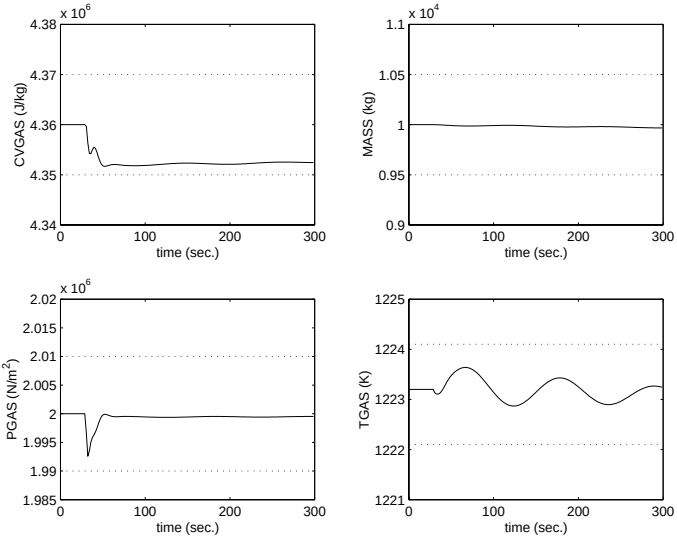

Figure 9.4: Output response to step disturbance for 100% load

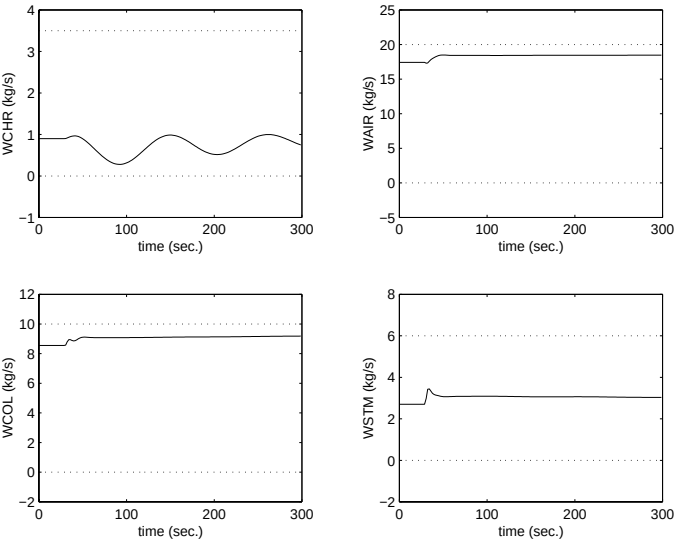

Figure 9.5: Control input response to step disturbance for 100% load

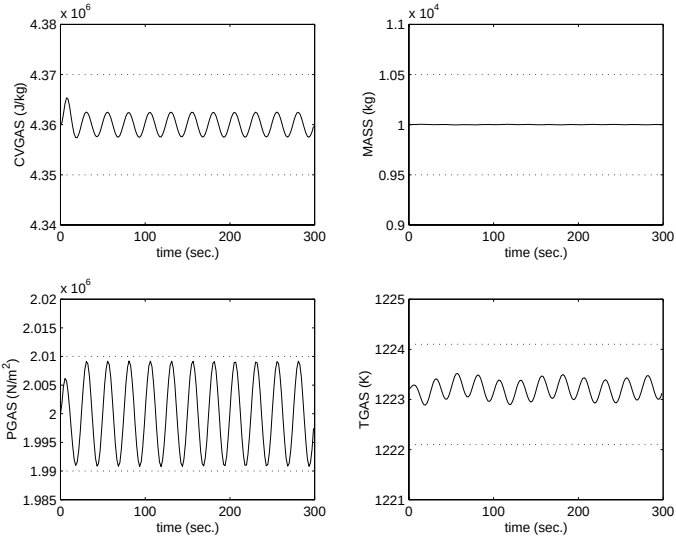

Figure 9.6: Output response to sine wave disturbance for 100% load

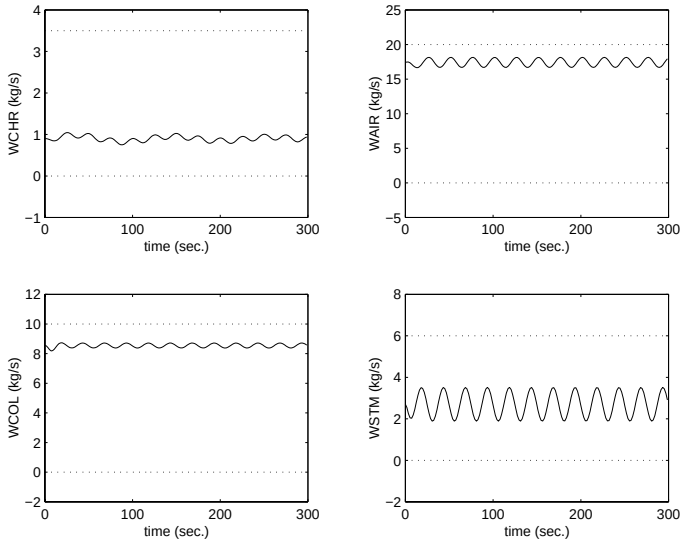

Figure 9.7: Control input response to sine wave disturbance for 100% load

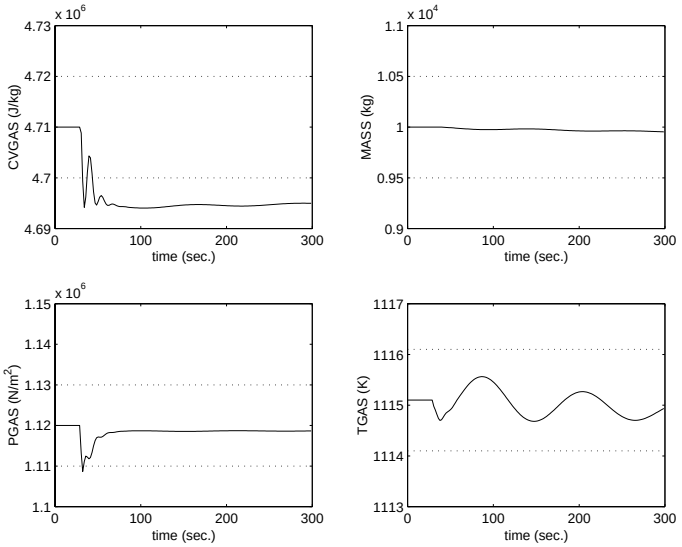

Figure 9.8: Output response to step disturbance for 0% load

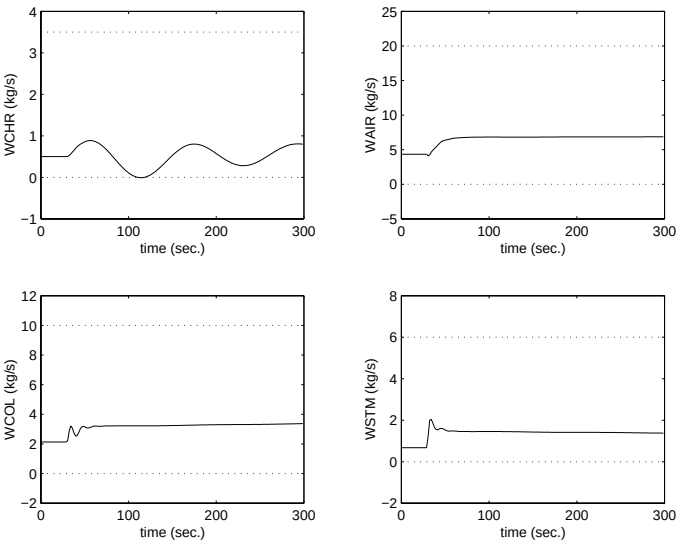

Figure 9.9: Control input response to step disturbance for 0% load

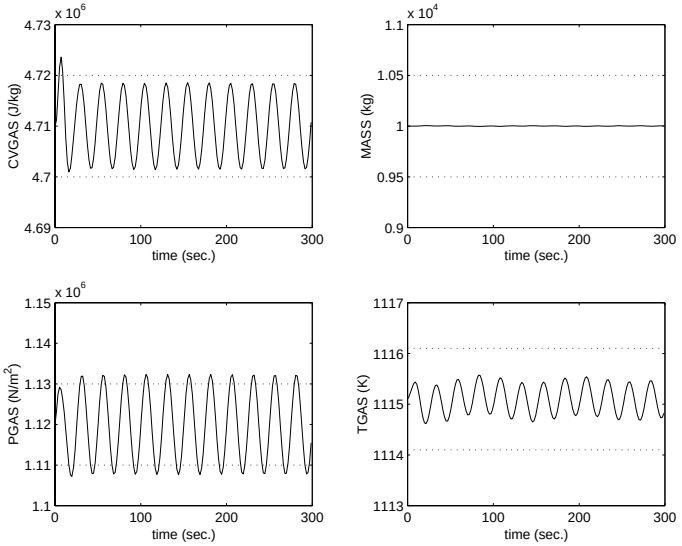

Figure 9.10: Output response to sine wave disturbance for 0% load

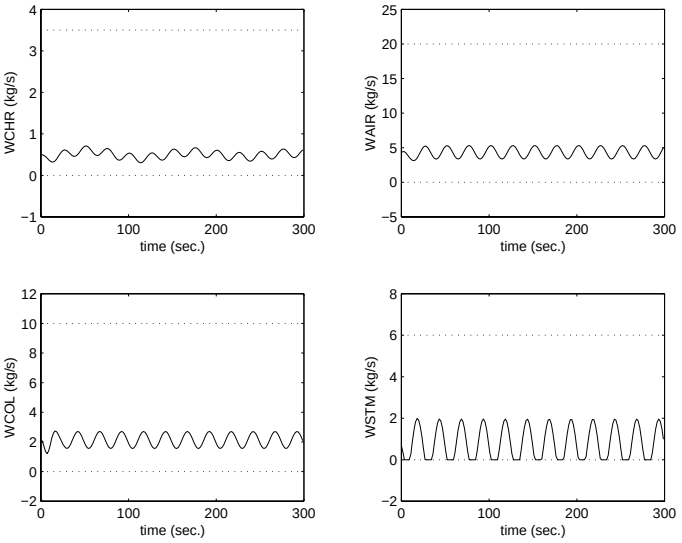

Figure 9.11: Control input response to sine wave disturbance for 0% load

It has been shown from the above that when the PI controller parameters designed for the 100% load case were applied to the 0% load case, the control requirements of the benchmark challenge were not satisfied. Thus, to cope with the 0% load case, the multiobjective optimal-tuning mechanism was switched on again to adapt to the change in operating point of the gasifier. Similar control performance results to the 100% load case were then obtained. Though this is not required by the benchmark challenge, it shows that the multiobjective optimal-tuning control is able to cope with a wide range of gasifier operating points.

9.7 Summary

A PI controller optimised using multiobjective methods has been designed for the ALSTOM benchmark challenge on gasifier control. The specifications on the outputs, inputs and input rates of the gasifier were formulated as multiobjective performance function criteria that are expressed by a set of inequalities. For the 100% load operating point, the MIMO PI controller has been successfully designed using multiobjective control techniques to satisfy all specifications when the step pressure disturbance was applied to the gasifier. The same controller also met all performance requirements of the gasifier at 50% load. Although some specifications at 0% load were not satisfied by this controller, this is not thought to be a major problem since they were satisfied when the multiobjective optimal-tuning mechanism was switched on again. Compared with fixed parameter controllers, the multiobjective optimal-tuning PI control needs more computation to re-tune the controller parameters for the case where the system operating point or dynamics changes. This may cause a problem in the implementation of this control strategy for fast response control systems. But it provides much better control performance than the fixed parameter controllers. Since the PI controller presented in this chapter is stable, has a simple structure and provides good control performance, it will be a very suitable controller for industrial gasifiers.

Chapter 10

Multiobjective PID Controller Implementation Design

10.1 Introduction

It is well known that controller implementations with fixed-point arithmetic offer a number of advantages over floating-point implementations (Clarke, 1997; Masten and Panahi, 1997). Fixed-point processors contribute to lower overall system costs and require less on-chip hardware. They have a smaller package size, less weight, consume less power and are often faster than floating-point processors. In addition, the simplicity of the architecture contributes to safer implementations. Thus for high-volume, price-sensitive applications, safety critical applications and hand-held and aerospace applications where power consumption and weight are important considerations, fixed-point processors with a low word-length are preferred over low word-length, floating-point processors. However, a closed-loop control system will suffer a performance degradation and may even become unstable when the designed infinite-precision controller is implemented with a fixed-point digital processor due to the finite precision of the parameter representation resulting from the Finite Word-Length (FWL). This so-called FWL effect is in fact strongly dependent upon the parameterisation of the controller. Thus, over the years, many results have been reported in the literature dealing with FWL implementation and their relevant parameterisation issues, for example, Knowles and Edwards (1965), Morony (1983), Williamson (1991), Gevers and Li (1993), Istepanian *et al.* (1998b), Istepanian and Whidborne (2001).

Consider the discrete time system shown in Figure 10.1 with the plant $G(z)$. Let $(A_k, B_k, C_k,\ D_k)$ be a state-space description of the state-space

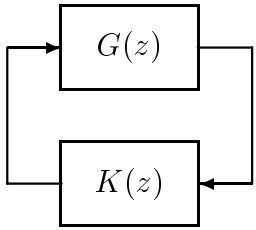

Figure 10.1: Discrete time feedback system

controller,

$$K(z) := C_k(zI - A_k)^{-1}B_k + D_k \tag{10.1.1}$$

In this chapter, (A_k, B_k, C_k, D_k) is also called a realisation of $K(z)$. The realisations of $K(z)$ are not unique, if $(A_k^0, B_k^0, C_k^0, D_k^0)$ is a realisation of $K(z)$, then $(T^{-1}A_k^0T, T^{-1}B_k^0, C_k^0T, D_k^0)$ is an equivalent realisation for any non-singular similarity transformation T. A common approach, for example Gevers and Li (1993), Istepanian *et al.* (1998b), Whidborne *et al.* (2000), to the FWL problem for state-space controllers is to to find equivalent controller realisations (or similarity transformations T) such that the closed-loop system is in some way insensitive to perturbations in the controller parameters.

A more direct Genetic Algorithm (GA) based approach has been developed (Whidborne and Istepanian, 2001a) and is presented in this chapter. Basically, the approach is to find an FWL controller that is near to the originally designed controller such that the closed-loop performance and robustness degradation and the FWL implementation cost are simultaneously minimised. The approach is based on the generation of near-equivalent finite word-length controller representations by means of the solution to a linear system equivalence completion problem followed by a rounding operation. A Multiobjective Genetic Algorithm (MOGA) (Fonseca and Fleming, 1993a,b, 1995), is then used to find sets of Pareto-optimal near-equivalent FWL controllers. This allows the designer to trade-off FWL implementation performance and robustness against the cost and other requirements of the implementation by enabling the designer to simply choose the most appropriate design from the set.

Two features of GAs make them very attractive for solving this problem. Firstly, GAs require the solution space to be encoded in binary strings, and since controllers implemented with FWL are also coded in binary, a one-to-one relationship between the genotype and the phenotype within the GA can be defined. Secondly, GAs allow the optimal structure of the solution to be found (Man *et al.*, 1997; Tang *et al.*, 1996; Dakev *et al.*, 1997). This means that the implementation word-length does not need to be defined *a priori*, but

the GA will select the best from a predefined set, and so the implementation cost in the form of the memory requirement can also be minimised.

The developed approach is applied to two problems. The first is the implementation of a PID controller for a steel rolling mill problem (Hori, 1996) which was also solved using a hill-climbing approach by Istepanian *et al.*(1998b). The second problem is the implementation of a PID controller designed for the IFAC93 benchmark problem (Graebe, 1994) by Whidborne *et al.* (1995a). The method is entirely generic with respect to the objective cost functions, so any set of stability/performance and implementation measures could be used. In this chapter, for the first problem, the performance and stability are assessed simply by the maximal change in closed-loop poles, and for the second, by the H_∞-norm of the difference between the FWL closed-loop transfer function and the original closed-loop transfer function. The implementation cost for both examples is taken as the parameter memory storage requirement.

10.2 FWL Fixed-Point Representation

A typical 2s complement FWL fixed-point representation of a number $q(x)$ is shown in Figure 10.2. The number $q(x)$ is represented by an $m+n+1$ binary string x where $x = [x_0, x_1 \ldots, x_m, x_{m+1}, \ldots, x_{m+n}]$, $x_i \in \{0,1\}$, $m \in \{0,1,2,\ldots\}$, $n \in \{0,1,2,\ldots\}$. The value $q(x)$ is given by

$$q(x) = -x_0 2^m + \sum_{i=1}^{m} x_i 2^{i-1} + \sum_{i=m+1}^{m+n} x_i 2^{m-i} \tag{10.2.1}$$

The set of possible values which can be taken by an FWL variable represented by a $m+n+1$ binary string x is defined as $\mathcal{Q}_{n,m}$ given by $\mathcal{Q}_{n,m} = \{q : q = q(x), x_i \in \{0,1\}, i = 0, \ldots, n+m\}$. The set of $p \times r$ matrices with elements from $\mathcal{Q}_{n,m}$ is denoted by $\mathcal{Q}_{n,m}^{p\times r}$.

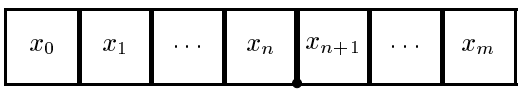

Figure 10.2: FWL fixed-point representation

10.2.1 A Linear System Equivalence Completion Problem

In the proposed method, near-equivalent finite word-length controller representations are generated by means of the solution to a linear system equivalence completion problem; some preliminary results are first required. The definition below is from Hernstein and Winter (1988, pp. 154).

Definition 10.1 *A matrix A is similar to a matrix $\tilde{A}$ if and only if there exists a non-singular matrix T such that $\tilde{A} = T^{-1}AT$.*

Lemma 10.1 *Given a matrix $A \in \mathcal{R}^{2\times 2}$ such that $A \neq \alpha I \; \forall \; \alpha \in \mathcal{R}$, where*

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{10.2.2}$$

and given the real pair $(\tilde{a}_{11}, \tilde{a}_{12} \neq 0)$, then A is similar to $\tilde{A}$ where

$$\tilde{A} = \begin{bmatrix} \tilde{a}_{11} & \tilde{a}_{12} \\ \tilde{a}_{21} & \tilde{a}_{22} \end{bmatrix} \tag{10.2.3}$$

and where $\tilde{a}_{22} = a_{11} + a_{22} - \tilde{a}_{11}$ and $\tilde{a}_{21} = (\tilde{a}_{11}\tilde{a}_{22} - \det A)/\tilde{a}_{12}$.

Proof: To prove this lemma, we use the property that if A and $\tilde{A}$ have the same minimal polynomial and the same characteristic polynomial, and their minimal polynomial is the same as their characteristic polynomial, then A and $\tilde{A}$ are similar (Horn and Johnson, 1985, pp. 150). The characteristic equations of A and $\tilde{A}$ are $p_A(t) = \det(tI - A) = t^2 - (a_{11} + a_{22})t + (a_{11}a_{22} - a_{21}a_{12})$ and $p_{\tilde{A}}(t) = \det(tI - \tilde{A}) = t^2 - (\tilde{a}_{11} + \tilde{a}_{22})t + (\tilde{a}_{11}\tilde{a}_{22} - \tilde{a}_{21}\tilde{a}_{12})$ respectively. By equating coefficients we obtain the expressions for $\tilde{a}_{21}$ and $\tilde{a}_{22}$. It remains to be shown that the minimal polynomial of A is the same as the characteristic polynomial. For the case where the eigenvalues of A are unique, then it is clear that the minimal polynomial is the same as the characteristic polynomial. For the case where the eigenvalues of A are repeated and A is non-derogatory, then the Jordan canonical form of A is

$$J_A = \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix} \tag{10.2.4}$$

where λ is the repeated eigenvalue and the minimal polynomial is the same as the characteristic polynomial (Barnett, 1971, pp. 6). However, if A is diagonal with repeated eigenvalues, that is $A = \lambda I$, then $A = T^{-1}AT$ for all non-singular T. Thus the Jordan canonical form equals A and A is derogatory and so the minimal polynomial is of lower order than the characteristic polynomial. This form is explicitly excluded.

The derogatory case excluded in the above lemma is shown to be impractical for controller implementation by the following lemma.

Lemma 10.2 *For a two state SISO LTI system $F(z)$, where*

$$F \stackrel{s}{=} \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \tag{10.2.5}$$

$$\stackrel{s}{=} \left[\begin{array}{cc|c} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ \hline c_1 & c_2 & d \end{array}\right] \tag{10.2.6}$$

then the system is unobservable if $A = \alpha I, \quad \forall\, \alpha \in \mathcal{R}$.

Proof: If $A = \alpha I$, the observability matrix

$$\mathcal{O} = \begin{bmatrix} C \\ CA \end{bmatrix} \tag{10.2.7}$$

is given by

$$\mathcal{O} = \begin{bmatrix} c_1 & c_2 \\ \alpha c_1 & \alpha c_2 \end{bmatrix} \tag{10.2.8}$$

which is rank deficient, hence the system is unobservable.

The main theorem in which a linear system equivalence completion problem is solved can now be stated.

Theorem 10.1 *Given an observable two state SISO LTI system $F(z)$, where*

$$F \stackrel{s}{=} \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \tag{10.2.9}$$

$$\stackrel{s}{=} \left[\begin{array}{cc|c} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ \hline c_1 & c_2 & d \end{array}\right] \tag{10.2.10}$$

then given $(\tilde{a}_{11}, \tilde{a}_{12} \neq 0, \tilde{c}_1, \tilde{c}_2)$ there exists an equivalent state-space representation $\tilde{F}$ where

$$\tilde{F} \stackrel{s}{=} \left[\begin{array}{c|c} \tilde{A} & \tilde{B} \\ \hline \tilde{C} & \tilde{D} \end{array}\right] \tag{10.2.11}$$

$$\stackrel{s}{=} \left[\begin{array}{cc|c} \tilde{a}_{11} & \tilde{a}_{12} & \tilde{b}_1 \\ \tilde{a}_{21} & \tilde{a}_{22} & \tilde{b}_2 \\ \hline \tilde{c}_1 & \tilde{c}_2 & \tilde{d} \end{array}\right] \tag{10.2.12}$$

such that $\tilde{F}(z) = F(z)$ where

$$F(z) = C\left[zI - A\right]^{-1} B + D \tag{10.2.13}$$

and

$$\tilde{F}(z) = CT\left[zI - T^{-1}AT\right]^{-1} T^{-1}B + D \tag{10.2.14}$$

where

$$T = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \tag{10.2.15}$$

is non-singular if $(\tilde{A}, \tilde{C})$ *is observable.*

Proof: The proof is by construction. From (10.2.14),

$$\tilde{A} = T^{-1}AT \tag{10.2.16}$$

From definition 10.1, and from lemma 10.2, since the system is observable, the case $A = \alpha I \; \forall \; \alpha \in \mathcal{R}$ can be excluded. Hence from lemma 10.1, $\tilde{a}_{22}$ is given by $\tilde{a}_{22} = a_{11} + a_{22} - \tilde{a}_{11}$ and $\tilde{a}_{21}$ is given by $\tilde{a}_{21} = (\tilde{a}_{11}\tilde{a}_{22} - \det A)/\tilde{a}_{12}$. From (10.2.16), $AT - T\tilde{A} = 0$, which gives (Horn and Johnson, 1991, pp. 255)

$$[I \otimes A - \tilde{A}^T \otimes I]\,\mathbf{t} = 0 \tag{10.2.17}$$

where $\mathbf{t} = [t_{11}, t_{21}, t_{12}, t_{22}]^{\mathcal{T}}$ and $[I \otimes A - \tilde{A}^T \otimes I]$ is rank 2. Now, from (10.2.14), $CT = \tilde{C}$. Hence, given $\tilde{c}_1, \tilde{c}_2$,

$$\begin{bmatrix} c_1 & c_2 & 0 & 0 \\ 0 & 0 & c_1 & c_2 \end{bmatrix} \mathbf{t} = \begin{bmatrix} \tilde{c}_1 \\ \tilde{c}_2 \end{bmatrix} \tag{10.2.18}$$

A $Y \in \mathcal{R}^{4\times 4}$ can be constructed from (10.2.17) and (10.2.18) where $Y\mathbf{t} = z$, where Y is non-singular rank 4 and where $z \in \mathcal{R}^4$ is a column vector with 2 elements of $\tilde{C}$ and two zero elements. Hence $\mathbf{t}$ can be calculated and T obtained. Since F is observable, then the observability matrix

$$\mathcal{O} = \begin{bmatrix} C \\ CA \end{bmatrix} \tag{10.2.19}$$

is rank 2. Since the pair $(\tilde{A}, \tilde{C})$ is required to be observable, the observability matrix

$$\tilde{\mathcal{O}} = \begin{bmatrix} \tilde{C} \\ \tilde{C}\tilde{A} \end{bmatrix} \tag{10.2.20}$$

is rank 2 and since

$$\begin{bmatrix} \tilde{C} \\ \tilde{C}\tilde{A} \end{bmatrix} = \begin{bmatrix} CT \\ CAT \end{bmatrix} \tag{10.2.21}$$

then clearly T must be non singular. Thus $\tilde{B}$ can be calculated. From (10.2.14), $\tilde{d} = d$.

Note that by redefining the transformation matrix as T^{-1}, the problem given $\tilde{B}$ instead of $\tilde{C}$ can be solved (Whidborne and Istepanian, 2001b). Note also that the only restrictions on the problem are that the original system and the equivalent system are observable, and that certain canonical realisations (*i.e.* diagonal and lower triangular) are excluded by the constraint that a_{12} is not zero. These realisations can be separately programmed into the method if necessary.

10.3 MOGA for Optimal FWL Controller Structures

10.3.1 Multiobjective Genetic Algorithm

Most GAs have been used for single objective problems, although several multiobjective schemes have been proposed (*e.g.* Schaffer, 1985; Wienke *et al.*, 1992). Fonseca and Fleming (1993a,b, 1994) have used an approach called the multiobjective genetic algorithm (MOGA), which is an extension on an idea by Goldberg (1989). This formulation maintains the genuine multiobjective nature of the problem, and is essentially the scheme used here. The idea behind the MOGA is to develop a population of Pareto-optimal or near Pareto-optimal solutions. To restrict the size of the near Pareto-optimal set and to give a more practical setting to the MOGA, Fonseca and Fleming have formulated the problem in a similar way to the MoI. The aim is to find a set of solutions which are non-dominated and which satisfy a set of inequalities. An individual j with a set of objective functions $\phi^j = (\phi_1^j, \ldots, \phi_n^j)$ is said to be *non-dominated* if for a population of N individuals, there are no other individuals $k = 1, \ldots, N, k \neq j$ such that

$$\begin{array}{ll} \text{a)} & \phi_i^k \leq \phi_i^j \text{ for all } i = 1, \ldots, n \quad \text{and} \\ \text{b)} & \phi_i^k < \phi_i^j \text{ for at least one } i. \end{array}$$

The MOGA is set into a multiobjective context by means of the fitness function. The mechanism is described later.

The problem which is solved by the MOGA can be expressed as

Problem 10.1 *Find a set of M admissible points $\mathbf{p}^j, j = 1, \ldots M$ such that $\phi_i^j \leq \varepsilon_i \quad (j = 1, \ldots M, i = 1, \ldots, n)$ and such that $\phi^j (j = 1, \ldots M)$ are non-dominated.*

Algorithm

The algorithm used in this study is:

- **S1** Create a chromosome population of N individuals
- **S2** Decode chromosomes to obtain phenotypes $\mathbf{p}^j \in \mathcal{P} (j = 1, \ldots, N)$
- **S3** Calculate index vectors $\phi^j (j = 1, \ldots, N)$.
- **S4** Rank individuals and calculate fitness functions $f_j (j = 1, \ldots, N)$
- **S5** Make selection of N individuals based on fitness
- **S6** Perform cross-over on selected individuals
- **S7** Perform mutation on some individuals
- **S8** With new chromosome population, return to S2

The algorithm is terminated when M admissible points have been found. The MATLAB Genetic Algorithms Toolbox (Chipperfield *et al.*, 1994) has been used to implement the GA.

Many variations of the GA have been suggested, with different schemes for chromosoid representation, ranking and fitness calculation, selection, crossover and mutation. Usually, the MOGA will use schemes for selection, crossover and mutation which are typical for standard genetic algorithms, as described in Chapter 5. It is in the scheme for the ranking and fitness where the multi-objective nature of the MOGA is characterised. In addition, the MOGA also requires an additional mechanism in assessing the fitness, known as 'fitness sharing'. In general, the scheme for encoding the chromosoid representation of the population for a MOGA will also be as for standard genetic algorithms (see section 5.4). However, for the FWL controller structure application described in this chapter, a particular representation is required, the encoding of which is described in section 10.3.3.

Ranking and Fitness

Fonseca and Fleming (1993a) have proposed a fitness scheme for MOGA to solve the MoI which maintains the genuine multiobjective nature of the problem. The scheme needs the concept of one individual being *preferable* to another.

Definition 10.2 *Consider two individuals a and b with objective function sets* ϕ^a *and* ϕ^b *respectively, and with a set of design goals* $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)$. *Three possible cases can occur with respect to individual a:*

a) **Individual** a **satisfies none of the inequalities**
That is, if $\phi_i^a > \varepsilon_i \ \forall \ i = 1, \ldots, n$, *then individual a is* preferable *to individual b if and only if*

$$\phi_i^a \leq \phi_i^b, \quad \forall \, i = 1, \ldots, n \tag{10.3.1}$$

and there exists at least one ϕ_i^a *such that* $\phi_i^a < \phi_i^b$.

b) **Individual** a **satisfies some of the inequalities**
That is, if there exists at least one ϕ_i^a *such that* $\phi_i^a \leq \varepsilon_i$ *and there exists at least one* ϕ_i^a *such that* $\phi_i^a > \varepsilon_i$, *then individual a is* preferable *to individual b if*

$$\phi_i^a \leq \phi_I^b, \quad \forall \ i \text{ such that } \phi_i^a > \varepsilon_i \tag{10.3.2}$$

and there exists at least one $\phi_i^a > \varepsilon_i$ *such that* $\phi_i^a < \phi_i^b$;
or if

$$\phi_i^a = \phi_i^b \ \forall \ i \text{ such that } \phi_i^a > \varepsilon_i \tag{10.3.3}$$

then individual a is preferable *to individual b if*

$$\phi_i^a \leq \phi_i^b, \quad \forall \ i \text{ such that } \phi_i^a \leq \varepsilon_i \tag{10.3.4}$$

and there exists at least one $\phi_i^a \leq \varepsilon_i$ *such that* $\phi_i^a < \phi_i^b$,
or if there exists a

$$\phi_i^b > \varepsilon_i \text{ for some } i \text{ such that } \phi_i^a \leq \varepsilon_i \tag{10.3.5}$$

c) **Individual a satisfies all the inequalities**
That is, if $\phi_i^a \leq \varepsilon_i \ \forall \ i = 1, \ldots, n$, *then individual* a *is* preferable *to individual* b *if*

$$\phi_i^a \leq \phi_i^b, \quad \forall\, i = 1, \ldots, n \tag{10.3.6}$$

and there exists at least one ϕ_i^a *such that* $\phi_i^a < \phi_i^b$*;*
or if there exists a $\phi_i^b > \varepsilon_i$.

All the individuals are assigned a rank according to how many individuals which are preferable to an individual. Thus, the rank of an individual j is given by r^j and the number of individuals in that generation that are preferable to j are k^j, then $r^j = 1 + k^j$.

To calculate the fitness, the population is sorted according to rank. A fitness value $f^j (j = 1, \ldots, N)$, $0 \leq f^j \leq 2$, is assigned to each individual by a linear interpolation from the best individual ($f^{best} = 2$) to the worst individual ($f^{worst} = 0$). The fitness of all individuals with the same rank is then averaged, so that they are sampled at the same rate. Thus, if n_{gt} is the number of individuals with a rank $r > r_j$, and if n_{eq} is the number of individuals with a rank $r = r_j$, then

$$f_j = \frac{2n_{gt} + n_{eq}}{N - 1} \tag{10.3.7}$$

Figure 10.3 shows an example for 10 individuals where $\phi = (\phi_1, \phi_2)$. The goals $\varepsilon_1, \varepsilon_2$ are also shown. The preference relationship between each pair of individuals and their subsequent rank and fitness is shown in Table 10.1.

Clearly, this scheme could be adapted, if desired, to provide some ordering to the importance of the indices and the inequalities, *e.g.* if some of the inequalities could be described as 'hard' constraints and others as 'soft'.

An alternative, less computationally demanding, ranking scheme has been proposed by Liu *et al.* (1994).

Table 10.1: Preference relationship for ranking example

	a	b	c	d	e	f	g	h	i	j
a is preferable to	–	×	×	×	×	×	×	×	×	×
b is preferable to	√	–	×	×	×	×	×	×	×	×
c is preferable to	√	×	–	×	×	×	×	×	×	×
d is preferable to	×	×	×	–	×	×	×	×	×	×
e is preferable to	×	×	×	√	–	×	×	×	×	√
f is preferable to	√	√	√	√	√	–	×	×	√	√
g is preferable to	√	√	√	√	√	√	–	×	√	√
h is preferable to	√	√	√	√	√	×	×	–	√	√
i is preferable to	×	×	×	√	√	×	×	×	–	√
j is preferable to	×	×	×	×	×	×	×	×	×	–
rank	6	4	4	6	5	2	1	1	4	6
fitness f	0.22	1.11	1.11	0.22	0.67	1.56	1.89	1.89	1.11	0.22

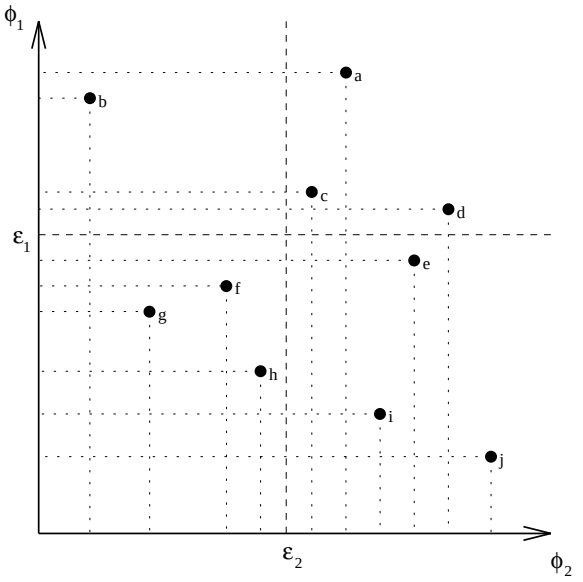

Figure 10.3: Example of multiobjective ranking

Fitness Sharing

In addition to finding a set of near Pareto-optimal individuals, it is desirable that the sample of the whole Pareto-optimal set given by the set of non-dominated individuals is fairly uniform. A common mechanism to ensure this is fitness sharing (Fonseca and Fleming, 1995), which works by reducing the fitness of individuals that are genetically close to each other. However, as will be seen in section 10.3.3, not all the bits within a candidate solution bit string are necessarily active. Thus, two individuals may have the same genotype, but different gene strings. Thus it becomes difficult to measure the difference between two genotypes in order to implement fitness sharing; so, for the sake of simplicity in this study, multiple copies of genotypes are simply removed from the population.

10.3.2 Procedure Outline

The proposed approach is to use the MOGA to evolve a set of near Pareto-optimal solutions to the following problem.

Problem 10.2 *Given a discrete time nominal plant $G(z)$ and designed PID (or other second order) controller $K(z)$, find an FWL controller $K_q(z)$ and state-space parameterisation such that the difference (in some sense) between the closed-loop system and the original closed-loop system and the implemen-*

tation costs are simultaneously minimised.

For each individual in the population, the following procedure is used to generate a possible solution candidate.

S1 Generate a partially filled random FWL parameterisations of the two state controller *i.e.*

$$\tilde{K} \stackrel{s}{=} \left[\begin{array}{cc|c} q_1 & q_2 & ? \\ ? & ? & ? \\ \hline q_3 & q_4 & D_K \end{array}\right] \tag{10.3.8}$$

where $q_j \in \mathcal{Q}_{n,m}, j = 1, \ldots, 4$, *i.e.* each q_j is FWL.

S2 By Theorem 10.1, solve a linear system equivalence completion problem such that $\tilde{K}(z) = K(z)$.

S3 Obtain $K_q \approx \tilde{K}$ by rounding the non-FWL parameters in $\tilde{K}$ so that they are FWL, *i.e.*

$$K_q \stackrel{s}{=} \left[\begin{array}{cc|c} q_1 & q_2 & q_7 \\ q_5 & q_6 & q_8 \\ \hline q_3 & q_4 & q_9 \end{array}\right] \tag{10.3.9}$$

where $q_j \in \mathcal{Q}_{n,m}, j = 1, \ldots, 9$.

S4 Calculate a set of robust stability/performance degradation indices and implementation cost indices, $\{\phi_i : i = 1, \ldots, n\}$ for K_q.

10.3.3 Encoding of Solution Space

The encoding of the solution space is described in this section. Here, it is assumed there is a maximum possible word-length of 16-bits, however, it is a trivial matter to change this to use an 8, 32 or 64-bit maximum word-length.

In order to generate the partially filled parameterisations of K given by (10.3.8), the genotype of each individual consists of a 71-bit binary string $[x_1, \ldots, x_{71}], x_i \in \{0, 1\}$. The bit length of the integer part of the parameters' representation $m \in 0, \ldots, 7$ is represented by $[x_1, x_2, x_3]$, and is given by $m = \sum_{i=1}^{3} x_i 2^{i-1}$. The bit length of the fractional part of the parameters' representation $n \in 0, \ldots, 15$ is represented by $[x_4, \ldots, x_7]$, and is given by $n = \min(\sum_{i=4}^{7} x_i 2^{i-4}, 15 - m)$. The four $m + n + 1$ word-length parameters $q_j \in \mathcal{Q}_{n,m}, j = 1, \ldots, 4$, where $(q_1, q_2, q_3, q_4) = (\tilde{a}_{11}, \tilde{a}_{12}, \tilde{c}_1, \tilde{c}_2)$ respectively, are represented by $[x_{8+16(j-1)}, x_{9+16(j-1)}, \ldots, x_{8+m+n+16(j-1)}]$. Thus not all the bits in x are necessarily active. The values of q_j are calculated by (10.2.1).

10.4 Example – Steel Rolling Mill System

The proposed approach is applied to a PID controller of a steel rolling mill (Hori, 1996). Note that data given in this section are shown to only 4 decimal places.

10.4.1 Performance indices

For this example, the measure of the difference between the FWL implemented closed-loop system and the original closed-loop system is based on the difference between the closed-loop pole positions of the systems.

Consider the discrete time system shown in Figure 10.1. Let (A_g, B_g, C_g, D_g) be a state-space description of the SISO plant $G(z) = C_g(zI - A_g)^{-1}B_g + D_g$, where $A_g \in \mathcal{R}^{n_g \times n_g}$, $B_g \in \mathcal{R}^{n_g \times 1}$, $C_g \in \mathcal{R}^{1 \times n_g}$ and $D_g \in \mathcal{R}$. Let (A_k, B_k, C_k, D_k) be a state-space description of the two state controller, $K(z) = C_k(zI - A_k)^{-1}B_k + D_k$, where $A_k \in \mathcal{R}^{2\times 2}$, $B_k \in \mathcal{R}^{2\times 1}$, $C_k \in \mathcal{R}^{1\times 2}$ and $D_k \in \mathcal{R}$. The transition matrix of the closed-loop system is

$$\bar{A} = \begin{bmatrix} A_g + B_g D_k (I - D_g D_k)^{-1} C_g & B_g (I - D_k D_g)^{-1} C_k \\ B_k (I - D_g D_k)^{-1} C_g & A_k + B_k (I - D_k D_g)^{-1} D_g C_k \end{bmatrix} \tag{10.4.1}$$

When (A_k, B_k, C_k, D_k) is implemented with a finite word-length device, the elements are rounded to the realisation $K_q = (A_q, B_q, C_q, D_q)$, that is $K(z)$ is perturbed to $K_q(z)$, where $K_q(z) = C_q(zI - A_q)^{-1}B_q + D_q$ and where $A_q \in \mathcal{Q}_{n,m}^{2\times 2}$, $B_q \in \mathcal{Q}_{n,m}^{2\times 1}$, $C_q \in \mathcal{Q}_{n,m}^{1\times 2}$ and $D_q \in \mathcal{Q}_{n,m}$. The transition matrix of the FWL implemented closed-loop system is thus

$$\bar{A}_q = \begin{bmatrix} A_g + B_g D_q (I - D_g D_q)^{-1} C_g & B_g (I - D_q D_g)^{-1} C_q \\ B_q (I - D_g D_q)^{-1} C_g & A_q + B_q (I - D_q D_g)^{-1} D_g C_q \end{bmatrix} \tag{10.4.2}$$

A closed-loop measure, η, of the accuracy of the FWL implementation is taken as

$$\eta(K_q) \triangleq \max_{i \in \{1,\ldots,n_g+2\}} \left| \lambda_i - \lambda_i^Q \right| \tag{10.4.3}$$

where $\{\lambda_i : i = 1, \ldots, m+2\}$ represents the set of all eigenvalues of $\bar{A}$ and $\{\lambda_i^Q : i = 1, \ldots, n_g + 2\}$ the corresponding set of all eigenvalues of $\bar{A}_q$.

Two objective functions, $\phi_1(x), \phi_2(2)$ are defined. A measure of implementation accuracy is defined as

$$\phi_1 = \begin{cases} \eta(K_q) & \text{if} \quad \left|\lambda_i^K\right| \leq 1 \forall i \\ & \text{and} \quad \left|\lambda_i^Q\right| < 1 \forall i \\ \max_i \left|\lambda_i^K\right| & \text{if} \quad \left|\lambda_i^K\right| > 1 \exists i \\ \\ \max_i \left|\lambda_i^Q\right| & \text{if} \quad \left|\lambda_i^K\right| \leq 1 \forall i \\ & \text{and} \quad \left|\lambda_i^Q\right| \geq 1 \exists i \end{cases} \tag{10.4.4}$$

where $\{\lambda_i^K : i = 1, 2\}$ represents the pair of eigenvalues of A_q.

An implementation memory function, ϕ_2, is defined as the total number of bits used to implement K_q. This function is calculated bearing in mind that parameters of K_q that are $1, 0, -1$ or that are a power of 2 require less memory requirement than the $m + n + 1$ bits from (10.2.1).

10.4.2 Nominal Plant Model

The continuous time linearised ideal two-mass inertia model of the system plant, $G(s) = C(zI - A)^{-1}B$, is given as

$$A = \begin{bmatrix} 0 & -9763.7203 & 0 \\ 1 & 0 & -1 \\ 0 & 13424.9859 & 0 \end{bmatrix} \tag{10.4.5}$$

$$B = \begin{bmatrix} 249.03 \\ 0 \\ 0 \end{bmatrix}, \quad C = [1 \quad 0 \quad 0] \tag{10.4.6}$$

The plant, $G(s)$, is discretised with sampling period of 0.001 seconds to give $G(z) = C_g(zI - A_g)^{-1}B_g$ as

$$A_g = \begin{bmatrix} 0.9951 & -9.7260 & 0.0049 \\ 0.0010 & 0.9884 & -0.0010 \\ 0.0067 & 13.3732 & 0.9933 \end{bmatrix} \tag{10.4.7}$$

$$B_g = \begin{bmatrix} 0.2486 \\ 0.0001 \\ 0.0006 \end{bmatrix}, \quad C_g = [1 \quad 0 \quad 0] \tag{10.4.8}$$

10.4.3 Controller

A PID vibration suppression and disturbance rejection controller is designed as

$$K(s) = \frac{0.00269s}{0.001s + 1} - 0.435 - \frac{14.26}{s} \tag{10.4.9}$$

The bilinear transform,

$$s = \frac{2}{h}\frac{1 - z^{-1}}{1 + z^{-1}} \tag{10.4.10}$$

gives the digital PID controller as

$$K(z) = -\frac{0.01426}{z - 1} - \frac{1.1956}{z - 0.3333} + 1.3512 \tag{10.4.11}$$

The initial realisation K^0 is set to

$$A_k^0 = \begin{bmatrix} 1 & 0 \\ 0 & 0.3333 \end{bmatrix}, \quad B_k^0 = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \tag{10.4.12}$$

$$C_k^0 = [0.01426 \quad 1.1956], D_k^0 = 1.3512 \tag{10.4.13}$$

10.4.4 Design Results

The MOGA is implemented in MATLAB using the GA Toolbox (Chipperfield *et al.*, 1994). An elitism strategy is used whereby all non-dominated individuals propagate through to the next population. Selection is performed using stochastic universal sampling with a fitness determined by the number of dominating individuals. Single point crossover is used with a probability of 0.7. Each bit has a probability of mutation of 0.00933.

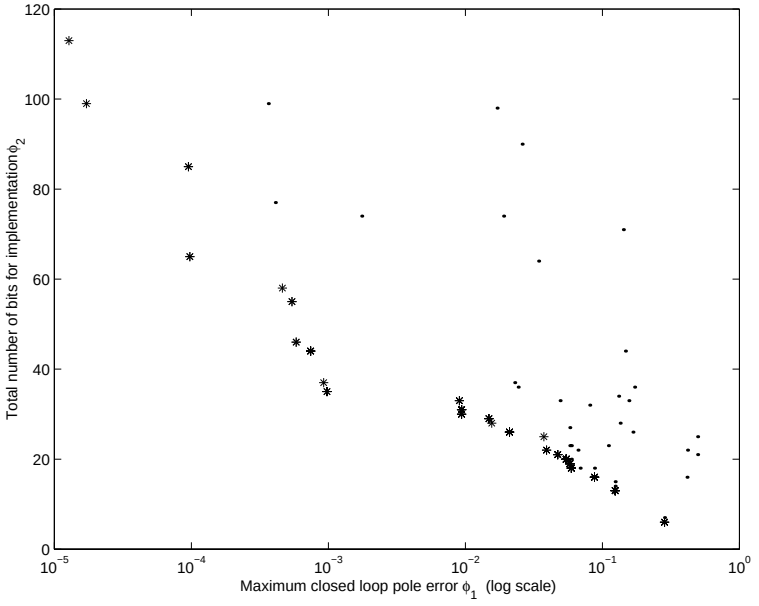

Figure 10.4: Solution set showing trade-off with MOGA non-dominated set ($*$) and MOGA dominated set ($\cdot$)

The MOGA is run with a population of 120. After 400 generations (which takes approximately 1 hour on a 450 MHz Pentium II), a set of non-dominated solutions is obtained. This set is shown in Figure 10.4. Note that the axis for ϕ_1 is shown to log scale for clarity. The figure clearly shows the trade-off between the memory requirements and the stability degradation.

10.5 Example – IFAC93 Benchmark Design

The proposed approach is illustrated by application to the "stress level 2" PID controller (Whidborne *et al.*, 1995a) designed for the IFAC93 benchmark problem (Graebe, 1994). In this problem, the nominal plant is provided along with perturbation ranges for 5 of the plant parameters. The PID controller

has been optimally tuned for a set of robust performance criteria. For further details, see Whidborne *et al.* (1995a).

10.5.1 Performance Indices

To obtain the optimal digital controller structure for this example, two indices are defined for the MOGA. The first index, ϕ_1, is the closed-loop measure of the implementation accuracy of the FWL controller compared to the original controller and is taken as the H_∞-norm of the difference between the closed-loop pole transfer functions of the systems. If both the implemented controller and closed-loop system are stable, ϕ_1 is defined as

$$\phi_1 := \| R - R_q \|_\infty \tag{10.5.1}$$

where

$$R = \frac{GK}{1+GK} \tag{10.5.2}$$

and

$$R_q = \frac{GK_q}{1+GK_q} \tag{10.5.3}$$

This is also a measure of the robust stability/performance degradation.

An implementation memory function, ϕ_2, is defined as the total number of bits used to implement K_q. This function is calculated bearing in mind that parameters of K_q that are $1, 0, -1$ or that are a power of 2 require less memory requirement than the $m+n+1$ bits from (10.2.1).

10.5.2 Nominal Plant Model and Controller

The continuous time nominal plant, $G(s)$, is given as

$$G(s) = \frac{25(1-0.4s)}{(s^2+3s+25)(5s+1)} \tag{10.5.4}$$

The plant, $G(s)$, is discretised with sampling period of $t_s = 0.05$ seconds. A PID controller is designed as in Whidborne *et al.* (1995a):

$$K(s) = 1.311 + 0.431/s + 1.048s/(s+12.92) \tag{10.5.5}$$

and discretised using the bilinear transform. The initial realisation K^0 is set to

$$K_q \stackrel{s}{=} \left[\begin{array}{cc|c} 0 & -0.51172 & 1 \\ 1 & 1.5117 & 0 \\ \hline -0.36524 & -0.17638 & 2.1139 \end{array}\right] \tag{10.5.6}$$

10.5.3 Design Results

The MOGA is implemented in MATLAB using the GA Toolbox (Chipperfield *et al.*, 1994). An elitism strategy is used whereby all non-dominated individuals propagate through to the next population. Selection is performed using stochastic universal sampling with a fitness determined by the number of dominating individuals. Single point crossover is used with a probability of 0.7. Each bit has a probability of mutation of 0.00933.

The MOGA is run with a population of 120. After 800 generations (which takes about 3 hours on a 450 MHz Pentium II), a set of non-dominated solutions is obtained. This set is shown in Figure 10.5. The figure also shows the stable dominated solutions along with FWL implementations of the initial realisation, K_q^0, for various word-lengths. In addition, the equivalent balanced realisation (Zhou *et al.*, 1996, pp. 72-78) of $K(z)$ is calculated and FWL implementations of the realisation for various word-lengths are shown. Note that the axis for $\phi_2 = \| R - R_q \|_\infty$ is shown to log scale for clarity. The figure clearly shows that improved FWL state-space realisations for the controller can be obtained using the approach.

The 32-bit solution labelled (1) in Figure 10.5 is selected and is the controller

$$K_q^{(1)} \stackrel{s}{=} \left[\begin{array}{cc|c} 2^{-1} & -2^{-1} & 2^{-4} \\ 0 & 1 & 0.34375 \\ \hline -1 & -0.9375 & 2.125 \end{array}\right] \tag{10.5.7}$$

which requires $m = 2$ and $n = 5$ for the FWL representation given by (10.2.1). In addition, one of the parameters is zero, two are ± 1, and three others are powers of 2, and can hence be implemented by register shifts. Figure 10.6 shows the frequency response of the original closed-loop transfer function $\left|R(e^{j\omega t_s})\right|$ and the difference between the systems $\left|R(e^{j\omega t_s}) - R_q^{(1)}(e^{j\omega t_s})\right|$. Figure 10.7 shows the step response of the system with the original controller design, $K(s)$, and the digital controller, K_q for the nominal plant and the envelope provided by the 32 plants with the parameters at their extreme values. There is very little difference between the two sets of responses.

For comparison, the 43 bit balanced FWL controller parameterisation labelled (2) in Figure 10.5 is selected and is the controller

$$K_q^{(2)} \stackrel{s}{=} \left[\begin{array}{cc|c} 1 & 0 & -0.15625 \\ 0 & 2^{-1} & -0.625 \\ \hline -0.15625 & 0.625 & 2.125 \end{array}\right] \tag{10.5.8}$$

which also requires $m = 2$ and $n = 5$ for the FWL representation given by (10.2.1). Figure 10.8 shows the step response of the system with the original controller design, $K(s)$, and the digital controller, $K_q^{(2)}$ for the nominal plant and the envelope provided by the 32 plants with the parameters at their extreme values. The deterioration in the step response here is marked.

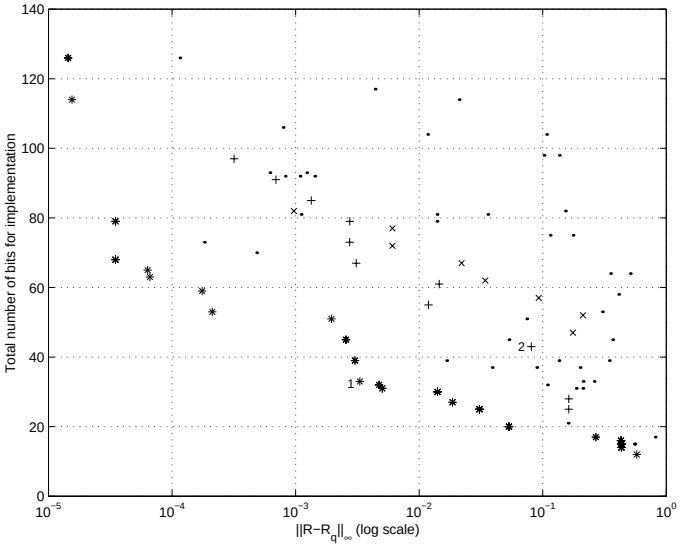

Figure 10.5: Solution set showing trade-off with MOGA non-dominated set ($*$), MOGA dominated set ($\cdot$), K_q^0 realisations ($\times$) and equivalent balanced realisations ($+$). The selected controllers are marked 1 and 2.

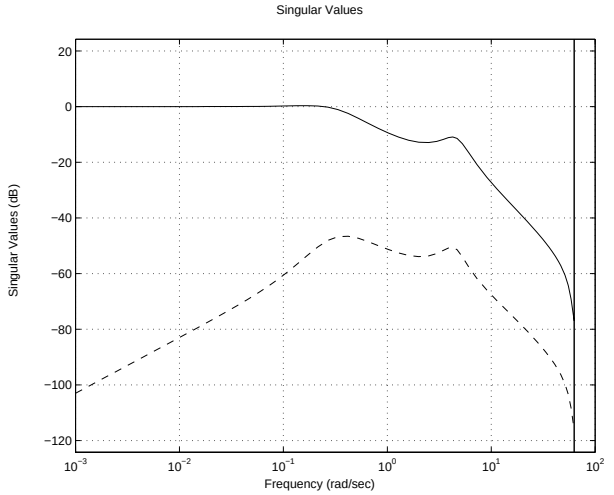

Figure 10.6: Singular values of R (—) and $R - R_q^{(1)}$ (- - -)

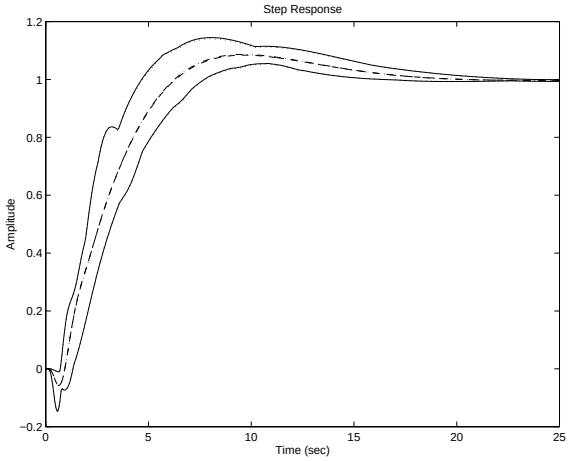

Figure 10.7: Step response for original design, K, with nominal plant ($\cdot$ - $\cdot$ -) and envelope of extreme plants ($\cdots$) and for controller, $K_q^{(1)}$, with nominal plant (- - -) and envelope of extreme plants (—)

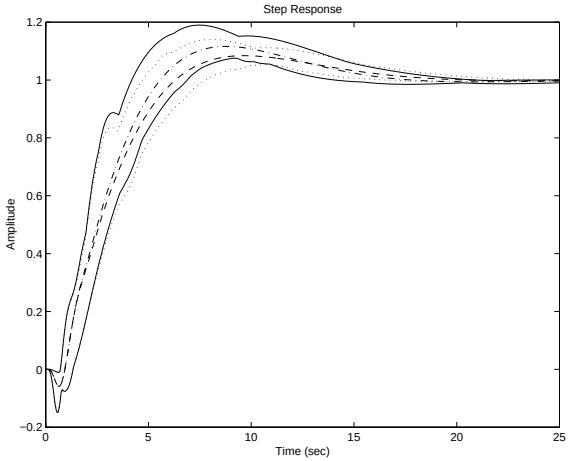

Figure 10.8: Step response for original design, K, with nominal plant ($\cdot$ - $\cdot$ -) and envelope of extreme plants ($\cdots$) and for controller, $K_q^{(2)}$, with nominal plant (- - -) and envelope of extreme plants (—)

10.6 Summary

A multiobjective genetic algorithm based method to determine optimal FWL structures for PID digital controllers has been developed. The method is illustrated by practical examples. The method exploits the fact that the implementation of FWL controllers is by means of binary numbers, as is the representation in genetic algorithms.

The method requires the solution of a linear system similarity completion problem. A solution to the linear system similarity completion problem for two state SISO systems has been presented. This solution would need to be extended to general linear systems for the methodology to be extended to higher order controllers. Otherwise, the method is entirely generic, and any computable set of stability and performance measures could be included. In addition, other implementation measures could be included, such as computation time, as well as measures of other important quantisation effects, namely quantisation noise and scaling requirements. The method is also suitable for controllers implemented using δ operators, which are well-known to have good properties at high sampling rates (Goodwin and Middleton, 1980).

Chapter 11

Multiobjective Nonlinear Identification

11.1 Introduction

Nonlinear system identification can be posed as a nonlinear functional approximation problem. From the Weierstrass Theorem (Powell, 1981) and the Kolmogorov theorem (Sprecher, 1965) in approximation theory, it is known that the polynomial and many other approximation schemes can approximate a continuous function arbitrarily well. In recent years, many nonlinear system identification approaches, particularly identification using neural networks (Chen *et al.*, 1990; Narendra and Parthasarathy, 1990; Billings and Chen, 1992; Qin *et al.*, 1992; Willis *et al.*, 1992; Kadirkamanathan and Niranjan, 1993; Kuschewski *et al.*, 1993; Liu *et al.*, 1996, 1998, 1999), based on the universal approximation theorem (Cybenko, 1989), are applications of a similar mathematical approach.

For nonlinear system identification using the approximation approach, two key questions are important: how to judge the accuracy for the nonlinear function being approximated and how to choose nonlinear function units to guarantee the accuracy. Many nonlinear system identification approaches fix the number of nonlinear function units and use only a single performance function, *e.g.*, L_2-norm of the difference between the real nonlinear system and the nonlinear model which results in the well-known least squares algorithm, to measure and judge the accuracy of the identification model and to optimise the approximation. The assumption behind choosing the L_2-norm is that the noise in the process and measurements has Gaussian (normal) distributions.

However, in nonlinear system identification there are often a number of objectives to be considered. The objectives are often conflicting and no identification which can be considered best with respect to all objectives exists. Hence, there is an inevitable trade-off between objectives, for example, the

distance measurement and maximum difference measurement between the real nonlinear system and the nonlinear model. Model comparison methods, such as Information Criterion (Akaike, 1974), Bayesian model selection (MacKay, 1992) and Minimum Description Length (MDL) (Rissanen, 1989), consider two such objectives, namely, Euclidean distance (L_2-norm) and model complexity. These procedures allow the selection of the best amongst a small number of candidate models (MacKay, 1992). We consider in addition to the above two objectives, the L_∞-norm of the difference between the real nonlinear system and the nonlinear model because it represents the accuracy bound of the approximation achieved by the estimated model. These considerations lead to the study of multiobjective nonlinear system identification.

Recently, artificial neural networks (or simply neural networks, also referred to as connectionist models) have become popular in engineering areas. The interest mainly stems from two aspects: one is to understand and model biological intelligent systems and the other is to devise machines that mimic human behaviour especially in recognising speech and image patterns. The former attempts to model the brain at the microscopic level of neurons and synapses, while the latter tries to find a macroscopic model that is functionally similar to the performance of the human brain. A neural network is modelled by an associative memory which retrieves an appropriate output when presented with an input. The associative memory performs a mapping between the input and the output and is built by learning from examples that are provided in the form of input-output pairs. Neural network learning is thus the identification of an input-output mapping from given examples, which is equivalent to approximating a function based on the information at a set of discrete points. Neural networks are being successfully used as learning machines in applications related to pattern classification and system identification. The success of neural networks has often been attributed to their analogy with their biological counter-part, the human brain. These networks are however far from anything like the human brain and their learning mechanism is even less comparable. The theoretical approaches to understand neural networks have resulted in the use of neural networks to function approximation and nonlinear system modelling.

In this chapter, three multiobjective performance functions are introduced to measure the approximation accuracy and the complexity of the nonlinear model for noise with mixed distribution. Those functions are the L_2- and L_∞-norms of the difference measurements between the real nonlinear system and the nonlinear model, and the number of nonlinear units in the nonlinear model. Genetic algorithms are used to search for a sub-optimal set of nonlinear basis functions of the model to simplify model estimation. A neural network is applied for the model representation of the nonlinear systems, which is the Gaussian radial basis function (GRBF) network. A numerical algorithm for multiobjective nonlinear model selection and identification using neural networks and genetic algorithms is also detailed. Two applications in

identification of a nonlinear system and approximation of a nonlinear function with a mixed noise demonstrate the operation of multiobjective nonlinear identification.

11.2 Neural Networks

The field of neural networks has its roots in neurobiology. The structure and functionality of neural networks has been motivated by the architecture of the human brain. Following the complex neural architecture, a neural network consists of layers of simple processing units coupled by weighted inter-connections. A number of neural networks have been proposed in recent years.

The multilayer Perceptron (MLP)(Rumelhart *al.*, 1986a,b) is a network that is built upon the McGulloch and Pitts' model of neurons (McCulloch and Pitts, 1943) and the Perceptron (Rosenblatt, 1958). The Perceptron maps the input, generally binary, onto a binary valued output. The MLP uses this mapping to real valued outputs for binary or real valued inputs. The decision regions that could be formed by this network extend beyond the linear separable regions that are formed by the Perceptron. The nonlinearity inherent in the network enables it to perform better than the traditional linear methods (Lapedes and Farber, 1987). It has been observed that this input-output network mapping can be viewed as a hypersurface constructed in the input space (Lapedes and Farber, 1988). A surface interpolation method, called the radial basis functions, has been cast into a network whose architecture is similar to that of MLP (Broomhead and Lowe, 1988). Other surface interpolation methods, for example, the multivariate adaptive regression splines (MARS) (Friedman, 1991) and B-Splines (Lane *al.*, 1989), have also found their way into new forms of networks. Another view presented in Lippmann (1987) and Lapedes and Farber (1988) is that the network provides an approximation to an underlying function. This has resulted in applying polynomial approximation methods to neural networks, such as the Sigma - Pi units (Rumelhart *al.*, 1986a,b), the Volterra polynomial network (Rayner and Lynch, 1989) and the orthogonal network (Qian *al.*, 1990). Recently the application of wavelet transforms to neural networks (Pati and Krishnaprasad, 1990) has also derived its inspiration from function approximation.

While these networks may have little relationship to the biological neural networks, it has become common in the neural network area to refer to them as neural networks. These networks share one important characteristic; they are able to approximate any continuous mapping to a sufficient accuracy if they have resources to do so (Friedman, 1991; Stinchcombe and White, 1989).

As its name implies, a neural network is a network of simple processing elements called neurons connected to each other via links. The architecture of the network and the functionality of the neurons determine the response of the network to an input pattern. The network does no more than provide

an input-output mapping. Thus, a simple mathematical model can represent these networks. Now, let us investigate the neural network architectures and their functional representation by considering the multilayer network, which laid the foundation for the development of many other classes of feed-forward networks.

A neuron is an information-processing unit that is fundamental to the operation of a neural network. The model of a neuron is illustrated in Figure 11.1 (Haykin, 1994). There are three basic elements in the neuron model: connecting links, an adder and an activation function. Each of the connecting links is characterised by a weight or strength of its own. The adder sums the input signals weighted by the respective connecting link of the neuron. The activation function limits the amplitude of the output of a neuron, which is also referred to in the literature as a squashing function, in that it limits the permissible amplitude range of the output signal to some finite value. In mathematical terms, a neuron may be described by the following pair of equations:

$$v = \sum_{k=1}^{n} w_k u_k , \qquad y = \varphi(v) \tag{11.2.1}$$

where u_k is the input signal, w_k the weight of the neuron, v the linear combiner link, $\varphi(.)$ the activation function and y the output signal of the neuron.

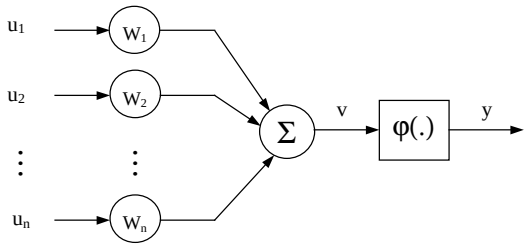

Figure 11.1: Model of a neuron

The multilayer network has a input layer, one or several hidden layers and an output layer. Each layer consists of neurons with each neuron in a layer connected to neurons in the layer below. This network has a feed-forward architecture which is shown in the Figure 11.2. The number of input neurons defines the dimensionality of the input space being mapped by the network and the number of output neurons as well as the dimensionality of the output space into which the input is mapped. Let the input to the network be denoted by u and the output by y. The neural network maps an input pattern to an output pattern, described by

$$f : u \to y \tag{11.2.2}$$

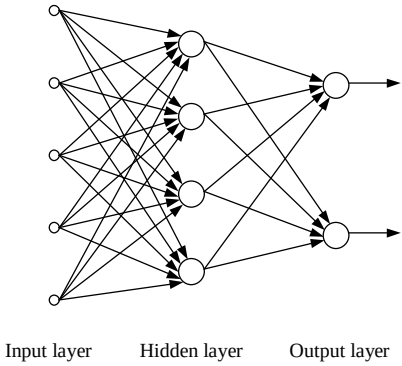

Figure 11.2: Architecture of a multilayer network

In a feedforward neural network, the overall mapping is achieved via intermediate mappings from one layer to another. These intermediate mappings depend on two factors. The first is the connection mapping that transforms the output of the lower layer neurons to an input to the neuron of interest and the second is the activation function of the neuron itself.

11.3 Gaussian Radial Basis Function Networks

Radial basis functions (RBF) have been introduced as a technique for multivariable interpolation (Powell, 1987). Broomhead and Lowe demonstrated that these functions can be cast into an architecture similar to that of the multilayer network, and hence named as the RBF network (Broomhead and Lowe, 1988).

In the RBF network that is a single hidden layer network, its input to the hidden layer connection transforms the input into a distance from a point in the input space, unlike in the MLP, where it is transformed into a distance from a hyperplane in the input space. However, it has been seen from multilayer networks that the hidden neurons can be viewed as constructing basis functions which are then combined to form the overall mapping. For the RBF network, the basis function constructed at the k-th hidden neuron is given by

$$\varphi_k(u) = g(\| u - d_k \|) \tag{11.3.1}$$

where $\| . \|$ is a distance measure, u the input vector, d_k the unit centre in the input space and $g(.)$ a nonlinear function. The basis functions are radially symmetric with the centre on d_k in the input space, hence they are named as radial basis functions. Some examples of nonlinear functions used as a radial basis function $g(.)$ are the following

$$g(r) = \sqrt{r^2 + c^2} \quad (\text{multi} - \text{quadratic}) \tag{11.3.2}$$

$$g(r) = r^2 \ln(r) \qquad \text{(thin plate splines)} \tag{11.3.3}$$

$$g(r) = \exp\left(\frac{r^2}{\sigma^2}\right) \qquad \text{(Gaussian)} \tag{11.3.4}$$

The radial basis function network with Gaussian hidden neurons is named as the Gaussian RBF network, also referred to as the network of localised receptive fields by Moody and Darken, who were inspired by the biological neurons in the visual cortex (Moody and Darken, 1989). The Gaussian RBF network is related to a variety of different methods (Niranjan and Fallside, 1988), particularly, Parzen window density estimation which is the same as kernel density estimation with a Gaussian kernel, potential functions method for pattern classification, and maximum likelihood Gaussian classifiers, which all can be described by a Gaussian RBF network formalism.

Following (11.3.1) and (11.3.4), the Gaussian RBF network can be described in a more general form. Instead of using the simple Euclidean distance between an input and a unit centre as in the usual formalism, a weighted distance scheme is used as the following:

$$\| u - d_k \| = (u - d_k)^T C_k (u - d_k) \tag{11.3.5}$$

where C_k is a weighting matrix of the k-th basis function whose centre is d_k. The effect of the weighting matrix is to transform the equidistant lines from being hyperspherical to hyperellipsoidal. Thus, a Gaussian RBF is given by

$$g_k(u, d, C) = \exp\{(u - d_k)^T C_k (u - d_k)\} \tag{11.3.6}$$

where d and C represent the centres and the weighting matrices. The Gaussian RBF network mapping is given by

$$f(u, p) = \sum_{k=1}^{n} w_k g_k(u, d, C) \tag{11.3.7}$$

where $p = \{w, d, C\}$. Clearly, the Gaussian RBF network is determined by the set of parameters $\{w_k, d_k, C_k\}$. To produce a mapping using this network, one can estimate all of these parameters or alternatively, provide a scheme to choose the widths C_k and the centres d_k of the Gaussian and adapt only the weights w_k. Adapting only the weights is much easier and more popular, since the problem of estimation is then linear.

11.4 Nonlinear Modelling with Neural Networks

The modelling of nonlinear systems has been posed as the problem of selecting an approximate nonlinear function between the inputs and the outputs of the systems. For a single-input single-output system, it can be expressed by

the NARMAX model (nonlinear auto-regressive moving average model with exogenous input, Leontaritis and Billings, 1985), that is

$$y(t) = f(y(t-1), ..., y(t-n_y), u(t-1), ..., u(t-n_u) + e(t) \qquad (11.4.1)$$

where $f(.)$ is an unknown nonlinear function, y is the output, u is the control input and e is the noise, respectively, n_y, n_u, n_e are the corresponding maximum delays. It is assumed that the noise $e(t)$ is a white noise. For the colour noise case, the modelling of the system using neural networks below needs some slight modifications, as suggested in Nerrand *et al.* (1994).

The nonlinear function $f(.)$ in the above NARMAX model can be approximated by a single-layer neural network, *i.e.*, a linear combination of a set of basis functions (Billings and Chen, 1992; Liu *et al.*, 1998)

$$f^*(x, p) = \sum_{k=1}^{N} w_k f_k(x, d_k) \qquad (11.4.2)$$

where

$$x = [y(t-1), ..., y(t-n_y), u(t-1), ..., u(t-n_u)] \qquad (11.4.3)$$

$f_k(x, d_k)$ $(k = 1, 2, ..., N)$ is the basis function and p is the parameter vector containing the weights w_k and the basis function parameter vectors d_k. If the basis functions $f_k(x, d_k)$ do not have the parameters d_k, then it is denoted by $f_k(x)$.

Radial basis functions were introduced as a technique for multivariable interpolation (Powell, 1987), which can be cast into an architecture similar to that of the multilayer Perceptron (Broomhead and Lowe, 1988). Radial basis function networks provide an alternative to the traditional neural network architectures and have good approximation properties. One of the commonly used radial basis function networks is the Gaussian radial basis function (GRBF) neural network, also called the localised receptive field network (Moody and Darken, 1989). The nonlinear function approximated by the GRBF network is expressed by

$$f^*(x, p) = \sum_{k=1}^{N} w_k \exp(-(x - d_k)^T C_k (x - d_k)) \qquad (11.4.4)$$

where C_k is the weighting matrix of the kth basis function, and p is the parameter vector containing the weights w_k, the centres d_k and the weighting matrix C_k $(k = 1, 2, ..., N)$.

11.5 Modelling Selection by Genetic Algorithms

Many different techniques are available for optimising the design space associated with various systems. In recent years, the direct-search techniques,

which are problem-independent, have been proposed as a possible solution for the difficulties associated with the traditional techniques. One direct-search method is the genetic algorithm (Goldberg, 1989). Genetic algorithms are search procedures which emulate the natural genetics. They are different from traditional search methods encountered in engineering optimisation. Recently, genetic algorithms have been applied to control system design (see, *e.g.*, Davis, 1991; Patton and Liu, 1994). GAs have also been successfully used with neural networks to determine the network parameters (Schaffer *et al.*, 1990; Whitehead and Choate, 1994) and with NARMAX models (Fonseca *et al.*, 1993). Here, the GA approach is applied to the model selection and identification of nonlinear systems using multiobjective criteria as the basis for selection.

The model selection can be seen as a subset selection problem. For the model represented by the GRBF network, the maximum number of the model terms is given by N, the number of the Gaussian functions, and there are 2^N possible models for selection and also N possible radial basis functions with their centres d_k. For the sake of simplicity, it assumes that $C_k = I$. Thus a chromosome representation in genetic algorithms consists of an N-bit binary model code c and N real number basis function centres d_k $(k = 1, 2, ..., N)$, *i.e.*,

$$[c, d_1^T, d_2^T, \ldots, d_N^T] \tag{11.5.1}$$

For example, if $N = 5$, $x \in \mathcal{R}^2$ and the chromosome

$$[\,[0 \quad 1 \quad 0 \quad 0 \quad 1\,], [d_{11}, d_{12}], \ldots, [d_{51}, d_{52}]] \tag{11.5.2}$$

then the model is given by

$$f^*(x, p) = w_2 \exp(-\sum_{j=1}^{2}(x_j - d_{2j})^2) + w_5 \exp(-\sum_{j=1}^{2}(x_j - d_{5j})^2) \tag{11.5.3}$$

It is evident from the above that only the basis functions corresponding to the non-zero bits of the binary model code c are included in the selected model. Given a parent set of binary model codes and basis function parameter vectors, a model satisfying a set of performance criteria is sought by the numerical algorithm given later.

11.6 Multiobjective Identification Criteria

This section presents multiobjective performance criteria for nonlinear model selection and identification. Let us define the following performance functions:

$$\phi_1(p) = \|f(x) - f^*(x, p)\|_2 \tag{11.6.1}$$

$$\phi_2(p) = \|f(x) - f^*(x, p)\|_\infty \tag{11.6.2}$$

$$\phi_3(p) = \sigma(c) \tag{11.6.3}$$

where $\|.\|_2$ and $\|.\|_\infty$ are the L_2- and L_∞-norms of the function (.), $\sigma(c)$ is the number of the non-zero elements in the binary model code c.

For model selection and identification of nonlinear systems, there are good reasons for giving attention to the performance functions $\phi_i(p)$ $(i = 1, 2, 3)$. The practical reasons for considering the performance function $\phi_1(p)$ is even stronger than the other performance functions $\phi_2(p)$ and $\phi_3(p)$. Statistical considerations show that it is the most appropriate choice for data fitting when errors in the data have a normal distribution. Often the performance function $\phi_1(p)$ is preferred because it is known that the best approximation calculation is straightforward to solve. The performance function $\phi_2(p)$ provides the foundation of much of approximation theory. It shows that when this is small, the performance function $\phi_1(p)$ is small also. But the converse statement may not be true. A practical reason for using the performance function $\phi_2(p)$ is based on the following. In practice, an unknown complicated nonlinear function is often estimated by one that is easy to calculate. Then it is usually necessary to ensure that the greatest value of the error function is less than a fixed amount, which is just the required accuracy of the approximation. The performance function $\phi_3(p)$ is used as a measure of the model complexity. A smaller performance function $\phi_3(p)$ indicates a simpler model in terms of the number of unknown parameters used. Under similar performances in $\phi_1(p)$ and $\phi_2(p)$ by two models, the simpler model is statistically likely to be a better model (Geman *et al.*, 1992).

In order to give a feel for the usefulness of the multiobjective approach as opposed to single-objective design techniques, let us consider the minimisation of the cost functions $\phi_i(p)$ $(i = 1, 2, 3)$. Let the minimum value of ϕ_i be given by $\phi_i^\star$, for $i = 1, 2, 3$, respectively. For these optimal values $\phi_i^\star$ there exist corresponding values given by $\phi_j[\phi_i^\star]$ $(j \neq i, j = 1, 2, 3)$, for $i = 1, 2, 3$, respectively, and the following relations hold:

$$\min\{\phi_1[\phi_2^\star], \phi_1[\phi_3^\star]\} \geq \phi_1^\star \tag{11.6.4}$$

$$\min\{\phi_2[\phi_1^\star], \phi_2[\phi_3^\star]\} \geq \phi_2^\star \tag{11.6.5}$$

$$\min\{\phi_3[\phi_1^\star], \phi_3[\phi_2^\star]\} \geq \phi_3^\star \tag{11.6.6}$$

If one of the performance functions ϕ_i $(i = 1, 2, 3)$ is minimised individually (single-objective approach), then unacceptably large values may result for other performance functions ϕ_j $(j \neq i,\ j = 1, 2, 3)$. Generally, there does not exist a solution for all performance functions $\phi_i(p)$ for $i = 1, 2, 3$ to be minimised by the same parameter vector p.

Following the method of inequalities (Zakian and Al-Naib, 1973), we reformulate the optimisation into a multiobjective problem as (Liu and Kadirkamanathan, 1999)

$$\phi_i(p) \leq \varepsilon_i, \quad \text{for } \ i = 1, 2, 3 \tag{11.6.7}$$

where the positive real number ε_i represents the numerical bound on the performance function $\phi_i(p)$ and is determined by the designer. Generally

speaking, the number ε_i is chosen to be a reasonable value corresponding to the performance function ϕ_i according to the requirements of the practical system. For example, ε_1 should be chosen between the minimum of ϕ_1 and the practical tolerable value on ϕ_1. The minimum of ϕ_1 can be determined by the least squares algorithm. The practical tolerable value means if ϕ_1 is greater than it, the modelling result cannot be accepted.

11.7 Multiobjective Identification Algorithm

With three objectives (or cost functions) for model selection and identification, the numerical algorithm is not a straightforward optimisation algorithm, such as for the least squares algorithm. This section introduces a numerical algorithm which uses genetic algorithm approaches and the method of inequalities to obtain a numerical solution satisfying the performance criteria.

Now, let us normalise the multiobjective performance functions as the following

$$\psi_i(p) = \frac{\phi_i(p)}{\varepsilon_i} \tag{11.7.1}$$

Let Γ_i be the set of parameter vectors p for which the i-th performance criterion is satisfied:

$$\Gamma_i = \{p : \psi_i(p) \leq 1\} \tag{11.7.2}$$

Then the admissible or feasible set of parameter vectors for which all the performance criteria hold is the intersection

$$\Gamma = \Gamma_1 \cap \Gamma_2 \cap \Gamma_3 \tag{11.7.3}$$

Clearly, p is an admissible parameter vector if and only if

$$\max\{\psi_1(p), \psi_2(p), \psi_3(p)\} \leq 1 \tag{11.7.4}$$

which shows that the search for an admissible p can be pursued by optimisation, in particular by solving

$$\min_p\{\max\{\psi_1(p), \psi_2(p), \psi_3(p)\}\} \tag{11.7.5}$$

subject to (11.7.4).

Following the boundary moving method (Zakian and Al-Naib, 1973), the optimisation is carried out using iterative schemes. Now, let p^q be the value of the parameter vector at the q-th iteration step in optimisation, and define

$$\Gamma_i^q = \{p : \psi_i(p) \leq \Delta^q\}, \quad \text{for} \quad i = 1, 2, 3 \tag{11.7.6}$$

where

$$\Delta^q = \max\{\psi_i(p^q)\} \tag{11.7.7}$$

and also define

$$\Gamma^q = \Gamma_1^q \cap \Gamma_2^q \cap \Gamma_3^q \tag{11.7.8}$$

Γ^q is the qth set of parameter vectors for which all performance functions satisfy

$$\psi_i(p) \leq \Delta^q, \quad \text{for} \quad i = 1, 2, 3 \tag{11.7.9}$$

It is clear that Γ^q contains both p^q and the admissible set Γ. If we find a new parameter vector $\bar{p}^q$, such that

$$\bar{\Delta}^q < \Delta^q \tag{11.7.10}$$

where $\bar{\Delta}^q$ is defined similarly to Δ^q, then we accept $\bar{p}^q$ as the next value of the parameter vector. Then, we set $p^{q+1} = \bar{p}^q$. We then have

$$\psi_i(p^{q+1}) \leq \psi_i(p^q), \quad \text{for} \quad i = 1, 2, 3 \tag{11.7.11}$$

and

$$\Gamma \subset \Gamma^{q+1} \subset \Gamma^q \tag{11.7.12}$$

so that the boundary of the set in which the parameters are located has been moved towards the admissible set. The process of finding the optimisation solution is terminated when Δ^q cannot be reduced any further. But the process of finding an admissible parameter vector p stops when

$$\Delta^q \leq 1 \tag{11.7.13}$$

i.e., when the boundaries of Γ^q have converged to the boundaries of Γ. Equation (11.7.13) is always achievable if ε_i is properly set, for $i = 1, 2, 3$. On the other hand, if the Δ^q persists in being larger than 1, this may be taken as an indication that the performance criteria may be inconsistent, whilst their magnitude gives some measure of how closely it is possible to approach the objectives. In this case, some of the parameters ε_i need to be increased. Generally speaking, the parameter ε_i corresponding to the largest normalised performance function $\psi_i(p^q)$ should be considered to increase first and then that corresponding to the second largest one and so on. This means that some of the performance criteria should be relaxed until they are satisfied. From a practical viewpoint, the approximate optimal solution is also useful if the optimal solution is not achievable. Genetic algorithms have been used in multiobjective optimisation and have provided better results over conventional search methods (Davis, 1991; Hajela and Lin, 1992; Schaffer, 1985). Here, we combine genetic algorithms with that of least squares in deriving the estimation algorithm. The steps of the identification algorithm to be executed for the GA implementation are as follows:

Algorithm 11.1

S1 Each chromosome in the population consists of an N-bit binary model code c and a real number basis function parameter vector D, where N is the number of the basis functions for the nonlinear model selection. So, for the GRBF network the vector D contains all basis function centres d_k $(k = 1, 2, ..., N)$, *i.e.*, $D = [d_1^T, d_2^T, ..., d_N^T]$

S2 The M chromosomes $[c, D]$ for the initial population are randomly generated, where M is the population size and is often chosen to be an odd number.

S3 Given the j-th binary model code c_j and basis function parameter vector D_j, then the structure of the j-th nonlinear model is known. Using the least squares algorithm, the j-th weight vector w_j can be computed easily, based on the data of the vector x, the binary model code c_j and the basis function parameter vector D_j. Then evaluate the normalised performance functions $\psi_i(\mathbf{s}_j)$ $(i = 1, 2, 3)$, where $s_j = [w_j, c_j, D_j]$, and

$$\Delta_j = \max_{i=1,2,3} \psi_i(\mathbf{s}_j) \tag{11.7.14}$$

These above computations are completed for all M sets of chromosomes, *i.e.* $j = 1, 2, ..., M$.

S4 According to the fitness of the performance functions for each chromosome, delete the $(M-1)/2$ weaker members of the population and reorder the chromosomes. The fitness of the performance functions is measured by

$$F_j = \frac{1}{\Delta_j}, \quad \text{for} \quad j = 1, 2, \ldots, M \tag{11.7.15}$$

S5 Offspring binary model codes are produced from two parent binary model codes so that their first half elements are preserved. The second half elements in each parent are exchanged. The average crossover operator is used to produce offspring basis function parameter vectors. The average crossover function is defined as

$$\frac{D_{j+1} + D_j}{2}, \quad \text{for} \quad j = 1, 2, \ldots, \frac{M-1}{2} \tag{11.7.16}$$

Then the $(M-1)/2$ offsprings are produced.

S6 A mutation operator, called creep (Davis, 1991), is used. For the binary model codes, it randomly replaces one bit in each offspring binary model code with a random number 1 or 0. For the offspring of the basis function

parameter vectors, the mutation operation is defined as

$$D_j + \beta\xi_j, \quad \text{for } j = 1, 2, \ldots, \frac{M-1}{2} \tag{11.7.17}$$

where β is the maximum to be altered and $\xi_j \in [-1, 1]$ is a random variable with zero mean.

S7 The elitist strategy copies the best chromosome into the succeeding generation. It prevents the best chromosome from being lost in the next generation. It may increase the speed of domination of a population by a super individual, but on balance it appears to improve genetic algorithm performance. The best chromosome is defined as one satisfying

$$\Delta_b = \min_{j=1,2,\ldots,M} \{\Delta_j\}, \tag{11.7.18}$$

Thus, the best chromosome is one that has the smallest Δ_j, for $j = 1, 2, ..., M$.

S8 Add the $(M-1)/2$ new offsprings to the population which are generated in a random fashion. Actually, the new offsprings are formed by replacing randomly some elements of the best binary model code and mutating the best basis function parameter vector with a certain probability.

S9 Continue the cycle initiated in Step 3 until local convergence of the algorithm is achieved. This local convergence is defined as the population satisfying

$$\Delta_j - \Delta_b \leq \varepsilon \quad \text{for } j = 1, 2, \ldots, (M-1)/2 \tag{11.7.19}$$

where ε is a positive number. This implies that the difference between the chromosomes in the first half population and the best chromosome is small in the sense of their performance measurement Δ_j.

Take the best solution in the converged generation and place it in a second "initial generation". Generate the other $M-1$ chromosomes in this second initial generation at random and begin the cycle again until a satisfactory solution is obtained or Δ_b cannot be reduced any further. In addition, for mixed noise distribution, the least squares algorithm in Step 3 should be replaced by a more robust modified least squares algorithm as suggested in Chen and Jain (1994).

11.8 Examples

This section introduces two examples. The first one considers identification of a real system. The second one demonstrates approximation of a nonlinear

function with a mixed noise with different variance.

11.8.1 Example 1

We use the data generated by a large pilot scale liquid level nonlinear system with zero mean Gaussian input signal (Fonseca *et al.*, 1993). 1000 pairs of input-output data were collected. The first 500 pairs were used in the model selection and identification of the system, while the remaining 500 pairs for validation tests. The Gaussian radial basis function network were applied to select and identify the model of the system by the numerical algorithm developed in section 11.7.

The time lags n_y and n_u were obtained by a trial and error process based on estimation of several models. During the simulation, it was found that for the GRBF network, if n_y and n_u were greater than 2 the performance functions did not reduce significantly. The parameters for the algorithm are given in Table 11.1.

Table 11.1: Parameters

Parameter Name	GRBF Network
model term number N	10
chromosome length	50
variable vector x	$\begin{bmatrix} y(t-1) \\ y(t-2) \\ u(t-1) \\ u(t-2) \end{bmatrix}$
ε_1	1.5
ε_2	0.3
ε_3	7

Although the maximum number of the model terms is only 10 (*i.e.*, 1024 possible models for selection), the search dimension of the basis function centre parameters is 40 in real number space (*i.e.*, infinite possibilities for selection). After 700 generations the performance criteria are almost satisfied. At this stage, $\phi_1(p) = 1.5643$, $\phi_2(p) = 0.2511$, $\phi_3(p) = 5$. In order to obtain the better performance, the basis function parameter vector was searched for another 100 generations using the algorithm with the fixed number of the model terms, *i.e.*, let $\phi_3(p) = 5$ for this case. Finally, the performance functions are

$$\phi_1(p) = 1.2957, \quad \phi_2(p) = 0.1724, \quad \phi_3(p) = 5$$

The model represented by the GRBF network is

$$y(t) = \sum_{i=1}^{5} w_i \exp(-\sum_{j=1}^{2}(y(t-j) - d_{ij})^2 - \sum_{j=1}^{2}(u(t-j) - d_{ij})^2)$$

where

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} = \begin{bmatrix} -2.6363 \\ -1.2470 \\ -1.7695 \\ 0.9437 \\ -0.5341 \end{bmatrix}$$

$$\{d_{ij}\} = \begin{bmatrix} -2.1577 & -1.8855 & -0.8975 & -0.2841 \\ -1.2717 & -2.2730 & 0.3445 & 0.3315 \\ -0.6345 & -1.1223 & -1.1615 & -0.3666 \\ 0.7344 & 1.0223 & 0.5469 & 0.1989 \\ -1.2336 & -0.5928 & 0.3212 & 0.5754 \end{bmatrix}$$

The performance of the GRBF network is shown in Figures 11.3 – 11.5. Figure 11.3 shows the convergence of the performance functions with respect to generations. The measured and estimated outputs, and residual error of the system for the training data for the model identified via the GRBF network is shown in Figure 11.4. The measured and estimated outputs, and estimation error of the system for the validation test data for the model identified via the GRBF network is shown in Figure 11.5.

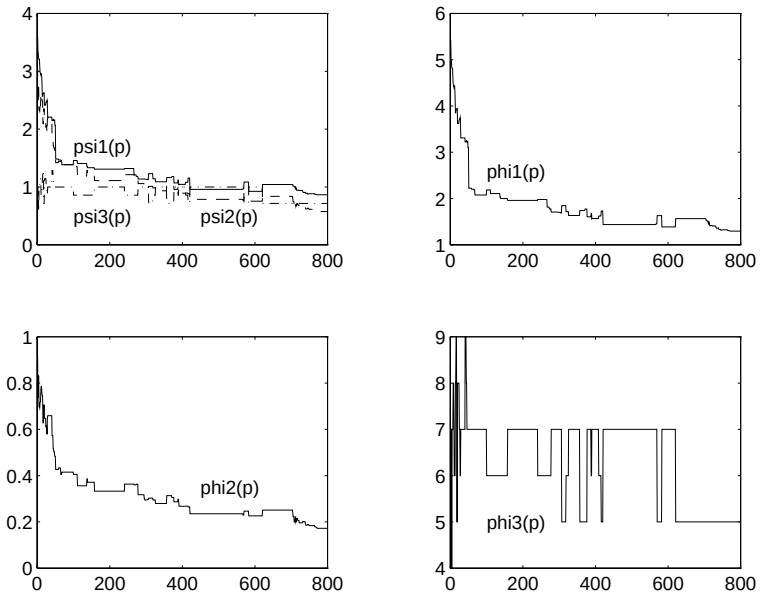

Figure 11.3: Convergence of the performance functions using GRBF network

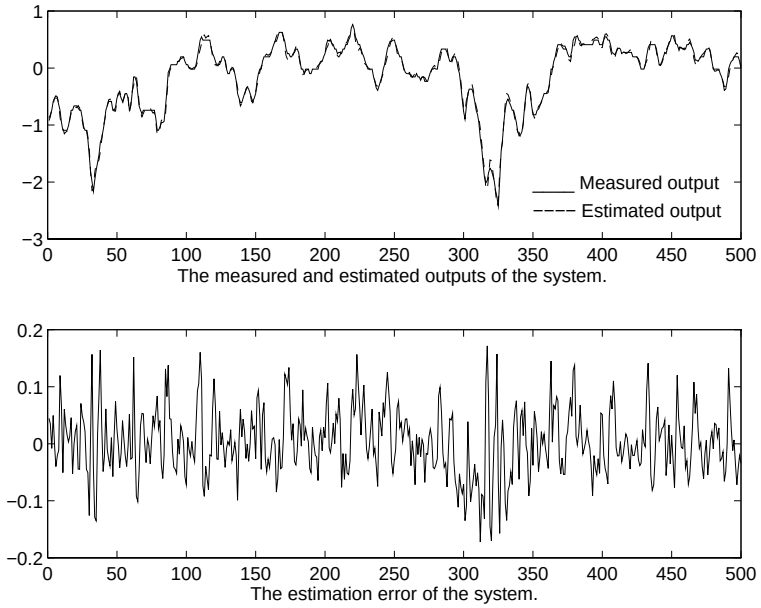

Figure 11.4: The training data results for the system using GRBF network.

In order to see the importance of the L_∞-norm measure of the accuracy bound of the approximation, the performance function $\phi_2(p)$ is not used in the next simulation. So, only two performance functions $\phi_1(p)$ and $\phi_3(p)$ are considered. Their required upper bounds ε_1 and ε_3 are still set to be 1.5 and 7. The simulation procedure is exactly the same as the above. The following performance is obtained:

$$\phi_1(p) = 1.2900, \quad \phi_3(p) = 4$$

The weight vector and centres of the network are

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 1.2394 \\ -2.4092 \\ -2.8293 \\ -2.5141 \end{bmatrix}$$

$$\{d_{ij}\} = \begin{bmatrix} 1.3219 & 0.4971 & 0.4451 & 0.0935 \\ -0.5826 & -2.1796 & 0.2636 & 0.5724 \\ -1.6041 & -0.0912 & -0.7477 & -0.0275 \\ -2.1362 & -1.9554 & -0.7189 & -0.2974 \end{bmatrix}$$

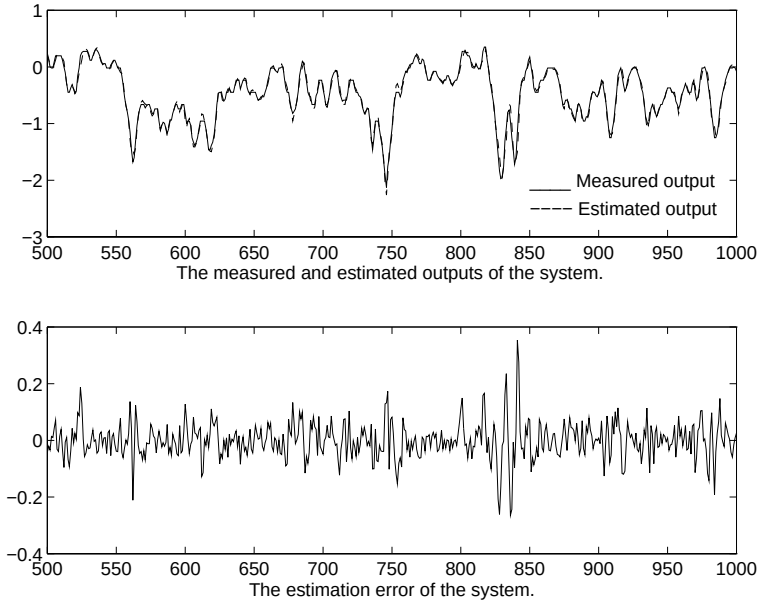

Figure 11.5: Validation results for the system using GRBF network

The simulation results are shown in Figures 11.6-11.8. It is clear from the results that although the performance functions $\phi_1(p)$ and $\phi_3(p)$ are reduced, the maximum difference $\phi_2(p)$ of the approximation for identification and validity test is much greater than the previous case. So, it shows that if the performance functions $\phi_1(p)$ and $\phi_3(p)$ are sacrificed somewhat, the performance function $\phi_2(p)$ is improved significantly.

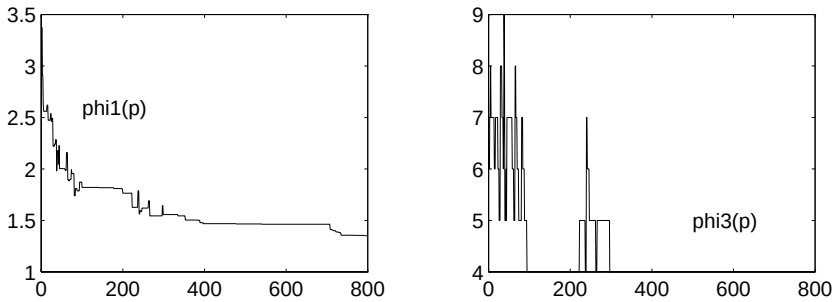

Figure 11.6: Convergence of the performance functions using GRBF network without ϕ_2

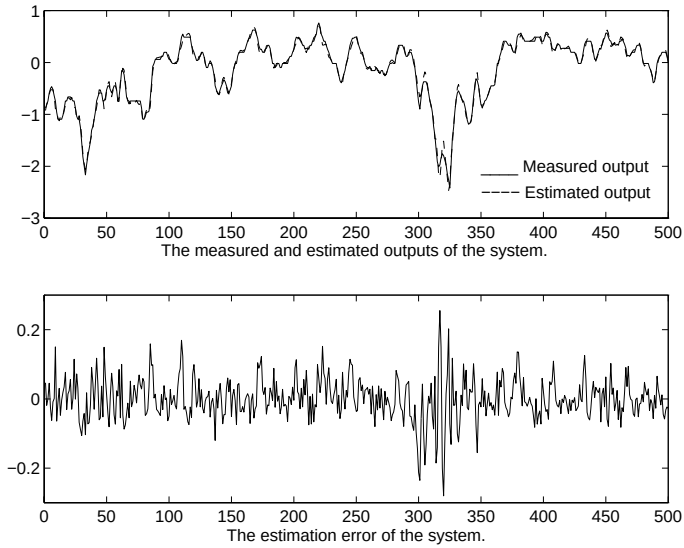

Figure 11.7: Training results for the GRBF network without ϕ_2

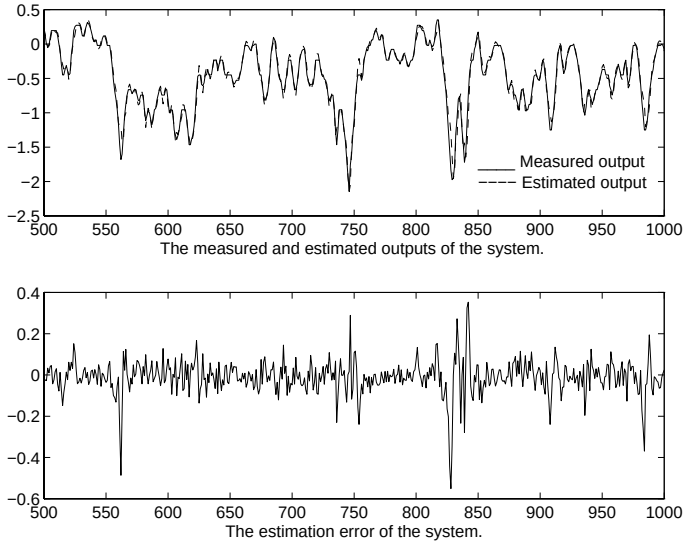

Figure 11.8: Validation results for the GRBF network without ϕ_2

11.8.2 Example 2

Consider the following underlying nonlinear function to be approximated.

$$f^*(x) = 1.1(1 - x + x^2)\exp(-0.5x^2)$$

where x is a variable. A random sampling of the interval $[-4, 4]$ is used in obtaining the 40 input-output data for approximation.

In order to see the effect of noise, the output of the function f to a given input x is given by

$$f(x) = f^*(x) + e$$

where e is a mixed noise. The noise consists of uniformly and normally distributed noises, *i.e.*,

$$e = \frac{1}{\sqrt{2}}(e_{U[0,\sigma]} + e_{D[0,\sigma]})$$

where $e_{U[0,\sigma]}$ is a zero mean uniform noise with finite variance σ and $e_{D[0,\sigma]}$ is a zero mean normal noise with finite variance σ. It is assumed that the uniform noise $e_{U[0,\sigma]}$ and the normal noise $e_{D[0,\sigma]}$ are uncorrelated. Thus, the mean and variance of the mixed noise e are zero and σ, respectively.

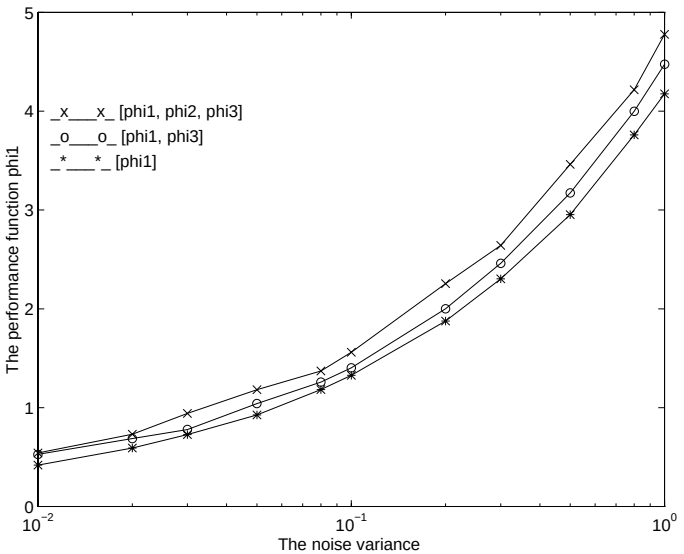

Figure 11.9: Performance function $\phi_1(p)$ against the noise variance σ

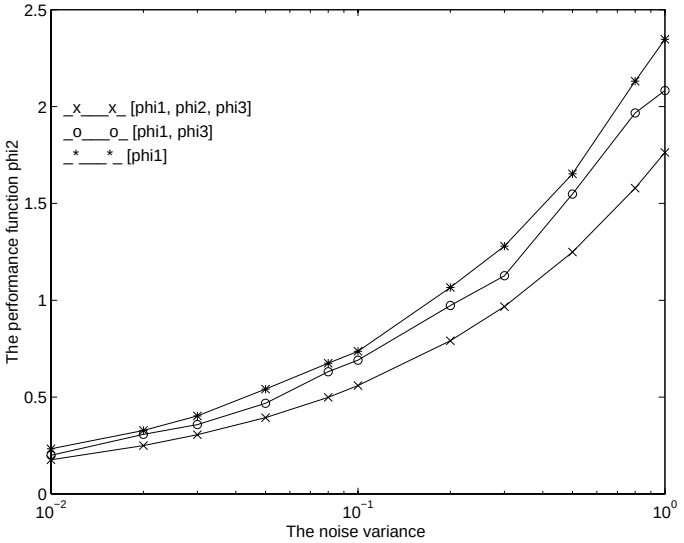

Figure 11.10: Performance function $\phi_2(p)$ against the noise variance σ

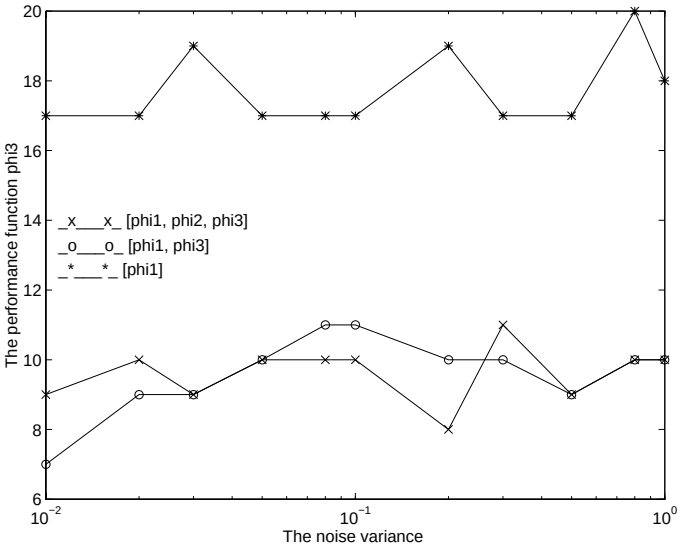

Figure 11.11: Performance function $\phi_3(p)$ against the noise variance σ

Here, the Gaussian radial basis function network was used to approximate the nonlinear function by the numerical algorithm developed in section 11.7. Three cases were considered in this simulation. The first used three performance functions $[\phi_1(p), \phi_2(p), \phi_3(p)]$ during approximation. The second considered two performance functions $[\phi_1(p), \phi_3(p)]$. The third used only one performance function $\phi_1(p)$. The effects of the mixed noise with different variance on the performance functions $\phi_1(p)$, $\phi_2(p)$ and $\phi_3(p)$ for the above three cases are illustrated in Figures 11.9-11.11, respectively. It can be seen from the simulation results that the performance of the approximation of the nonlinear function changes little at low level variance of noise and the multiobjective case using three performance criteria gives a good approximation even though three performance functions conflict each other.

11.9 Summary

This chapter has addressed the problems of model selection and identification of nonlinear systems using neural networks, genetic algorithms and multiobjective optimisation techniques. A set of performance functions that measure approximation accuracy and a model complexity measure are used as the multiobjective criteria in the identification task. The optimisation is carried out using genetic algorithms which select the nonlinear function units to arrive at the simplest model necessary for approximation, along with optimising the multiobjective criteria. The Gaussian radial basis function (GRBF) is subjected to the algorithm in the task of a liquid level nonlinear system identification. The model selection procedure results in the selection of the basis function centres for the GRBF model. The experimental results demonstrate the convergence of the developed algorithm and its ability to arrive at a simple model which approximates the nonlinear system well. The approach discussed in this chapter can also be extended in many ways, for example, to adaptively modify the numerical bounds on the performance functions. Furthermore, cross-validation techniques can be used to guide the optimisation and also in the adaptation of the bounds on the performance functions.

Chapter 12

Multiobjective Fault Diagnosis

12.1 Introduction

Fault diagnosis has become an important subject in modern process automation as it provides pre-requisites for fault tolerance, reliability or security, which constitute fundamental design features in complex engineering systems. The general procedure of fault diagnosis in dynamic systems, with the aid of analytical redundancy, consists of the following two steps: a) Generation of so-called residuals, *i.e.*, functions that carry information about faults. b) Decision on the occurrence of a fault and isolation of the faulty element, *i.e.*, localisation of the fault. In order to make the residual insensitive to modelling uncertainty and sensitive to sensor faults, a number of performance indices have been defined to achieve good fault diagnosis performance. Some of the performance indices are defined in the frequency domain to account for the fact that the effects of modelling uncertainty and faults occupy different frequency ranges. Robust fault diagnosis in the frequency domain has been attracting enormous attention.

This chapter considers the design of optimal residuals in order to diagnose incipient faults based on multiobjective optimisation. The residual is generated via an observer. To reduce false and missed alarm rates in fault diagnosis, a number of performance indices are introduced into the observer design. These performance indices are expressed in the frequency domain to take account of the frequency distributions of faults, noise and modelling uncertainty. All objectives then are reformulated into a set of inequality constraints on performance indices. A multiobjective fault diagnosis algorithm is thus used to search for an optimal solution to satisfy all the objectives expressed by a set of criteria. This algorithm is applied to fault diagnosis of a flight control system. The example and simulation results show that incipient

sensor faults can be detected reliably in the presence of modelling uncertainty.

12.2 Overview of Robust Fault Diagnosis

In order to ensure reliable operation of control systems, a fault diagnosis system is needed, including fault detection and isolation (FDI). Model-based FDI approaches have been of interest in aerospace and various other fields for many years, and have been demonstrated to be capable of detecting and isolating abrupt and hard faults very quickly and reliably. However, the detection of soft, slowly developing (or incipient) faults is not very straightforward without any consideration of robustness of FDI algorithms (Patton *et al.*, 1989; Frank, 1990; Gertler, 1991). In safety-critical systems such as aircraft, hard faults in some system components may not be tolerable and such faults must be detected before they actually occur. Hopefully, faults are detected during the maintenance stage; but, the situation is different for soft and/or incipient faults. Their effects on the system are very small and almost unnoticeable during their incipient stages. They may develop slowly to cause very serious effects on the system, although these incipient faults may be tolerable when they first appear. Hence, the most important issue of a reliable system operation is to detect and isolate incipient faults as early as possible. Early indication of incipient faults can give the operator enough information and time to take proper measures to prevent any serious consequence on the system.

Due to the inseparable mixture of fault effects and modelling uncertainty, the detection of incipient faults presents a challenge to model-based FDI techniques. All model-based methods use the model of the monitored system to produce residuals for FDI. If the system is described accurately by the mathematical model, FDI is very straightforward. In real complex systems, however, the modelling uncertainty that arises, for example, from process to noise, turbulence, parameter variations and modelling errors, is inevitable. This presents a challenge for the FDI algorithm design. Hard or sudden faults normally have a larger effect on diagnostic residuals than a modelling uncertainty effect. Thus, such faults can be detected by placing an appropriate threshold on the residual. On the other hand, an incipient fault has a lower effect, which sometimes is even lower than the response due to modelling uncertainty, so that thresholding cannot be directly used to detect and isolate incipient faults reliably. In order to solve this problem, robust approaches to FDI have been attracting research interest for many years (Patton *et al.*, 1989; Frank, 1990; Patton and Chen, 1992;). The principle of the robust approaches is to make the FDI algorithm sensitive to incipient faults and insensitive to modelling uncertainty. If an FDI scheme has this robust-sensitivity property, it will be called a robust FDI scheme. There is a trade-off between sensitivity and robustness, and hence this is an issue of prime concern.

A number of methods have been developed to design robust FDI systems.

The most commonly-used approach uses the disturbance de-coupling concept (Patton *et al.*, 1987; Frank and Wunnenberg, 1989; Gertler and Kunwer, 1993), *i.e.*, to de-couple the effect of exogenous disturbances on the residual. The main disadvantage is that the distribution of disturbances is required to facilitate de-coupling designs, although the disturbance itself does not need to be known. For most uncertain systems, the modelling uncertainty is expressed in terms of modelling errors. Hence, the disturbance de-coupling approach cannot be applied directly. There have been some studies on representing modelling errors as unknown disturbances with an approximate distribution matrix (Chen, 1995; Patton and Chen, 1993a). In this way, robust FDI is partially achievable, as has been illustrated by some successful applications. But, the robust solution is still not very satisfactory. The main idea of the disturbance de-coupling approach is to completely eliminate the disturbance effect from the residual. Generally, complete elimination of disturbance effects may not be possible, owing to the lack of design freedom and unavailable disturbance distribution matrices. The fault effect may also be undesirably decreased. So, an appropriate criterion for robust residual design should take account of both modelling errors and faults. There is also a trade-off between sensitivity to faults and robustness to modelling uncertainty. In fact, robust residual generation can be considered as a multiobjective optimisation problem, *i.e.* the maximisation of fault effects and the minimisation of uncertainty effects. This problem is tackled by minimising a ratio between disturbance effects and fault effects when the disturbance distribution matrix is known (Ding and Frank, 1989).

Although robust control design in the frequency domain has been attracting enormous attention, little research exists on the use of frequency domain techniques for robust FDI. Patton *et al.* (1987) discussed the possibility of using frequency distribution information to design fault diagnosis algorithms; but, they did not give further guidance as to how this could be achieved. Ding and Frank (1989) proposed an optimal observer design method for FDI in the frequency domain by assuming known disturbance distribution matrices. Viswanadham *et al.* (1987) and Ding and Frank (1990) studied the frequency domain residual generation method via factorisation of the system transfer matrix; however, the robustness issue is not their primary concern in the design. Recently, Frank and Ding (1993) and Qiu and Gertler (1993) made some important contributions in robust FDI design by using H^{∞}-optimisation. But, they only consider cases when the disturbance distribution matrix is known *a priori*, and this is the main drawback of their studies. Mangoubi *et al.* (1992) also applied the H_p technique to the design of a robust FDI algorithm; however, the fault effect has not been considered in their performance criterion.

More recently, a novel approach to designing optimal residuals for robust fault diagnosis has been proposed (Chen *et al.*, 1996). In this approach an observer is used to generate residuals. In order to make the residual insensitive to modelling uncertainty and sensitive to sensor faults, a number of perfor-

mance indices have been defined to achieve good fault diagnosis performance. Some performance indices are defined in the frequency domain to account for the fact that the effects of modelling uncertainty and faults occupy different frequency ranges. The disturbance distribution matrix is not required, although it can be included in the design, if it is available. Hence, this method can be applied to a wide range of problems. Instead of a single (or a mixed) performance index, multiple performance indices are introduced to achieve a good diagnosis. A numerical optimisation technique (easily implemented with a computer) is used for the robust residual design which avoids the use of complicated theoretical analysis. The multiobjective optimisation problem is solved by the method of inequalities. Hence, different diagnosis preferences (low missed alarm or low false alarm) can be achieved by adjusting inequality constraints. The numerical search is performed via a genetic algorithm, thus obviating the requirement for the calculation of cost function gradients and hence increasing the possibility of finding the global optimum. In this approach, frequency-dependent weighting factors are introduced in the performance indices (cost function) based on knowledge of the frequency band of the modelling uncertainty and faults. The approach has been applied to the detection of incipient sensor faults in a flight control system, which shows that the fault detection algorithm designed by the approach has detected incipient sensor faults very effectively.

12.3 Observer Based Fault Diagnosis

Consider the following continuous monitored system which is disturbed by an additive unknown input term:

$$\dot{x}(t) = Ax(t) + Bu(t) + R_1 f(t) + Ed(t) \tag{12.3.1}$$

$$y(t) = Cx(t) + Du(t) + R_2 f(t) \tag{12.3.2}$$

where $x(t) \in \mathcal{R}^n$ is the state vector, $y(t) \in \mathcal{R}^r$ is the output vector, $u(t) \in \mathcal{R}^m$ is the known input vector and $d(t) \in \mathcal{R}^q$ is the unknown input (or disturbance) vector, $f(t) \in \mathcal{R}^g$ represents the fault vector which is considered as an unknown time function. A, B, C, D and E are known matrices with appropriate dimensions. The matrices R_1 and R_2 are fault distribution matrices which are known when the designer has been told where in the system faults are to be detected and isolated. In addition, the matrix E is assumed to be full column rank.

The vector $d(t)$ is the disturbance vector which can also be used to represent modelling errors such as

$$d(t) = \Delta A x(t) + \Delta B u(t) \tag{12.3.3}$$

Note that this form of uncertainty representation is very general, as the distribution matrix is not required. The matrices R_1 and R_2 are fault distribution

matrices that represent the influence of faults on the system. They can be determined if one has defined which faults are to be diagnosed. For the two most common cases (sensor and actuator faults), these matrices are

$$R_1 = \begin{cases} 0 & \text{sensor faults} \\ B & \text{actuator faults} \end{cases} \tag{12.3.4}$$

$$R_2 = \begin{cases} I_m & \text{sensor faults} \\ D & \text{actuator faults} \end{cases} \tag{12.3.5}$$

The residual generator used here is based on a full-order observer. The basic idea is to estimate the system output from the measurements using an observer. The weighted output estimation error is then used as a residual. The flexibility in selecting the observer gain and the weighting matrix provides freedom to achieve good diagnosis performance. The residual generator is thus described as

$$\dot{\hat{x}}(t) = (A - KC)\hat{x}(t) + (B - KD)u(t) + Ky(t) \tag{12.3.6}$$

$$\hat{y}(t) = C\hat{x}(t) + Du(t) \tag{12.3.7}$$

$$r(t) = Q(y(t) - \hat{y}(t)) \tag{12.3.8}$$

where $r \in \mathcal{R}^p$ is the residual vector, and $\hat{x}$ and $\hat{y}$ are the state and output estimations. The matrix $Q \in \mathcal{R}^{p \times m}$ is the residual weighting factor which is static for most cases but can also be dynamic. Note that the residual is a linear transformation of the output estimation error. Hence, the residual dimension p cannot be larger than the output dimension m. This is because the linearly dependent extra residual components do not provide additional useful information in FDI.

When the residual generator represented by (12.3.8) is applied to the system described by (12.3.1), the state estimation error $e(t) = x(t) - \hat{x}(t)$ and the residual are governed by the following equations:

$$\dot{e}(t) = (A - KC)e(t) + Ed(t) + R_1 f(t) - KR_2 f(t) \tag{12.3.9}$$

$$r(t) = QCe(t) + QR_2 f(t) \tag{12.3.10}$$

The Laplace-transformed residual response to faults and disturbances is thus

$$\begin{aligned} r(s) \quad = \quad & QR_2 f(s) + QC(sI - A + KC)^{-1}(R_1 - KR_2)f(s) \\ & + QC(sI - A + KC)^{-1}E(d(s) + e(0)) \end{aligned} \tag{12.3.11}$$

where $e(0)$ is the initial value of the state estimation error.

One can see that the residual $r(t)$ and the state estimation error are not zero, even if no faults occur in the system. Indeed, it can be difficult to distinguish the effects of faults from the effects of disturbances acting on the system. The effects of disturbances obscure the performance of FDI and act as a source of false and missed alarms. Therefore, in order to minimise the false and missed alarm rates, one should design the residual generator such that the residual itself becomes de-coupled with respect to disturbances.

12.4 Multiple Objectives of Fault Diagnosis

Both faults and disturbances affect the residual, and discrimination between these two effects is difficult. To reduce false and missed alarm rates, the effect of faults on the residual should be maximised and the effect of disturbances on the residual should be minimised. One can maximise the effect of the faults by maximising the following performance index, in the required frequency range $[\omega_1, \omega_2]$:

$$\inf_{\omega\in[\omega_1,\omega_2]} \underline{\sigma}\{QR_2 + QC(j\omega I - A + KC)^{-1}(R_1 - KR_2)\} \tag{12.4.1}$$

This is equivalent to the minimisation of the performance index below.

$$\phi_1(K,Q) = \sup_{\omega\in[\omega_1,\omega_2]} \bar{\sigma}\{[QR_2 + QC(j\omega I - A + KC)^{-1}(R_1 - KR_2)]^{-1}\} \tag{12.4.2}$$

where $\underline{\sigma}\{.\}$ and $\bar{\sigma}\{.\}$ denote the minimal and maximal singular values of a matrix, respectively.

Similarly, one can minimise the effects of both disturbance and initial condition by minimising the following performance index:

$$\phi_2(K,Q) = \sup_{\omega\in[\omega_1,\omega_2]} \bar{\sigma}\{QC(j\omega I - A + KC)^{-1}\} \tag{12.4.3}$$

Besides faults and disturbances, noise in the system can also affect the residual. To illustrate this, assume that $\zeta(t)$ and $\eta(r)$ are input and sensor noise signals; the system equations in this case are

$$\dot{x}(t) = Ax(t) + Bu(t) + R_1 f(t) + \zeta(t) \tag{12.4.4}$$

$$y(t) = Cx(t) + Du(t) + R_2 f(t) + \eta(t) \tag{12.4.5}$$

It can be seen that the sensor noise, as well as faults acting through $R_2 f(t)$, affects the system at the same excitation point and hence affect the residual in the same way. To reduce the noise effect on the residual, the norm

$$\left\| Q - QC(j\omega I - A + KC)^{-1}K \right\|_\infty \tag{12.4.6}$$

should be minimised. This contradicts the requirement to maximise the effects of faults on the residual. Fortunately, the frequency ranges of the faults and noise are normally different. For an incipient fault signal, the fault information is contained within a low-frequency band as the fault develops slowly for incipient cases. However, the noise comprises mainly high-frequency signals. Based on these observations, the effects of noise and faults can be separated by using different frequency-dependent weighting penalties. In this case, the performance index $\phi_1(K,Q)$ is

$$\begin{aligned}\phi_1(K,Q) \quad = \quad & \sup_{\omega\in[\omega_1,\omega_2]} \bar{\sigma}\{W_1(j\omega)[QR_2 + QC(j\omega I - A + KC)^{-1} \times \\ & (R_1 - KR_2)]^{-1}\}\end{aligned} \tag{12.4.7}$$

To minimise the effect of noise on the residual, a new performance index is introduced as

$$\phi_3(K,Q) = \sup_{\omega\in[\omega_1,\omega_2]} \bar{\sigma}\{W_3(j\omega)Q(I - C(j\omega I - A + KC)^{-1}K)\} \quad (12.4.8)$$

In order to maximise the effects of faults at low frequencies and minimise the noise effect at high frequencies, the frequency-dependent weighting factor $W_1(j\omega)$ should have a large magnitude in the low-frequency range and a small magnitude at high frequencies. The frequency effect of $W_3(j\omega)$ should be opposite to that of $W_1(j\omega)$ and can be chosen as $W_3(j\omega) = W_1^{-1}(j\omega)$. The disturbance (or modelling error) and input noise affect the residual in the same way. As both effects should be minimised, the performance index $\phi_2(K,Q)$ does not necessarily need to be weighted. However, modelling uncertainty and input noise effects may be more serious in one or more frequency bands. The performance index should reflect this fact, and hence a frequency-dependent weighting factor must also be placed on $\phi_2(K,Q)$, in some situations, that is

$$\phi_2(K,Q) = \sup_{\omega\in[\omega_1,\omega_2]} \bar{\sigma}\{W_2(j\omega)QC(j\omega I - A + KC)^{-1}\} \quad (12.4.9)$$

Now, consider the steady-state value of the residual

$$\begin{aligned} r(\infty) &= QR_2f(\infty) + QC(A - KC)^{-1}(KR_2 - R_1)f(\infty) \\ &\quad -(A - KC)^{-1}d(\infty) \end{aligned} \quad (12.4.10)$$

After the transient period, the residual steady-state value plays an important role in FDI. Ideally, it should reconstruct the fault signal. The disturbance effects on residual can be minimised by minimising the following performance index:

$$\phi_4(K) = \left\| (A - KC)^{-1} \right\|_\infty \quad (12.4.11)$$

When $\phi_4(K)$ has been minimised, the matrix K is very large and the norm $\left\| (A - KC)^{-1}K \right\|_\infty$ approaches a constant value. This means that the fault effect on the residual has not been changed by reducing the disturbance effect. This is what is required for good FDI performance.

12.5 Disturbance Distribution and Fault Isolation

In this section, the disturbance distribution is considered. The information on the disturbance distribution can also be incorporated into performance indices, if it is available. Assume the disturbance distribution matrix is known, *i.e.*

$$d(t) = Ed'(t) \quad (12.5.1)$$

where E is a known matrix and $d'(t)$ is an unknown vector. In this case, the performance index $\phi_2(K,Q)$ can be modified as

$$\phi_2(K,Q) = \sup_{\omega\in[\omega_1,\omega_2]} \bar{\sigma}\{W_2(j\omega)QC(j\omega I - A + KC)^{-1}E\} \qquad (12.5.2)$$

To isolate faults, a structured residual set should be generated (Chen, 1995; Gertler, 1991; Patton and Chen 1993b). The word 'structured' here signifies the sensitivity and insensitivity relations that any residual will have, *i.e.* whether it is designed to be sensitive to a group of faults while insensitive to another group of faults. The faults contained in the vector $f(t)$ can be divided into two groups, $f^1(t)$ and $f^2(t)$, and the system equation in this case is

$$\dot{x}(t) = Ax(t) + Bu(t) + R_1^1 f^1(t) + R_1^2 f^2(t) + d(t) \qquad (12.5.3)$$
$$y(t) = Cx(t) + Du(t) + R_2^1 f^1(t) + R_2^2 f^2(t) \qquad (12.5.4)$$

If the residual is to be designed to be sensitive to $f^1(t)$ and insensitive to $f^2(t)$, the performance index $\phi_1(K,Q)$ should be modified as

$$\phi_1(K,Q) = \sup_{\omega\in[\omega_1,\omega_2]} \bar{\sigma}\{W_1(j\omega)[Q R_2^1 + QC(j\omega I - A + KC)^{-1} \times (R_1^1 - KR_2^1)]^{-1}\} \qquad (12.5.5)$$

In addition to the four performance indices defined, a new performance index $\phi_5(K,Q)$ which is to be minimised should be introduced to make the residual insensitive to $f^2(t)$.

$$\phi_5(K,Q) = \sup_{\omega\in[\omega_1,\omega_2]} \bar{\sigma}\{W_5(j\omega)[Q R_2^2 + QC(j\omega I - A + KC)^{-1} \times (R_1^2 - KR_2^2)]^{-1}\} \qquad (12.5.6)$$

For the case where only sensor faults are to be isolated, the design problem is easier to solve. This is because, if the residual is to be sensitive to a group of sensor faults, only the measurements from this set of sensors will be used in the residual generation (Chen, 1995).

12.6 Parameterisation of Fault Diagnosis

Five performance indices $\phi_i(K,Q)$, for $i = 1,2,...,5$ have been defined. To achieve robust fault detection (in terms of minimising false and missed alarm rates), one needs to solve a multiobjective optimisation problem. One of the parameter sets to be designed is the observer gain matrix K which must guarantee stability of the observer. This leads to a constrained optimisation problem which is difficult to solve.

Within the context of control system design, the stability constraint is normally changed to the assignment of eigenvalues in the left-hand side of the

complex plane. The observer design is a dual problem of the controller design and all techniques in control design can be applied. Here an eigenstructure assignment method is chosen to give the parameterisation of the gain matrix K (Liu and Patton, 1998c). It assumes that the eigenvalues of the matrix $(A^T - C^T K^T)$ are distinct and are not the same as ones of the matrix A. According to the definition of eigenvectors and eigenvalues, there is the following relation:

$$(A^T - C^T K^T) R_i = \lambda_i R_i \tag{12.6.1}$$

where R_i is the i-th right eigenvector corresponding to the i-th eigenvalue λ_i of the matrix $(A^T - C^T K^T)$. Let the parameter vector be defined as

$$W_i = K^T R_i \tag{12.6.2}$$

Then

$$R_i = (A^T - \lambda_i I)^{-1} C^T W_i \tag{12.6.3}$$

The gain matrix can then be calculated by

$$\begin{aligned} K^T \quad = \quad & W[(A^T - \lambda_1 I)^{-1} C^T W_1, \ (A^T - \lambda_2 I)^{-1} C^T W_2, \ \ldots \\ & \ldots \ (A^T - \lambda_n I)^{-1} C^T W_n] \end{aligned} \tag{12.6.4}$$

where $W = [W_1, W_2, ..., W_n] \in \mathcal{R}^{m \times n}$ and the vector $W_i \in \mathcal{R}^m$, for $i = 1, 2, ..., n$ can be chosen freely, the eigenvalues λ_i, for $i = 1, 2, ..., n$ may need to be given by the designer before the design procedure.

In practice, the eigenvalues do not need to be assigned to a specific point in the complex plane. However, we do need to assign eigenvalues in predefined regions to meet stability and response requirements, *i.e.* $\lambda_i \in [L_i, U_i]$. This increases the design freedom, but results in eigenvalue constraints. To remove these constraints, we introduce a simple transformation (Liu and Patton, 1998):

$$\lambda_i = L_i + (U_i - L_i) \sin^2(z_i) \tag{12.6.5}$$

where $z_i \in \mathcal{R}$ can be freely chosen. Although the use of only real eigenvalues is discussed here, the method can also be extended to the complex eigenvalue case. The parameterisation of K for complex conjugate eigenvalues can be found in Liu and Patton (1998a).

Now constrained performance indices $\phi_i(K, Q)$, for $i = 1, 2, ..., 5$ have been transformed into unconstrained performance indices $\phi_i(Z, W, Q)$, for $i = 1, 2, ..., 5$, where W, Q and $Z = [z_1, z_2, ..., z_n]$ can be chosen freely. Thus, the design of the observer based fault diagnosis becomes the determination of the parameter matrices W and Q and vector Z.

12.7 Multiobjective Fault Diagnosis

The use of multiobjective optimisation is very common in engineering design problems. Generally, there does not exist a solution which minimises all performance indices. A parameter which minimises a particular performance index may let other performance indices become very large and unacceptable. Hence, some compromises must be made in the design (Vincent and Grantham, 1981). An approach to solving the multiobjective optimisation problem in control system design is the method of inequalities, proposed by Zakian and Al-Naib (1973) and later combined with the genetic algorithm by Patton and Liu (1994). The main philosophy behind this approach is to replace the minimisation of performance indices by inequality constraints on performance indices. The simultaneous minimisation of all performance indices is normally impossible. However, in engineering design problems, what one requires is to retain performance indices in given regions. To be more specific, the multiobjective optimisation is being reformulated into the problem of searching for a parameter set Z, W, Q to satisfy the following inequalities:

$$\phi_i(Z, W, Q) \leq \varepsilon_i, \quad \text{for} \quad i = 1, 2, ..., 5 \tag{12.7.1}$$

where the real number ε_i represents the numerical bound on the performance index $\phi_i(Z, W, Q)$ required by the designer. If the minimal value of $\phi_i(Z, W, Q)$ achieved by minimising $\phi_i(Z, W, Q)$ itself is ϕ_i^*, the objective bound must be set as

$$\varepsilon_i \geq \phi_i^* \tag{12.7.2}$$

which is based on the fact that a parameter set which minimises a particular performance index can make other performance indices very large. If $\phi_i^*(Z^*, W^*, Q^*)$ is the minimal value of $\phi_i(Z, W, Q)$ achieved at the parameter set $\{Z_i^*, W_i^*, Q_i^*\}$, the following inequalities hold true:

$$\phi_i(Z_j^*, W_j^*, Q_j^*) \geq \phi_i^*(Z_i^*, W_i^*, Q_i^*) \tag{12.7.3}$$

where $i \neq j$, for $i, j \in \{1, 2, 3, 4, 5\}$. As a general rule, the performance boundaries ε_i should be set as

$$\phi_i^*(Z_i^*, W_i^*, Q_i^*) \leq \varepsilon_i \leq \max_{j, i \in \{1,2,3,4,5\}, j \neq i} \{\phi_i(Z_j^*, W_j^*, Q_j^*)\} \tag{12.7.4}$$

for $i = 1, 2, ..., 5$. The problem of multiobjective optimisation is to find a parameter set to place all performance indices in an acceptable region. By adjusting the bound ε_i, we can put a different emphasis on each of the objectives. If the performance index ϕ_j is important for the problem, one can let ε_j be close to ϕ_j^*. If the performance index ϕ_k is less important, one can let ε_k be far away from ϕ_k^*.

Let Φ_i be the set of parameters (Z, W, Q) for which the i-th objective is satisfied:

$$\Phi_i = \{(Z, W, Q) : \phi_i(Z, W, Q) \leq \varepsilon_i\} \tag{12.7.5}$$

Then, the admissible or feasible set of parameters for which all objectives hold is

$$\Phi = \{(Z, W, Q) : \phi_i(Z, W, Q) \leq \varepsilon_i, \quad \text{for} \quad i = 1, 2, ..., 5\} \tag{12.7.6}$$

The above admissible set can be solved by the method of inequalities. But, the difficult part of the method is the generation of a trial parameter set. Many methods have been proposed since Zakian introduced the method of inequalities, and a short review was given by Maciejowski (1989). It is suggested that relatively crude direct search methods, such as the simplex method, can be used to solve this problem. Here, we suggest a method to generate the trial parameter set via an genetic algorithm. This is inspired by Patton and Liu (1994) for their work in the design of robust controllers. The use of the genetic algorithm (GA) (Goldberg, 1989) obviates the requirement for the calculation of cost function gradients. GAs constitute a parallel search of the solution space, as opposed to a point-by- point search in gradient-descent methods. By using a population of trial solutions, the genetic algorithm can effectively explore many regions of the search space simultaneously, rather than a single region. This is one of the reasons why GAs are less sensitive to local minima. This is especially important when the cost function is not smooth, for example the maximal singular value functions used in this chapter. Finally, GAs manipulate representations of potential solutions, rather than the solutions themselves, and consequently do not require a complete understanding or model of the problem. The only problem-specific requirement is the ability to evaluate the trial solutions for relative fitness. Thus, a GA can be employed to search for the optimal solutions for multiobjective fault diagnosis.

12.8 An Example

The flight control system example we considered here is the lateral control system of a remotely piloted aircraft (Mudge and Patton, 1988). The linearised lateral dynamics are given by the state-space model matrices

$$A = \begin{bmatrix} -0277 & 0 & -32.9 & 9.81 & 0 \\ -0.1033 & -8.525 & 3.75 & 0 & 0 \\ 0.3649 & 0 & -0.639 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} -5.432 & 0 \\ 0 & -28.64 \\ -9.49 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D = 0_{3\times 2}$$

where the state vector and control input are

$$\begin{bmatrix} \mu \\ p \\ \gamma \\ \phi \\ \psi \end{bmatrix} = \begin{bmatrix} \text{sideslip} \\ \text{roll rate} \\ \text{yaw rate} \\ \text{bank angle} \\ \text{yaw angle} \end{bmatrix}$$

$$\begin{bmatrix} \tau \\ \zeta \end{bmatrix} = \begin{bmatrix} \text{rudder} \\ \text{aileron} \end{bmatrix}$$

The observer used in residual generation must be stable. To generate the required residual response, we consider the observer eigenvalues to be constrained within the following regions:

$$\begin{aligned} -5 \leq \quad & \lambda_1 \quad && \leq -0.2 \\ -15 \leq \quad & \lambda_2 \quad && \leq -3 \\ -10 \leq \quad & \lambda_{3,re} \quad && \leq -2 \\ 0.2 \leq \quad & \lambda_{3,im} \quad && \leq 4 \\ -30 \leq \quad & \lambda_5 \quad && \leq -8 \end{aligned}$$

Note that the eigenvalue λ_4 is the conjugate of the eigenvalue, λ_3, *i.e.* $\lambda_4 = \lambda_3^*$. The weighting penalty factors for the performance functions $\phi_1(Z, W, Q)$ and $\phi_3(Z, W, Q)$ are chosen as

$$\begin{aligned} W_1(s) &= \frac{500}{(s+10)(s+50)} \\ W_3(s) &= W_1^{-1}(s) \end{aligned}$$

which places emphasis on $\phi_1(Z, W, Q)$ at low frequencies and $\phi_3(Z, W, Q)$ at high frequencies. By minimising $\phi_1(Z, W, Q)$ and $\phi_3(Z, W, Q)$, the fault effect can be maximised and the noise effect can be minimised. To simplify the optimisation procedure, the residual weighting matrix Q here is set as

a 3×3 identity matrix. Table 12.1 lists the performance indices for different observer gains. In this table K_i^*, for $i = 1, 2, 3, 4$ is the observer gain matrix which minimises $\phi_i(Z, W, Q)$, for $i = 1, 2, 3, 4$. It can be seen that a design which minimises a particular performance function makes all other performance functions unacceptably large. Hence, multiobjective optimisation must be used to reach a reasonable compromise. In order to use the method of inequalities to solve this problem, a set of performance index bounds ε_i, $i = 1, 2, 3, 4$ are chosen as

$$\begin{aligned} \varepsilon_1 &= 2000 \\ \varepsilon_2 &= 0.16 \\ \varepsilon_3 &= 22.5 \\ \varepsilon_4 &= 0.006 \end{aligned}$$

Table 12.1: Performance indices for different designs

	ϕ_1	ϕ_2	ϕ_3	ϕ_4
K_1^*	189.58	2.5949	24.8288	0.00935
K_2^*	3865.26	0.07576	23.415	0.00798
K_3^*	3274.55	0.11232	22.40	0.00798
K_4^*	3.9e6	10700	34600	2e-7
$K_{optimal}$	1950.7	0.1492	22.42	0.00512
K_{place}	2800.39	0.1784	22.668	0.00965

The method of inequalities with genetic algorithms was used to search for solutions which satisfy all performance index boundaries. The optimal observer gain matrix found is

$$K_{optimal} = \begin{bmatrix} -189.1419 & 0.8083 & 18.8392 \\ 17.9317 & -0.7936 & -0.7943 \\ 15.4684 & 2.8543 & 7.6140 \\ -0.7606 & 6.9329 & 0-1537 \\ -1.2303 & 0.2329 & 9.8678 \end{bmatrix}$$

with corresponding eigenvalues

$$\{-1.5371 \quad -4.7045 \quad -3.4973 + 2.1194j \quad -3.4973 - 2.1194j \quad -19.9994\}$$

The performance indices under this gain are shown in Table 12.1. It provides an acceptable compromise. To demonstrate the effectiveness of the developed method, we also designed an observer gain matrix K_{place} using the MATLAB routine 'place' by assigning eigenvalues at $\{-0.5, -14, -4.8 + 1.6j, -4.8 - 1.6j, -20\}$. The performance indices for this design are also shown in Table 12.1.

A simulation is used to assess the performance of the observer-based residual generation in the detection of incipient faults. The system is unstable and needs to be stabilised. Since the purpose of the example is to illustrate the fault detection capability, we simply stabilise the system using a state feedback controller which is provided by Liu and Patton (1998a). The control commands in both control inputs are set as unit sinusoid functions. The sensor noise comprises a random summation of multi-frequency signals with all frequencies larger than 20 rad/s. In the simulation, all aerodynamic coefficients have been perturbed by $\pm 10\%$.

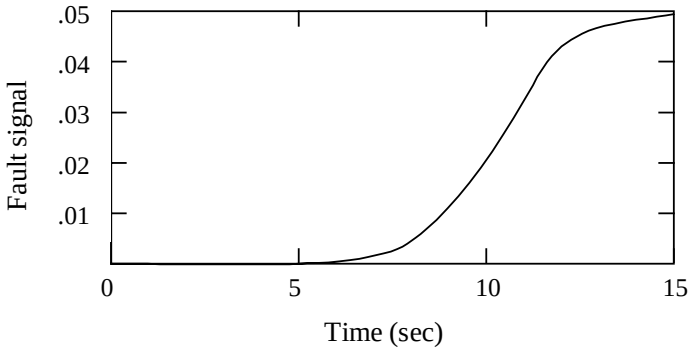

Figure 12.1: Fault signal shape

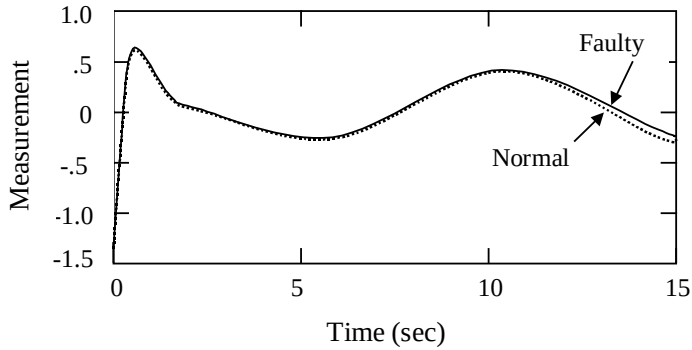

Figure 12.2: Faulty and normal measurements for roll rate p

The fault is a slowly developing signal whose shape is shown in Figure 12.1. To illustrate the small nature of the incipient fault, Figure 12.2 shows the plot of both normal and faulty measurements of the roll rate sensor when a fault occurs in the roll rate sensor. It can be seen that the fault is hardly noticeable in the measurement and cannot be detected easily, without the assistance of

the residuals.

Figure 12.3 shows the residual response for the case when a fault occurs in the roll rate sensor. The residual responses for other faulty cases are similar to the response shown in Figure 12.3. The residual response demonstrates that the residual changes very significantly after a fault occurs in one of sensors. Hence, the residual can be used to detect incipient sensor faults reliably even in the presence of modelling errors and noise. To reduce the effect of noise further, the residual signal has been filtered by a low-pass filter. Note that this chapter only has addressed the robust residual generation for fault detection, as it is believed that the design of an optimal residual is the most important task to be considered. Fault isolation can be achieved by designing structured residual sets. For the system considered in this chapter, we can design four different observer-based residual generators to generate four residual vectors. The four observers are driven by different subsets of measurements, namely $\{p, \phi, \psi\}$, $\{p, \gamma, \psi\}$, $\{p, \phi, \gamma\}$ and $\{r, \phi, \psi\}$. This chapter has only discussed the design of one of these observers, although the principle is valid for the design of the other observers.

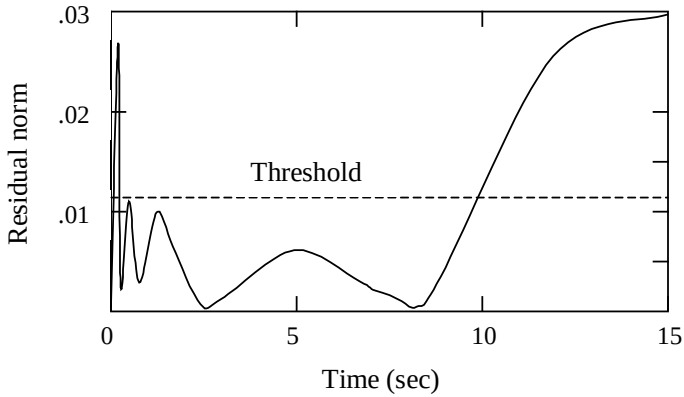

Figure 12.3: Residual norm when a fault occurs in the roll rate sensor

12.9 Summary

A systematic approach to the design of optimal residuals which satisfy a set of objectives has been described. These objectives are essential to achieve robust diagnosis of incipient faults. Five performance indices are expressed in the frequency domain which can take account of the frequency distribution of different factors which affect the residuals. It has been shown that the frequency-dependent weighting factors incorporated into performance indices play an important role in the optimal design. They are problem-dependent

and must be chosen very carefully. The multiobjective optimisation problem was reformulated into one of satisfying a set of inequalities on the performance indices. The genetic algorithm was used to search the optimal solution to satisfy these inequalities on the performance indices. The method has been applied to the design of an observer-based residual generator for detecting incipient sensor faults in a flight control system, and the simulation results show the effectiveness of the method. Considering the extreme difficulty of enhancing the fault diagnosis performance under modelling uncertainty and noise, any improvement in the robustness of residual design is very useful. The scope of application of this method can be extended to all systems with possible incipient faults.

Bibliography

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Trans. on Automatic Control*, AC-19, pp. 716-723.

Anderson, B. D. O. and J. B. Moore (1971). *Linear Optimal Control*, Prentice-hall.

Andry, A. E., E. Y. Shapiro and J. C. Chung (1983). Eigenstructure assignment for linear systems. *IEEE Trans. Aerospace Electron. Syst.*, vol. AES-19, pp. 711-729.

Astrom, K. J. (1970). *Introduction to Stochastic Control Theory*. Academic Press, New York.

Astrom, K. J. and T. Hagglund (1984). Automatic tuning simple regulators with specifications on phase and amplitude margins. *Automatica,* vol. 20, no. 5, pp. 645-651.

Back, T., F. Hoffmeister and H. P. Schwefel (1991). A Survey of Evolution Strategies. *Proc. JCGA*, 4, pp. 2-10.

Bada, A. T. (1987a). Robust brake system control for heavy duty truck. *Proc. IEE Part D*, vol. 134, no. 1, pp. 1-8.

Bada, A. T. (1987b). Criteria: a package for computer-aided control system design. *Computer-Aided Design*, vol. 19, no. 9, pp. 466-474.

Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. *Proc. JCGA* 1, pp. 101-111.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. *Proc. JCGA* 2, pp. 14-21.

Balas, G. J., J. C. Doyle, K. Glover, A. Packard and R. Smith (1991). *μ-Analysis and Synthesis Toolbox: User's Guide*. Natick, MA: The MathWorks, Inc.

Baras, J. S., M. K. H. Fan, W. T. Nye and A. L. Tits (1984). DELIGHT.LQG: A CAD system for control systems design using LQG controller structure. *18th Annual Conf. on Inf. Sci. and Syst.* Princeton, NJ.

Barnett, S. (1971). *Matrices in Control Theory - with applications to linear programming*. London, U.K.: Van Nostrand Reinhold.

Becker, H. G., A. J. Heunis and D. Q. Mayne (1979). Computer-aided design of control systems via optimisation. *Proc. Instn Elect. Engrs*, 126, pp. 573-578.

Ben-Tal, A. (1980). Characterization of Pareto and lexicographic optimal solutions. *Multiple Criteria Decision Making Theory and Application*, vol. 177 of Lecture Notes in Economics and Mathematical Systems, G. Fandel and T. Gal (eds.), Berlin: Springer-Verlag, pp. 1-11.

Billings, S. A. and S. Chen (1992). Neural Networks and System Identification. *Neural Networks for Systems and Control*, K. Warwick *et al.* (eds.), pp. 181-205.

Bollinger, K. E., P. A. Cook and P. S. Sandhu (1979). Synchronous generator controller synthesis using moving boundaries techniques with frequency domain constraints. *IEEE Trans. Power Appar. Syst.*, vol. PAS-98, no. 5, pp. 1497-1501.

Booker, L. (1987). Improving search in genetic algorithms, in *Genetic Algorithms and Simulated Annealing*, L Davis (ed.), pp. 61-73, Morgan Kaufmann Publishers.

Boyd, S. and C. Barratt (1991). *Linear Controller Design: Limits of Performance*. Englewood Cliffs, N.J.: Prentice-Hall.

Bramlette, M. F. (1991). Initialization, mutation and selection methods in genetic algorithms for function optimization. *Proc. JCGA*, 4, pp. 100-107.

Broomhead, D. S. and D. Lowe (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, vol. 2, pp. 321-355.

Burrows, S. P. and R. J. Patton (1991). Design of a low sensitivity, minimum norm and structurally constrained control law using eigenstructure assignment. *Optimal Control Application and Methods*, vol. 12, no. 3, pp. 131-140.

Cai, X. S. (1982). *Optimisation and Optimal Control*. Qinghua University Press, Beijing.

Caruana, R. A., L. A. Eshelman, J. D. Schaffer (1989). Representation and hidden bias II: Eliminating defining length bias in genetic search via shuffle crossover. *the Eleventh International Joint Conference on Artificial Intelligence*, N. S. Sridharan (ed.), vol. 1, pp. 750-755, Morgan Kaufmann Publishers.

Caruana, R. A. and J. D. Schaffer (1988). Representation and hidden bias: gray vs binary coding. *Proc. 6th Int. Conf. Machine Learning*, pp. l53-161.

Charnes, A. and W. W. Cooper (1977). Goal programming and multiple objective optimisation - part I. *European Journal of Operational Research*, vol. 1, pp. 39-54.

Chen, D. S. and R. C. Jain (1994). A robust back propagation learning algorithm for function approximation. *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 467-479.

Chen, S., S. A. Billings, and P. M. Grant (1990). Nonlinear system identification using neural networks. *International Journal of Control*, vol. 51, no. 6, pp. 1191-1214.

Chen, J. (1995). *Robust Residual Generation For Model-Based Fault Diagnosis of Dynamical Systems*. Ph.D Thesis, University of York, UK.

Chen, J., R. J. Patton and G. P. Liu (1996). Optimal residual design for fault diagnosis using multi-objective optimisation and genetic algorithms. *International Journal of Systems Science*, vol. 27, no. 6, pp. 567-576.

Chiang, Y. and G. Safonov (1988). *Robust Control Toolbox for Use with MATLAB*. MathWorks Inc.

Chiang, R. Y. and M. G. Safonov (1992). *Robust Control Toolbox: User's Guide*. Natick, MA: The MathWorks, Inc.

Chipperfield, A., P. J. Fleming, H. Pohlheim and C. M. Fonseca (1994). *Genetic Algorithm Toolbox: User's Guide*. Dept. Automatic Control and Systems Engineering, University of Sheffield, UK.

Chipperfield, A. J., J. F. Whidborne and P. J. Fleming (1999). Evolutionary algorithms and simulated annealing for MCDM. *Multicriteria Decision Making - Advances in MCDM Models, Algorithms, Theory, and Applications*, T. Gal, T. J. Stewart and T. Hanne (eds.). Boston: Kluwer, pp. 16.1-16.32.

Clarke, D. W., C. Mohtadi and P. S. Tuffs (1987). Generalized predictive control – Part I. the basic Algorithm. *Automatica*, vol. 22, no. 2, pp. 137-148.

Clarke, P. (1997). MATRIXx 6.0: a fixed point development process for automotive applications. *IMechE Seminar Publication - Automotive Electronics (Autotech '97)*. Birmingham, U.K.: Mechanical Engineering Publications, pp. 59-67.

Coelho, C. A. D. (1979). Compensation of the speed governor of a water turbine by the method of inequalities. *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 101, no. 3, pp. 205-211.

Cohon, J.L. (1978). *Multiobjective Programming and Planning*. New York: Academic Press.

Crossley, T. R. and A. M. Dahshan (1982). Design of a longitudinal ride-control system by Zakian's method of inequalities. *J. Airct.*, vol. 19. no. 9, pp. 730-738.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, no. 2, 303-314.

Dahleh, M. A. and J. B. Pearson (1987a). l^1 -optimal controllers for MIMO discrete-time systems. *IEEE Trans. Automat. Contr.*, vol. 32, pp. 314-322.

Dahleh, M. A. and J. B. Pearson (1987b). L^1-optimal compensators for continuous-time systems. *IEEE Trans. Automat. Contr.*, vol. 32, pp. 889-895.

Dahshan, A. M. (1981). A Comparison between the method of inequalities and other techniques in control systems design. *Int. J. Systems Sci.*, vol. 12, no. 9, pp. 1149-1155.

Daley, S. and G. P. Liu (1999). Optimal PID tuning using direct search algorithm, *IEE Computing and Control Engineering Journal*, vol. 10, no. 2, pp. 51-56.

Dakev, N. V., J. F. Whidborne, A. J. Chipperfield and P. J. Fleming (1997). H_∞ design of an EMS control system for a maglev vehicle using evolutionary algorithms. *Proc. IMechE, Part I: J. Syst. & Contr.*, vol. 311, no. 4, pp. 345-355.

Davis, L. (ed.) (1991). *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold.

Davis, L. (1989). Adapting operator probabilities in genetic algorithms. *Proc. IGGA* vol. 3, pp. 61-69.

De Jong, K. A. (1975). *Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD Thesis, Dept. of Computer and Communication Sciences, University of Michigan, Ann Arbor.

De Jong, K. A. and J. Sarma (1993). Generation gaps revisited. *Foundations of Genetic Algorithms 2*, L. D. Whitley (ed.), Morgan Kaufmann Publishers.

Ding, X. and P. M. Frank (1989). Fault detection via optimally robust detection filters. *Proc. 28th IEEE Conference on Decision amid Control*, Tampa, Florida. U.S.A. pp. 1767-1772.

Ding, X. and P. M. Frank (1990). Fault detection via factorization approach. *Systems and Control Letters*, vol. 14, pp. 431- 436.

Dixon, R., A. W. Pike and M. S. Donne (2001). The ALSTOM benchmark challenge on gasifier control. *IMechE Proceedings, Part I*, vol. 215, pp. 389-394.

Donne, M. S., R. Dixon, A. W. Pike, A. J. L. Odeku and B. E. Ricketts (1998). Dynamic modelling of the ABGC prototype integrated plant. *DTI Coal Research and Development Programme*, ETSU Report no. COAL R143.

Doyle, J. C., K. Glover, P. P. Khargonekar and B. A. Francis (1989). State space solutions to the standard H^2 and H^∞ control problems. *IEEE Trans. on Automatic Control*, vol. 34, pp. 831-847.

Doyle, J. C., B. A. Francis and A. R. Tannenbaum (1991). *Feedback Control Theory*. New York: Macmillan.

Duan, G. R. (1993). Robust eigenstructure assignment via dynamical compensators. *Automatica*, vol. 29, no. 2, pp. 469-474.

Fleming, P. J. (1986). Application of multi-objective optimization to compensator design for SISO control systems. *Electronics Letters*, vol. 22, no. 5, pp. 258-259.

Fleming, P. J. and A. P. Pashkevich (1986). Application of multi-objective optimization to compensator design for SISO control systems. *Electronics Letters*, vol. 22, no. 5, pp. 258-259.

Fogarty, T. C. (1989a). Varying the probability of mutation in the genetic algorithm. *Proc. IGGA*, vol. 3, pp. 104-109.

Fogarty, T. C. (1989b). An incremental genetic algorithm for real-time learning. *Proc. 6th Int. Workshop on Machine Learning*, pp. 416-419.

Fonseca, C.M. and P.J. Fleming (1993a). Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. *Genetic Algorithms: Proceeding of the Fifth International Conference*. San Mateo, CA: pp. 416-423.

Fonseca, C. M. and P. J. Fleming (1993b). Multiobjective genetic algorithms. *IEE Colloquium on Genetic Algorithms for Control Systems Engineering*. Number 1993/130 London, U.K.: pp. 6/1-6/5.

Fonseca, C. M. and P. J. Fleming (1994). Multiobjective optimal controller design with genetic algorithms. *Proc. CONTROL 94*, Coventry, U.K.: pp. 745-749.

Fonseca, C. M. and P. J. Fleming (1995). Multiobjective optimization and multiple constraint handling with evolutionary algorithms – part I: a unified formulation. *IEEE Trans. Syst. Man & Cybernetics – A*, vol. 28, no. 1, pp. 26–37.

Fonseca, C. M., E. M. Mendes, P. J. Fleming and S. A. Billings (1993). Nonlinear model term selection with genetic algorithms. *Proc. IEE/IEEE Workshop on Natural Algorithms for Signal Processing*, Essex, U.K., vol. 2, pp. 27/1-27/8.

Fourman, M. P. (1985). Compaction of symbolic layout using genetic algorithms. *Genetic Algorithms and Their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pp. 141-153.

Francis, B. A. (1987). A Course in H^∞ Control Theory. *Lecture Notes in Control and Information Sciences*, vol. 88, New York: Springer-Verlag.

Francis, B. A. and G. Zames (1984). On H^∞-optimal feedback theory for SISO feedback systems. *IEEE Trans. Automat. Contr.*, vol. AC-29, no. 1, pp. 9-16.

Frank, P. M. (1990). Fault diagnosis in dynamic system using analytical and knowledge based redundancy a survey and some new results. *Automatica*, vol. 26, pp.

459-474.

Frank, P. M. and X. Ding (1993). Frequency domain approach to minimizing detectable faults in FDI systems. *Applied Mathematics and Computer Science*. vol. 3, pp. 417-443.

Frank, P. M. and J. Wunnenberg (1989). Robust fault diagnosis using unknown input schemes. *Fault Diagnosis in Dynamical Systems: Theory amid Application*, R. J. Patton *al.* (eds.), Prentice Hall, pp. 47-98.

Franklin, G. F. and J. D. Powell (1980). *Digital Control of Dynamic Systems*. Addison-Wesley, Reading, MA.

Franklin, G. F., J. D. Powell, and A. Emami-Naeini (1986). *Feedback Control of Dynamic Systems*. Addison-Wesley, Reading, MA.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, vol. 19, no. 1.

Furuya, H. and R. T. Haftka (1993). Genetic algorithms for placing actuators on space structures. *Proc. ICGA*, vol. 5, pp. 536-542.

Geman, S., E. Bienenstock and R. Boursat (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, vol. 4, pp. 1-58.

Gembicki, F. W. (1974). *Vector Optimization for Control with Performance and Parameter Sensitivity Indices*. Ph.D. Dissertation, Case Western Reserve Univ., Cleveland, Ohio.

Gembicki, F. W. and Y. Y. Haimes (1975). Approach to performance and sensitivity multiobjective optimization: the goal attainment method. *IEEE Trans. Autom. Control*, vol. 20, no. 8, pp. 821-830.

Geoffrion, A. M., J. S. Dyer and A. Feinberg (1972). An interactive approach for multicriteria optimisation, with an application to the operation of an academic department. *Management Science*, vol. 19, no. 4, pp. 357-368.

Gertler, J. (1991). Analytical redundancy methods in failure detection and isolation. *IFAC IMACS Symposium SAEEPROCESS'91*, Baden-Baden, Germany, vol. 1. pp. 9-21.

Gertler, J. and M. K. Kunwer (1993). Optimal residual decoupling for robust fault diagnosis. *Proc. Int. Conf. on Fault Diagnosis: TOOLDIAG '93*, Toulouse, France, pp. 436-452.

Gevers, M. and G. Li (1993). *Parametrizations in Control, Estimations and Filtering Problems: Accuracy Aspects*. Berlin: Springer-Verlag.

Gill, P. E., W. Murray, and M. H. Wright. (1981). *Practical Optimization*. Academic Press, London.

Giesy, D. P. (1978). Calculation of pareto-optimal solutions to multiple-objective problems using threshold-of-acceptability constraints. *IEEE Trans. Autom. Control*, vol. 23, no. 6, pp. 1114-1115.

Glover, K. and D. C. McFarlane (1989). Robust stabilization of normalized coprime factor plant descriptions with H_∞-bounded uncertainty. *IEEE Trans. Autom. Control*, vol. AC-34, no. 8, pp. 821–830.

Glover, J. D. and F. C. Schweppe (1971). Control of linear dynamic systems with set constrained disturbances. *IEEE Trans. Autom. Control*, vol. 16, pp. 411-422.

Goldberg, D. E. (1989). *Genetic Algorithm in Search, Optimization, and Machine Learning*. Addison Wesley Publishing Company.

Goodwin, G. C. and R. H. Middleton (1980). High speed digital signal processing and control. *Proc. IEEE*, vol. 80, pp. 240–259.

Goodwin, G. C. and K. S. Sin (1984). *Adaptive Filtering Prediction and Control*. Prentice-Hall, Englewood Cliffs, NJ.

Grace, A. (1994). *Optimisation Toolbox for Use with MATLAB*. The MathWorks Inc.

Graebe, S. F. (1994). Robust and adaptive control of an unknown plant: a benchmark of new format. *Automatica*, vol. 30, no. 4, pp. 567-576.

Green, M. and D. J. N. Limebeer (1995). *Linear Robust Control*. Englewood Cliffs, N.J.: Prentice-Hall.

Grefenstette, J. J. (1987). Incorporating problem specific knowledge into genetic algorithms. *Genetic Algorithms and Simulated Annealing*, pp. 42-60, L Davis (ed.), Morgan Kaufmann.

Grubel, G., H.-D. Joos, M. Otter and R. Finsterwalder (1993). The ANDECS design environment for control engineering. *Proc. 12th IFAC World Congress*. Vol. 6 Sydney: pp. 447-454.

Haessig, D. (1995). Selection of LQR/LTR weighting matrices through constrained optimisation. *Proc. 1995 Amer. Contr. Conf.*, Seattle, USA: pp. 458-460.

Haimes, Y. Y., W. A. Hall and H. T. Freedman (1975). *Multiobjective Optimisation in Water Resource Systems, The Surrogate Worth Trade-off Methods*. Elsivier Scientific, New York.

Hajela, P. and C.-Y. Lin (1992). Genetic search strategies in multicriterion optimal design. *Structural Optimization*, vol. 4, pp. 99-107.

Han, Z.Z. (1989). Eigenstructure assignment using dynamical compensator. *International Journal of Control*, vol. 49, pp. 233-245.

Hang, C. C., K. J. Astrom and W. K. Ho (1991). Refinements of the Ziegler-Nichols tuning formula. *IEE Proceedings-D*, vol. 138, no. 2, pp. 111-118.

Hart, W. E. and R. K. Belew (1991). Optimizing an arbitrary function is hard for the genetic algorithm. *Genetic Algorithms: Proceedings of the Fourth International Conference*, San Mateo, pp. 190-195.

Haykin, S. (1994). *Neural Networks: A Comprenhensive Foundation*. Macmillan College Publishing Company, New York.

Hernstein, I. N. and D. J. Winter (1988). *Matrix Theory and Linear Algebra*. New York: Macmillan.

Holstien, R. B. (1971). *Artificial Genetic Adaptation an Computer Control Systems*. PhD Thesis, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press.

Hori, Y. (1996). A review of torsional vibration control methods and a proposal of disturbance observer-based new techniques. *Proc. 13th IFAC World Congress*, San Francisco, CA: pp. 7-13.

Horn, R. A. and C. R. Johnson (1985). *Matrix Analysis*. Cambridge, U.K: Cambridge University Press.

Horn, R. A. and C. R. Johnson (1991). *Topics in Matrix Analysis*. Cambridge, U.K.: Cambridge University Press.

Hoyle, D. J., R. A. Hyde and D. J. N. Limebeer (1991). An H_∞ approach to two degree of freedom design. *Proc. 30th IEEE Conf. Decision Contr.*, Brighton, England: pp. 1579-1580.

Huang, C. L. and A. S. Masud (1979a). *Multiple Objective Decision Making Methods and Applications, A State-of-Art Survey*. Springer-Verlag, Berlin.

Huang, C. L. and A. S. Masud (1979b). *Multiple Objective Decision Making - Methods and Applications*. vol. 164 of Lecture Notes in Economics and Mathematical Systems. Berlin: Springer-Verlag.

Huang, R. and T. C. Fogarty (1991). Adaptive classification and control-rule optimization via a learning algorithm for controlling a dynamic system. *Proc. 30th Conf. Decision and Control*, Brighton, England, pp. 867-868.

Hyde, R. A. (1995). H_∞ *aerospace control design: a VSTOL flight application*. Berlin: Springer.

Inooka, H. (1982). Improved time-response algorithm for the method of inequalities. *Int. J. Control*, vol. 35, no. 1, pp. 127-138.

Istepanian, R. H., G. Li, J. Wu and J. Chu (1998a). Analysis of sensitivity measures of finite-precision digital controller structures with closed-loop stability bounds. *IEE Proc. Control Theory and Appl.*, vol. 145, no. 5, pp. 472–478.

Istepanian, R. H., and J. F. Whidborne (2001). *Digital Controller Implementation and Fragility: A Modern Perspective*. Springer-Verlag, London.

Istepanian, R. H., J. Wu, J. F. Whidborne and J. Chu (1998b). Maximizing lower bound stability measure of finite precision PID controller realizations by nonlinear programming. *Proc. 1998 Amer. Contr. Conf.*, Philadelphia, PA: pp. 2596–2600.

Jakob, W., M. Gorges-Schleuter and C. Blume (1992). Application of genetic algorithms to task planning and learning. *Parallel Problem Solving from Nature*, R. Minner and B. Manderick (eds.), Amsterdam: North-Holland, pp. 291-300.

Janikow, C. Z. and Z. Michalewicz (1991). An experimental comparison of binary and floating point representations in genetic algorithms. *Proc. ICGA*, vol. 4, pp. 31-36.

Kadirkamanathan, V. and M. Niranjan (1993). A function estimation approach to sequential learning with neural networks. *Neural Computation*, vol. 5, pp. 954-957.

Karr, C. L. (1991). Design of an adaptive fuzzy logic controller using a genetic algorithm. *Proc. ICGA*, vol. 4, pp. 450-457.

Kautsky, J., N. K. Nichols and P. Van. Dooren (1985). Robust pole assignment in linear state feedback. *International Journal of Control*, vol. 41, no. 5, pp. 1129-1155.

Khargonekar, P. P. and M. A. Rotea (1991). Multiple objective optimal control of linear systems: the quadratic norm case. *IEEE Trans. Automat. Contr.*, vol. 36, no.1, pp. 14-24.

Kimura, H., (1975). Pole assignment by gain output feedback. *IEEE Transactions on automatic Control*, vol. 20, no. 4, pp. 509-516.

Kipp, H. W., (1989). *Narrow Fabric Weaving.* Arau, Frankfurt am Main, Sauerlander, Salzburg.

Knowles, J. B. and R. Edwards (1965). Effect of a finite-word-length computer in a sampled-data feedback system. *Proc. IEE*, vol. 112, no. 6, pp. 1197-1207.

Kortum, W. and A. Utzt (1984). Control law design and dynamic evaluations for a maglev vehicle with a combined lift and guidance suspension system. *ASME J. Dyn. Syst. Meas. & Control*, vol. 106, pp. 286-292.

Koussoulas, N. T. and C. T. Leondes (1986). The multiple linear quadratic Gaussian problem. *International Journal of Control*, vol. 43, no. 2, pp. 337-349.

Kreisselmeier, G. and R. Steinhauser (1983). Application of vector performance optimization to robust control loop design for a fighter aircraft. *International Journal of Control*, vol. 37, no. 2, pp. 251-284.

Kwon, B.H. and M.J. Youn (1987). Eigenvalue-generalized eigenvector assignment by output feedback. *IEEE Transactions on Automatic Control*, vol. 32, no. 5, pp. 417-421.

Kursawe, F. (1991). A variant of evolution strategies for vector optimization. *Parallel Problem Solving from Nature. vol. 496 of Lecture Notes in Computer Science*, H. P. Schwefel and R. Minner (eds.), Berlin: Springer-Verlag, pp. 193-197.

Kuschewski, J. G., S. Hui and S. H. Zak (1993). Application of feedforward neural networks to dynamical system identification and control. *IEEE Trans. on Control Systems Technology*, vol. 1, no. 1, pp. 37-49.

Lane, S. H., D. A. Handelman and J. J. Gelfand (1989). Development of adaptive B-splines using CMAC neural networks. *Proceedings of the International Joint Conference on Neural Networks*, Wanshington D.C.

Lapedes, A. S. and R. M. Farber (1987). Nonlinear signal processing using neural networks: prediction and system modelling. *Technical Report LA-UR-87-2662*, Los Alamos National Laboratory.

Lapedes, A. S. and R. M. Farber (1988). How neural networks work. *Neural Information Processing Systems*, D. Z. Anderson (ed.), American Institute of Physics, Denver, pp. 442-456.

Lehtomaki, N. A. (1981). *Practical Robustness Behaviors in Multivariable Control System Analysis*, Ph.D Thesis, MIT.

Li, D. and S. D. Qian (1982). *Operational Research.* Qinghua University Press, Beijing.

Li, D. and J. B. Yang (1996). Iterative parametric minimax method for a class of composite optimisation problems. *Journal of Mathematical Analysis and Applications*, vol. 198, pp. 64-83.

Li, D., J. B. Yang and M. P. Biswal (1999). Quantitative parametric connections between methods for generating noninferior solutions in multiobjective optimisation. *European Journal of Operational Research*, vol. 117, no. 1, pp. 84-99.

Lightner, M. R. and S. W. Director (1981). Multiple criterion optimisation for the design of electronic circuits. *IEEE Transactions on Circuits and Systems*, vol. 28,

no. 3, pp. 169-179.

Lin, J. G., (1976). Multi-objective problems: Pareto-optimal solutions by method of proper equality constraints. *IEEE Trans. Autom. Control*, vol. 21, no. 5, pp. 641-650.

Lin, J. G. and J. B. Yang (1999). GA based multiple objective optimisation for determining viscoplastic constitutive equations for superplastic alloys. *International Journal of Plasticity*, vol. 15, pp. 1181-1196.

Limebeer, D. J. N. (1991). The specification and purpose of a controller design case study. *Proc. 30th IEEE Conf. Decision Contr.*, Brighton, England: pp. 1579-1580.

Limebeer, D. J. N., E. M. Kasenally and J. D. Perkins (1993). On the design of robust two degree of freedom controllers. *Automatica*, vol. 29, no. 1, pp. 157-168.

Lippmann, R. P. (1987). An introduction to computing with neural networks. *IEEE ASSP Magazine*, pp. 4-22.

Liu, G. P. (1990). Frequency-domain approach for critical systems. *International Journal of Control*, vol. 52, no. 6, pp. 1507-1519.

Liu, G. P (1992a) *Theory and Design of Critical Control Systems.* Ph.D Thesis, Control Systems Centre, University of Manchester Institute of Science and Technology, U.K.

Liu, G. P. (1992b). Disturbance space and sup regulator in discrete time. *Systems and Control Letters*, vol. 18, no. 1, pp. 33-38.

Liu, G. P. (1992c). Mean regulators for discrete systems. *IEE Proceedings, Part D*, vol. 139, no.1, pp. 67-71.

Liu, G. P. (1992d). Mean controllers for discrete systems, *Int. J. Systems Sci.*, vol. 23, no. 12, pp. 2219-2230.

Liu, G. P. (1993). Input space and output performance in the frequency domain for critical systems. *IEEE Transactions on Automatic Control*, vol. 38, no. 1, pp. 152-155.

Liu, G. P. and S. Daley (1998). Stable dynamical controller design for robust polynomial pole assignment. *IEE Proceedings, Part D*, vol. 145, no. 3, pp. 259-264.

Liu, G. P. and S. Daley (1999). Optimal-tuning PID controller design in the frequency-domain with application to rotary hydraulic systems. *Control Engineering Practice*, vol. 7, pp. 821-830.

Liu, G. P. and S. Daley (2000). Optimal-tuning nonlinear PID control for hydraulic systems. *Control Engineering Practice,* vol. 8, no. 9, pp. 1045-1053.

Liu, G. P. and S. Daley (2001). Optimal-tuning PID control for industrial systems, *Control Engineering Practice*, vol. 9, 2001 (accepted).

Liu, G. P., R. Dixon and S. Daley (1998). Multiobjective optimal-tuning PI control design for a gasifier. *MEC Benchmark Challenge on Gasifier, IMechE Seminar*, Coventry, 1998.

Liu, G. P., R. Dixon and S. Daley (2000). Multiobjective optimal-tuning PI control design for the ALSTOM benchmark gasifier, *IMechE Proceedings, Part I*, vol. 214, pp. 395-404.

Liu, G. P. and V. Kadirkamanathan (1999). Multiobjective criteria for nonlinear model selection and identification with neural networks. *IEE Proceedings, Part*

D, vol. 146, no. 5, pp. 373-382.

Liu, G. P., V. Kadirkamanathan and S. A. Billings (1996). Stable sequential identification of continuous nonlinear dynamical systems by growing RBF networks. *International Journal of Control*, vol. 65, no. 1, pp. 53-69.

Liu, G. P., V. Kadirkamanathan and S. A. Billings (1998). On-line identification of nonlinear systems using Volterra polynomial basis function neural networks. *Neural Networks*, vol. 11, pp. 1645-1657.

Liu, G. P., V. Kadirkamanathan and S. A. Billings (1999). Variable neural networks for adaptive control of nonlinear systems. *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 29, no. 1, pp. 34-43.

Liu, G. P. and R. J. Patton (1995). Parametric state feedback controller design of multivariable systems. *International Journal of Control*, vol. 61, no. 6, pp. 1457-1464.

Liu, G. P. and R. J. Patton (1996a). Robust control design using eigenstructure assignment and multiobjective optimization. *Int. Journal of Systems Science*, vol. 27, no. 9, pp. 871-879.

Liu, G. P. and R. J. Patton (1996b). On eigenvectors and generalized eigenvectors for descriptor state-feedback systems. *Proceedings IME: Journal of Control System Engineering*, vol. 210, pp. 183-188.

Liu, G. P. and R. J. Patton (1998a). *Eigenstructure Assignment for Control System Design*. John Wiley & Sons, Chichester.

Liu, G. P. and R. J. Patton (1998b). Mixed time- and frequency domain robust eigenstructure assignment. *International Journal of Robust and Nonlinear Control*, vol. 8, pp. 61-78.

Liu, G. P. and R. J. Patton (1998c). Robust control design based on parametric eigenstructure assignment. *Int. Journal of Systems Science*, vol. 29, no. 1, pp. 65-74.

Liu, G. P. and R. J. Patton (1998d). Robust eigenstructure assignment for descriptor systems. *Int. Journal of Systems Science*, vol. 29, no. 1, pp. 75-84.

Liu, G. P. and R. J. Patton (1998e). Generic parametric controller design using output feedback eigenstructure assignment. *Proceedings IME: Journal of Control System Engineering*, vol. 212, pp. 71-78.

Liu, G. P. and R. J. Patton (1999). *Eigenstructure Assignment Toolbox for Use with Matlab*. Pacilantic International Ltd., Oxford.

Liu, G. P., H. Wang, and J. X. Tian (1990). The design of a hydraulic turbine control system – a critical system. *AMSE Periodicals: Modelling, Simulation and Control, B*, vol. 33, no. 3, pp. 23-34.

Liu, G. P., H. Unbehauen and R. J. Patton (1995). Robust control of multivariable critical systems. *International Journal of Systems Science*, vol.26, no. 10, pp. 1907-1918.

Liu, T. K., T. Satoh, T. Ishihara and H. Inooka (1994). An application of genetic algorithms to control system design. *Proc. 1st Asian Contr. Conf.* vol. III, Tokyo: pp. 701-704.

Lucasius, C. B. and G. Kateman (1992). Towards solving subset selection problems

with the aid of the genetic algorithm. *Parallel Problem Solving from Nature 2*, R. Manner and B. Manderick (eds.), Amsterdam: North-Holland, pp. 239-247.

Luenberger, D. G. (1969). *Optimization by Vector Space Methods.* New York, Wiley.

Luenberger, D. G. (1973). *Introduction to Linear and Nonlinear Programming.* Addison-Wesley, Reading, Massachusetts.

MacKay, D. J. C. (1992). Bayesian interpolation. *Neural Computation*, vol. 4, pp. 415-447.

Maciejowski, J. M. (1989). *Multivariable Feedback Design*, Addison-Wesley.

Man, K. F., K. S. Tang, S. Kwong and W. A. Halang (1997). *Genetic Algorithms for Control and Signal Processing.* Advances in Industrial Control London, U.K.: Springer.

Mangoubi, R. , B. D. Appleby and J. R. Farrel (1992). Robust estimation in fault detection. *Proceedings of the 31st IEEE Conference on Decision and Control*, Arizona, USA, pp. 2317-2122.

Masten, M. K. and I. Panahi (1997). Digital signal processors for modern control systems. *Control Engg Practice*, vol. 5, no. 4, pp. 449-458.

Mayne, D. Q., E. Polak and A. J. Heunis (1981). Solving non-linear inequalities in a finite number of iterations. *J. Optim. Theory Appl.*, vol. 33, pp. 207-221.

McCormack, A. S. and K. Godfrey (1998). Rule-based autotuning based on frequency domain identification. *IEEE Transactions on Control Systems Technology*, vol. 6, no. 1, pp. 43-61.

McCulloch, W. W. and W. Pitts (1943). A logical calculus of the ideas imminent in nervous activity. *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133.

McFarlane, D. C. and K. Glover (1990). *Robust Controller Design Using Normalized Coprime Factor Plant Descriptions.* vol. 138 of *Lect. Notes Control & Inf. Sci.* Berlin: Springer-Verlag.

McFarlane, D. C. and K. Glover (1992). A loop shaping design procedure using H_∞ synthesis. *IEEE Trans. Autom. Control*, vol. AC-37, no. 6, pp. 759–769.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs.* Springer Verlag.

Michalewicz, Z. and C. Z. Janikow (1991). Handling constraints in genetic algorithms. *Genetic Algorithms: Proceedings of the Fourth International Conference*, San Mateo, pp. 151-157.

Minoux, M. (1986). *Mathematical Programming - Theory and Algorithms.* John Wiley & Sons, Chichester.

Moody, J. E. and C. J. Darken (1989). Fast learning in networks of locally-tuned process units. *Neural Computation*, vol. 1, pp. 281-294.

Moore, B.C. (1976). On the flexibility offered by state feedback in multivariable systems beyond closed-loop eigenvalue assignment. *IEEE Transactions Automatic Control*, vol. 21, no. 6, pp. 689-692.

Morony, P. (1983). *Issues in the Implementation of Digital Feedback Compensators.* Number 5 *in* Signal Processing, Optimization, and Control Series Cambridge, MA: MIT Press.

Mudge, S. K. and R. J. Patton (1988). An analysis of the technique of robust eigen-

structure assignment with application to aircraft control. *IEE Proceedings, Part D, Control Theory and Applications*, vol. 135, no. 4, pp. 275-281.

Muhlenbein, H. and D. Schlierkamp-Voosen (1993). Predictive models for the breeder genetic algorithm. *Evolutionary Computation*, vol. 1, no. 1, pp. 25-49.

Muller, P. C. (1977). Design of optimal state-observers and its application to maglev vehicle suspension control. *Proc. 4th IFAC Symp. Multivariable Technological Systems*. Fredericton, Canada: pp. 175-182.

Murad, G., I. Postlethwaite, D.-W. Gu and J. F. Whidborne (1993). Robust control of a glass tube shaping process. *Proc. 2nd European Contr. Conf.* Groningen, Netherlands: pp. 2350-2355.

Narendra, K. S. and K. Parthasarathy (1990). Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks*, vol. 1, no. 1, pp. 4-27.

Nelder, J. A. and R. Mead (1965). A Simplex Method for Function Minimization. *Computer Journal*, vol. 7, pp. 308-313.

Nerrand, O., P. Rousselragot, L. Personnaz and G. Dreyfus (1994). Training recurrent neural networks: why and how? An illustration in dynamical process modeling. *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 178-184.

Ng, W-Y (1989). *Interactive Multi-objective Programming as A Framework for Computer-Aided Control System Design*. Springer-Verlag.

Niranjan, M and F. Fallside (1988). Neural networks and radial basis functions in classifying static speech patterns. *Technical report CUED/F-INFENG/TR.22*, Cambridge University Engineering Department.

Nye, W. T. (1983). DELIGHT: *An interactive system for optimization-based engineering design*. PhD thesis University of California Berkeley, CA.

Nye, W. T. and A. L. Tits (1986). An application-oriented, optimization-based methodology for interactive design of engineering systems. *Int. J. Control*, vol. 43, no. 6, pp. 1693-1721.

Ogata, K. (1975). *Modern Control Engineering*. Prentice-Hall, New Jersey.

Paddison, J. E., R. M. Goodall, R. M. Bals and G. Grubel (1994). Multi-Objective design study for a maglev suspension controller using the databased ANDECS-MATLAB environment. *Proc. IEEE Symp. on Comp. Aided Contr. Syst. Design (CACSD'94)*. Tuscon, USA: pp. 239-246.

Pantas, A. V. (1998). *Robustness and Economic Considerations in the Design of Multivariable Optimal Controllers*. PhD thesis Centre for Process Syst. Engg, Imperial College London, U.K.

Pareto, V. (1906). *Manuale di Economia Politica*. Milan, Italy: Societa Editrice Libraria.

Parlos, A. G., A. F. Henry, F. C. Schweppe, L. A. Gould and D. D. Lanning (1988). Nonlinear multivariable control of unclear power plants based on the unknown-but-bounded disturbance model. *IEEE Trans. Automatic Control*, vol. 33, no. 2, pp. 130-137.

Patel, R. V. and N. Munro (1982). *Multivariable System Theory and Design*. Pergamon Press, Oxford.

Pati, Y. C. and P. S. Krishnaprasad (1990). Analysis and synthesis of feedforward neural networks using discrete affine wavelet transformations. *Technical Report TR 90-44*, Electrical Engineering Department, University of Maryland at College Park.

Patton, R. J. and J. Chen (1992). Robustness in model-based fault diagnosis. *Concise Encyclopaedia of Simulation and Modelling*, D. Atherton and P. Borne (eds.), Pergamon Press, pp. 379-392.

Patton, R. J. and J. Chen (1993a). A survey of robustness in quantitative model-based fault diagnosis. *Applied Mathematics and Computer Science*, vol. 3, pp. 399-416.

Patton, R. J. and J. Chen (1993b). Optimal selection of unknown input distribution matrix in the design of robust observers for fault diagnosis. *Automatica*, vol. 29, no. 4, pp. 837-841.

Patton, R. J. and G. P. Liu (1994). Robust control design via eigenstructure assignment, genetic algorithms and gradient-based optimization. *IEE Proceedings Control Theory Appl.*, vol. 141, no. 3, pp. 202-208.

Patton, R. J., G. P. Liu and J. Chen (1994). Multiobjective controller design of multivariable systems using eigenstructure assignment and the method of inequalities. *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 4, pp. 862-864.

Patton, R. J., P. M. Frank and R. N. Clark (eds.) (1989). *Fault Diagnosis in Dynamic Systems, Theory and Application*. Control Engineering Series, New York: Prentice Hall.

Patton, R. J., Willcox. S. W.. and Winter, S. J. (1987). A parameter insensitive technique for aircraft sensor fault analysis. *Journal of Guidance, Control and Dynamics*, vol. 10, pp. 359-367.

Polak, E. (1979). Algorithms for a class of computer-aided design problem: a review. *Automatica*, vol. 15, pp. 531-538.

Polak, E. and D. Q. Mayne (1976). An algorithm for optimization problems with functional inequality constraints. *IEEE Trans. Autom. Control*, vol. 21, no. 2, pp. 184-193.

Polak, E., D.Q. Mayne and D. M. Stimler (1984). Control system design via semi-infinite optimisation: a review. *Proc. IEEE*, vol. 72, no. 12, pp. 1777-1794.

Polak, E. and S. E. Salcudean (1989). On the design of linear multivariable feedback systems via constrained nondifferentiable optimization in H^∞ spaces. *IEEE Trans. Autom. Control*, vol. 34, no. 3, pp. 268-276.

Porter, B. and M. Borairi (1992). Genetic design of linear multivariable feedback control systems using eigenstructure assignment, *Int. J. Systems Sci.*, vol. 23, no. 8, pp. 1387-1390.

Postlethwaite, I., J.F. Whidborne, G. Murad and D.-W. Gu (1994). Robust control of the benchmark problem using H_∞ methods and numerical optimization techniques. *Automatica*, vol. 30, no. 4, pp. 615-619.

Powell, D. and M. M. Skolnick (1993). Using genetic algorithms in engineering design optimization with non-linear constraints. *Genetic Algorithms: Proceedings of the Fifth International Conference*, San Mateo, pp. 424-431.

Powell, M. J. D. (1981). *Approximation Theory and Methods*. Cambridge: Cambridge University Press.

Powell, M. J. D. (1987). Radial basis functions for multivariable interpolation: A review. *Algorithms for Approximation*, J. C. Mason and M. G. Cox (eds.), Oxford: Oxford University Press, pp. 143-167.

Qian, S., Y. C. Lee, R. D. Jones, C. W. Barnes and K. Lee (1990). The function approximation with an orthogonal basis net. *Technical Report*, Los Alamos National Laboratory.

Qin, S. Z., H. T. Su and T. J. McAvoy (1992). Comparison of four net learning methods for dynamic system identification. *IEEE Trans. on Neural Networks*, vol. 3, no. 1, pp. 122-130.

Qiu, Z. and J. Gertler (1993). Robust FDI systems and optimization: disturbances and tall fault case. *Proceedings of the 32nd IEEE Conference on Decision and Control*, Texas, U.S.A. pp. 1710-1715.

Rayner, P. J. and M. Lynch (1989). A new connectionist model based on a nonlinear adaptive filter. *Proceedings of the International Conference on acoustics, Speech and signal Processing*, Glasgow, Scotland, pp. 1191-1194.

Richardson, J. T., M. R. Palmer, G. Liepins, and M. Hilliard (1989). Some guidelines for genetic algorithms with penalty functions. *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, pp. 191-197.

Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry*. World Scientific, Singapore.

Roppenecker, G. (1983). Minimum norm output feedback design under specified eigenvalue areas. *Syst. Control Lett.*, vol. 3, pp. 101-103.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organisation in the brain. *Psychological Review*, vol. 65, pp. 386-408.

Rosenbrock, H. H. (1960). An automatic method for finding the greatest or least value of a function. *Comp. J.*, vol. 3, pp. 175-184.

Rosenbrock, H. H. (1970). *State-Space and Multivariable Theory*. London: Nelson-Wiley.

Rumelhart, D. E., G. E. Hinton and J. L. McClelland (1986a). A general framework for parallel distributed processing. *Parallel distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations*, D. E. Rumelhart and J. L. McClelland (eds), Bradford Books/MIT Press, Cambridge, MA.

Rumelhart, D. E., G. E. Hinton and J. L. McClelland (1986b). Learning internal representations by error propagation. *Parallel distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations*, D. E. Rumelhart and J. L. McClelland (eds.), Bradford Books/MIT Press, Cambridge, MA.

Rutland, N. K. (1992). Illustration of a new principle of design: vehicle speed control. *International Journal of Control*, vol. 55, no.6, pp. 1319-1334.

Safonov, M. G., R. Y. Chiang and D. J. N. Limebeer (1987). Hankel model reduction without balancing – a descriptor approach. *Proc. 26th IEEE Conf. Decision Contr.* Los Angeles, CA.: pp. 112–117.

Safonov, M. G., A. J. Laub and G. L. Hartmann (1981). Feedback properties of

multivariable systems: the role and the use of the return difference matrix. *IEEE Transactions on Automatic Control*, vol. 26, no. 1, pp. 47-65.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. *Genetic Algorithms and Their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pp. 93-100.

Schaffer, J. D., R. A. Caruana, L. J. Eshelman (1990). Using genetic search to exploit the emergent behavior of neural networks. *Physica D*, vol. 42, no. 13, pp. 244-248.

Scherer, C., P. Gahinet and M. Chilali (1997). Multiobjective output-feedback control via LMI optimization. *IEEE Trans. Autom. Control*, vol. 42, pp. 7, pp. 896-911.

Schmitendorgf, W. E., O. Shaw, R. Benson and S. Forrest (1992). Using genetic algorithms for controller design: simultaneous stabilization and eigenvalue placement in a region. *Technical Report no. C592-9*, Dept. Computer Science, College of Engineering, University of New Mexico.

Sen, P. and J. B. Yang (1998). *Multiple Criteria Decision Support in Engineering Design*. Springer-Verlag, London.

Sinha, P. K. (1987). *Electromagnetic Suspension: Dynamics and Control*. London: Peter Peregrinus.

Skelton, R. E. and M. DeLorenzo (1985). Spacestructure control design by variance assignment, *J, Guid.*, vol. 8, no. 4, pp. 454-462.

Skogestad, S. and I. Postlethwaite (1986). *Multivariable Feedback Control: Analysis and Design*. Chichester, U.K.: John Wiley.

Spears, W. M. and K. A. De Jong (1991a). An analysis of multi-point crossover. *Foundations of Genetic Algorithms*, J. B. Rawlins (ed.), pp. 301-315.

Spears, W. M. and K. A. De Jong (1991b). On the virtues of parameterised uniform crossover. *Proc. ICGA*, vol. 4, pp. 230-236.

Sprecher, D. A. (1965). On the structure of continuous functions of several variables. *Transactions of the American Mathematical Society*, vol. 115, pp. 340-355.

Steuer, R. E. and E. U. Choo (1983). An interactive weighted procedure for multiple objective programming. *Mathematical Programming*, vol. 26, pp. 326-344.

Stinchcombe, M. and H. White (1989). Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. *Proceedings of the International Joint Conference on Neural Networks*, Washington D.C.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. *Proc. ICGA*, vol. 3, pp. 2-9.

Tabak, T., A. A. Schy, D. P. Giesy, and K. G. Johnson (1979). Application of multiple objective optimization in aircraft control systems design. *Automatica*, vol. 15, pp. 595-600.

Taiwo, O. (1978). Improvement of turbo-alternator response by the method of inequalities. *Int. J. Control*, vol. 27, no. 2, pp. 305-311.

Taiwo, O. (1979). On the design of feedback controllers for the continuous stirred tank reactor by Zakian's method of inequalities. *Chemical Eng. J.*, vol. 17, pp. 3-12.

Taiwo, O. (1980). Application of the method of inequalities to the multivariable control of binary distillation columns. *Chemical Eng. Science*, vol. 35, pp. 847-858.

Tang, K. S., K. F. Man and D.-W. Gu (1996). Structured genetic algorithm for robust H^{∞} control system design. *IEEE Trans. Ind. Electr.*, vol. 43, no. 15, pp. 575–582.

Tate, D. M. and A. E. Smith (1993). Expected allele convergence and the role of mutation in genetic algorithms. *Proc. ICGA*, vol. 5, pp.231-237.

Toivonen, H. T. and P. M. Makila (1989). Computer-aided design procedure for multi-objective LQG control problems. *International Journal of Control*, vol. 49, no. 2, pp. 655-666.

Tych, W. (1994). Multi-objective optimisation technique for mapping the technical design objectives into the values of the weighting matrices in the linear quadratic regulator design problem. *Technical Report TR-117*, Univ. Lancaster Centre for Research on Environmental Systems and Statistics Lancaster, UK.

Vidyasagar, M. (1985). *Control System Synthesis: A Factorization Approach*. MIT press, Cambridge, MA.

Vidyasagar, M. (1986). Optimal rejection of persistent bounded disturbances. *IEEE Trans. Aut. Control*, vol. AC-31, pp. 527-534.

Vidyasagar, M. (1991). Further Results on the optimal rejection of persistent bounded disturbances. *IEEE Trans. Aut. Control*, vol. AC-36, pp. 642-652.

Vincent, T. L. and W. J. Grantham (1981). *Optimality in Parametric Systems*. New York: Wiley.

Viswanadham, N., J. H. Taylor and F. C. Lucr (1987). A frequency-domain approach to failure detection and isolation with application to GE-21 turbine engine control systems. *Control Theory and Advanced Technology*, vol. 3, pp. 45-72.

Waltz, F. M. (1967). An engineering approach: hierarchical optimization criteria. *IEEE Trans.*, vol. AC-12, pp. 179-180.

Whidborne, J. F. (1992). Performance in sampled-data systems. *IEE Proc.-D*, vol. 139, no. 3, pp. 245-250.

Whidborne, J. F. (1993). EMS control system design for a maglev vehicle - a critical system. *Automatica*, vol. 29, no. 5, pp. 1345-1349.

Whidborne, J. F., D.-W. Gu and I. Postlethwaite (1994a). MODCONS – a MATLAB Toolbox for Multi-Objective Control System Design. *Technical Report 94-26*, Leicester University Engg Dept Leicester, U.K.

Whidborne, J. F., D.-W. Gu and I. Postlethwaite (1996a). Simulated annealing for multi-objective control system design. *IEE Proc. Control Theory and Appl.*, vol. 144, no. 6, pp. 582-588.

Whidborne, J. F. and R. H. Istepanian (2001a). A genetic algorithm approach to designing finite-precision controller structures. *IEE Proc. Control Theory and Appl.*, vol. 148, no. 5, pp. 377-382.

Whidborne, J. F. and R. H. Istepanian (2001b). An evolutionary algorithm approach to the design of finite word-length controller structures. *Digital Controller Implementation and Fragility: A Modern Perspective*, Springer-Verlag, London,

R. H. Istepanian and J. F. Whidborne, Chapter 9, pp. 153-160.

Whidborne, J. F., I. Postlethwaite and D.-W. Gu (1996b). A mixed optimization approach to multi-objective computer-aided control system design. In *Proc. IEEE Symp. on Comp. Aided Contr. Syst. Design (CACSD'96)*. Dearborn, Michigan: pp. 309-314.

Whidborne, J.F., G. Murad, D.-W. Gu and I. Postlethwaite (1995a). Robust control of an unknown plant – the IFAC 93 benchmark. *Int. J. Control*, vol. 61, no. 3, pp. 589–640.

Whidborne, J. F. and G. P. Liu (1993). *Critical Control Systems: Theory, Design and Applications*. Research Studies Press Limited, U.K.

Whidborne, J. F., I. Postlethwaite and D.-W. Gu (1994b). Robust controller design using H_∞ loop-shaping and the method of inequalities. *IEEE Trans. on Contr. Syst. Technology*, vol. 2, no. 4, pp. 455-461.

Whidborne, J.F., I. Postlethwaite and D.-W. Gu (1995b). Multiobjective control system design – a mixed optimization approach. *Recent Trends in Optimization Theory and Applications*, R. P. Agarwal (ed.), vol. 5 of *World Scientific Series in Applicable Analysis* Singapore: World Scientific pp. 467-482.

Whidborne, J. F., J. Wu and R. H. Istepanian (2000). Finite word length stability issues in an ℓ_1 framework. *Int. J. Control*, vol. 73, no. 2, pp. 166-176.

White, B. A., J. King, Y. Patel and B. A. Stacey (1995). Robust control of an ROV. *Proc. 3rd European Contr. Conf.* Rome, Italy: pp. 3857-3862.

Whitehead, B. A. and T. D. Choate (1994). Evolving space-filling curves to distribute radial basis functions over an input space. *IEEE Trans. on Neural Networks*, vol. 5, no. 1, pp. 15-23.

Whitley, D. (1989). The GENITOR algorithm and selection pressure: why rank-based allocations of reproductive trials is best. *Proc. ICGA*, vol. 3, pp. 116-121.

Whitley, D., K. Mathias and P. Fitzhorn (1991). Delta coding: an iterative search strategy for genetic algorithms. *Proc. ICGA*, vol. 4, pp. 77-84.

Wienke, D., C. Lucasius and G. Kateman (1992). Multicriteria target vector optimization of analytical procedures using a genetic algorithm. Part I. Theory, numerical simulations and application to atomic emission spectroscopy. *Analytica Chimica Acta*, vol. 265, no. 2, pp. 211-225.

University Press, Oxford.

Willems, J. L. (1970). *Stability Theory of Dynamical Systems*. Nelson-Wiley, London.

Williamson, D. (1991). *Digital Control and Implementation: Finite Wordlength Considerations*. Systems and Control Engineering Englewood Cliffs, NJ: Prentice Hall.

Willis, M. J., G. A. Montague, C. Di. Massimo, M. T. Tham and A. J. Morris (1992). Artificial neural networks in process estimation and control. *Automatica*, vol. 28, no. 6, pp. 1181-1187.

Wilson, P. B. and M. D. Macleod (1993). Low implementation cost digital filter design using genetic algorithms. *IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, vol. 1, pp. 4/1-4/8, Chelmsford, U.K.

Wonham, W.M. (1967). On pole assignment in multi-input, controllable linear systems. *IEEE Transactions on Automatic Control*, vol. 12, pp. 660-665.

Wright, A. H. (1991). Genetic algorithms for real parameter optimization. *Foundations of Genetic Algorithms*, J. E. Rawlins (ed.), Morgan Kaufmann, pp. 205-218.

Yang, J. B. (1996). *Multiple Objective Optimisation Methods and Applications.* Hunan Publishing House, Changsha.

Yang, J. B. (1999). Gradient projection and local region search for multiobjective optimisation. *European Journal of Operational Research*, vol. 112, no. 2, pp. 432-459.

Yang, J. B. (2000). Minimax reference point approach and its application for multiobjective optimisation. *European Journal of Operational Research*, vol. 126, no. 3, pp. 90-105.

Yang, J. B., C. Chen and Z. J. Zhang (1988). The interactive decomposition method for multiobjective linear programming and its applications. *Information and Decision Technologies*, vol. 14, no. 3, pp. 275-288.

Yang, J. B., C. Chen and Z. J. Zhang (1990). The interactive step trade-off method (ISTM) for multiobjective optimisation. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 3, pp. 688-695.

Yang, J. B. and P. Sen (1996a). Interactive tradeoff analysis and preference modelling for preliminary multiobjective ship design. *Systems Analysis, Modelling, and Simulation*, vol. 26, pp. 25-55.

Yang, J. B. and P. Sen (1996b). Preference modelling by estimating local utility functions for multiobjective optimisation. *European Journal of Operational Research*, vol. 95, no. 1, pp. 115-138.

Yaniv, O. and I. Horowitz (1991). Ill-conditioned plants : a case study. *Proc. 30th IEEE Conf. Decision Contr.* Brighton, England: pp. 1596–1600.

Youla, D. C., H. A. Jabr, and J. J. Bongiorno (1976). Modern Wiener-Hopf design of optimal controllers-Part 2: the multivariable case. *IEEE Trans. on Autom. Control*, vol. 21, pp. 319-338.

Zadeh, L. A. (1963). Optimality and nonscalar-valued performance criteria. *IEEE Trans. Automat. Contr.*, vol. AC-8, p.1.

Zakian, V. (1979). New formulation for the method of inequalities. *Proc. Instn. Elect. Engrs*, vol. 126, pp. 579-584 (reprinted in *Systems and Control Encyclopaedia*, Pergamon Press, pp. 3206-3215).

Zakian, V. (1983). A criterion of approximation for the method of inequalities. *Int. J. Control*, vol. 37, pp. 1103-1112.

Zakian, V. (1986). A performance criterion. *International Journal of Control*, vol. 43, no. 3, pp. 921-931.

Zakian, V. (1987). Input space and output performance. *International Journal of Control*, vol. 46, no. 1, pp. 185-191.

Zakian, V. (1989). Critical systems and tolerable inputs. *International Journal of Control*, vol. 49, no. 5, pp. 1285-1289.

Zakian, V. (1991). Well matched systems. *IMA J. Math. Contr. & Information*, vol. 8, pp. 29-38.

Zakian, V. and U. Al-Naib (1973). Design of dynamical and control systems by the method of inequalities. *Proc. Inst. Elec. Eng.*, vol. 120, pp. 1421-1427.

Zames, G. (1966). On input-output stability of time-varying nonlinear feedback system, part 1. *IEEE Trans. on Automatic Control*, vol. AC-11, no. 2, pp. 228-238.

Zames, G. (1981). Feedback and optimal sensitivity: model reference transformations, multiplicative seminorms, and approximate inverses. *IEEE Trans. on Automatic Control*, vol. 26, no. 2, pp. 301-320.

Zhou, K., J. C. Doyle and K. Glover (1996). *Robust and Optimal Control*. Upper Saddle River, NJ.: Prentice Hall.

Zhuang, M. and D. P. Atherton (1993). Automatic tuning of optimum PID controllers. *IEE Proceedings-D*, vol. 140, no. 3, pp. 216-224.

Ziegler, J. G. and N. B. Nichols (1942). Optimum settings for automatic controllers. *Trans. ASME*, vol. 64, pp. 759-768.

Index

activation function 262
admissible set 214
allowable eigenvector subspace 203
allowable left eigenvector 204
allowable right eigenvector 204
altitude-hold autopilot 178
analytical optimisation 147
antenna 178
a priori method 78
artificial variable 57
average crossover function 218
average crossover operator 218

binary representation 129
binary string 129

calculus-based technique 125
catalytic converter 178
chromosomal representation 218
complex eigenvalue 204
concave function 61
constrained optimisation 1, 45, 141
conventional multiobjective optimisation 2
convex 34
convex optimisation 6
coprime factorisation 150
critical control framework 179
critical control system 178
critical system 177
crossover 135, 218

decision variable 73
dichotomy method 23, 25
direct-search method 125
distillation column 163
disturbance distribution 287
dual function 61
dual method 60
dual problem 61

efficient solution 74, 75
efficient solution frontier 77
eigenstructure assignment 200
eigenvalue assignment 199
elitist strategy 218
encoding solution 127, 249
evaluation function 127
exterior penalty method 64
external input space 184
ε-constraint method 5

fault isolation 287
feasible direction 46
feasible set 214
Fibonacci method 26
Fibonacci sequence 26
finite word-length 239
fitness 218,246
fitness functions 130
FWL fixed-point representation 241

gasifier 224
genetic algorithms 9, 125, 126
Geoffrion's method 103
global minimum 32
global preference information 77
goal attainment method 5, 89
goal programming 91

golden section number 31
golden section search method 31
gradient projection method 116

Hessian matrix 33
H_∞ theory 149

ideal point method 82
indifference trade-off 117
individual cost function 199
individual eigenvalue sensitivity 210
inflexion point 33
initialisation 130
initial population 218
input space 183
instability 212
interactive method 78, 103
interactive multiobjective programming 6
interactive step trade-off method 112
interior penalty method 68
intermediate recombination 137
internal uncertainty 184
inverted pendulum 156

Kuhn-Tucker condition 47

Lagrange function 50
Lagrangean method 61
Lagrange multiplier 50
left parameter vectors 204
linear programming 45
line recombination 138
local minimum 33
local preference information 77
logarithmic interior penalty method 70
loop shaping design 148, 153

maglev vehicle 167
method of inequalities 7
MIMO critical system 191
MIMO PI controller 229
minimax method 82
minimax reference point method 95
mixed-optimisation 160
modal matrix 203
modelling error space 185
modelling selection 265
moving boundaries process 8
multilayer network 262
multiple objectives 73
multiple objective optimisation 73
multi-point crossover 135
multiobjective genetic algorithms 9
multiobjective control 10
multiobjective critical control 15, 184
multiobjective eigenstructure assignment 16, 210
multiobjective fault detection 19
multiobjective fault diagnosis 290
multiobjective genetic algorithm 245
multiobjective identification 268
multiobjective identification criteria 266
multiobjective nonlinear identification 18
multiobjective optimisation 1
multiobjective optimisation with GAs 141
multiobjective performance function 213
multiobjective PI control 228
multiobjective PID control 16
multiobjective PID controller 239
multiobjective problem 214
multiobjective robust control 14
mutation 138
mutation operator 218

NARMAX model 265
neural networks 261

neuron 262
Newton's method 38
non-dominance 74
non-dominated solution 74, 75
non-inferior solution 74, 75
nonlinear optimisation 23
nonlinear programming 6, 45
nonlinear system identification 259
non-Pareto optimisation 142
null space 206

objectives of fault diagnosis 286
observer based fault diagnosis 284
one-dimensional optimisation 23
optimal indifference trade-off 118
optimality conditions 46
optimal-tuning PI control 230
orthogonal triangular decomposition 206
output performance 184
overall eigenvalue sensitivity 210
over-attainment 93

parameterisation of fault diagnosis 288
parameter optimisation 147
parametric eigenstructure assignment 208
Pareto-based optimisation 143
Pareto optimality 3
Pareto-optimal set 3
Pareto-optimal solution 4, 74, 75
penalty coefficient 64
penalty method 64
plant uncertainty 151
p-norm method 82
population representation 129
posterior method 78
primal method 52

quadratic approximation 39
quasi-Newton's method 41

radial basis function 263
ranking 246
real-valued representation 129
reinsertion 140
residual generator 285
Riccati equation 152
right parameter vectors 204
robust control 147
robust fault diagnosis 282
robustness 212
robust stabilisation 151
robust stability 185
roulette wheel selection 133

saddle point 50
saturated constraint 46
selection 132
sensitivity function 151, 212
sequential linear programming 52
sequential quadratic programming 55
sequential unconstrained minimisation techniques 64
simple weighting method 78
Simplex method 52
single-point crossover 135
singular value decomposition 206
singular value technique 212
SISO critical system 186
small gain theorem 149
spectral matrix 203
stability 212
stationary point 33
steel rolling mill system 249
steepest decent method 34
STEM method 108
stochastic universal sampling 134
surplus variable 56

termination 141
trade-off surface 77

unconstrained optimisation 34
under-attainment 93
uniform crossover 136

utility function 77

weakly efficient solution 75
weighted sum method 5
weighting method 78
well posedness 181