AN EXPERT SYSTEM ENVIRONMENT FOR ROBUST CONTROLLER DESIGN

Thesis submitted for the degree of

Doctor of Philosophy

at the University of Leicester

by

Song-Joo Goh, BEng (Leicester)

Department of Engineering

University of Leicester

August 1996
To my wife Bee-Leng Tan and our parents
ABSTRACT

This thesis considers several important issues in the implementation of an expert system for robust controller design. The work focuses on the architecture of the expert system, robust controller design methods, implementation of knowledge bases and issues concerning the integration of these to form a useful robust controller design environment. The main contribution of this thesis is the development of an expert system package to support industrial control engineers in solving their controller design problems. The expert system captures the experiences and skills of human experts, leaving the user of the expert system free to concentrate more on the creative aspects of the controller design. The expert system makes use of recent results on the $H_{\infty}$ loop-shaping design procedure (LSDP) of McFarlane and Glover. A two degree of freedom loop-shaping design procedure (TDF-LSDP) of Limebeer and co-workers is also considered to enhance the capability of the expert system. Based on these two LSDP approaches, a systematic controller design procedure has been developed in the expert system for tackling generic control system problems. In addition, the relationships between the closed-loop time/frequency domain performance specifications and the open-loop singular value requirements were also considered. Methods for performing systematic loop shaping design based on the closed-loop time/frequency domain specifications are incorporated into rule knowledge bases of the expert system. The organization and handling of control related data by a frame system are explicitly described. Several numerical optimization techniques are also examined and incorporated to enhance the robust control system design approach used in the integrated environment. A systematic controller design procedure using Multi-Objective Genetic Algorithm is presented. A Multi-Layer Multi-Objective Genetic Algorithm is developed for designing controllers with varying structures. A number of case studies are carried out to demonstrate the feasibility of the expert system package.
ACKNOWLEDGEMENTS

Firstly, I would like to express my deepest gratitude to Dr. Da-Wei Gu, my supervisor, for his guidance and encouragement throughout the course of this research. His thorough reading of this thesis are gratefully acknowledged. I would also like to thank my co-supervisor, Professor Ian Postlethwaite, for his time in reading this thesis, despite of his busy time table.

I would like to express my gratitude to my wife, Bee-Leng, for her love, support and encouragement throughout the years of my research. I would like to thank our parents for their continuous care and concern for my study.

I have also benefited greatly from many conversations in our research group. In particular to Dr. Andrzej Bartoszewicz, Dr. Chris Edwards, Dr. Tim Fitzpatrick, Dr. Xiao-Yun Lu, Dr. Ghassan Murad and Dr. James Whidborne for their helpful and enlightening discussions throughout my stay in Leicester University. I would like to express my gratitude to Mr. Paul Carter for his time in proof-reading this thesis and many helpful suggestions. In addition, I am very grateful to Mr. Declan Bates, Mr. Aamer Bhatti, Mr. Fu-Wah Poon and Mr. Alex Smerias in showing their interest in the expert system and many interesting discussions.
Contents

1 INTRODUCTION
1.1 A General Overview .................................................. 1
1.2 Artificial Intelligence and Expert Systems ..................... 3
1.3 An Overview of Robust $H^\infty$ Control ....................... 5
1.4 History of Expert System in CACSD ............................... 8
1.5 Structure of Thesis ................................................... 10
1.6 Notations .................................................................. 12

2 EXPERT SYSTEMS ..................................................... 16
2.1 Introduction ................................................................ 16
2.2 Knowledge Representation .......................................... 16
2.3 Rule-Based Systems .................................................. 18
2.3.1 Introduction ...................................................... 18
2.3.2 Working Memory ............................................... 21
2.3.3 Inference Engine and Rule Knowledge Base .............. 21
2.3.4 Forward Chaining ............................................... 22
2.3.5 Backward Chaining ............................................. 25
2.3.6 Programmer Interface ........................................... 27
2.3.7 User Interface .................................................... 27
2.3.8 External Software Interface ................................. 28
2.3.9 Characteristics of Rule-Based Expert Systems .......... 28
2.4 Frame Systems ....................................................... 30
2.4.1 Definition of a Frame ........................................... 30
4.4 A Systematic Controller Design Using The One Degree Of Freedom Loop
Shaping Approach ................................................................. 79
4.4.1 Loop Shaping of Open-Loop Singular Values .................. 80
4.4.2 Robust Stabilization ..................................................... 81
4.4.3 Construction of the Controller and the Closed-Loop System .. 81
4.5 A Systematic Controller Design Using Two Degree Of Freedom Loop
Shaping Approach ................................................................. 81
4.6 Rule Knowledge Bases For The One Degree Of Freedom LSDP Design . 83
4.6.1 Open Loop Frequency Domain Design Rule Knowledge Base .. 83
4.6.2 Closed-Loop Frequency Domain Design Rule Knowledge Base .. 85
4.6.3 Closed Loop Time Domain Design Rule Knowledge Base ...... 87
4.7 Rule Knowledge Bases For The Two Degree Of Freedom LSDP Design . 88
4.7.1 Time Domain Control Signal Tuning Rule Knowledge Base ... 89
4.7.2 Closed-Loop Time Domain Design Rule Knowledge Base .... 90
4.7.3 Closed-Loop Frequency Domain Design Rule Knowledge Base .. 91
4.8 Overall Design Flow ........................................................ 92
4.9 Summary ................................................................. 94

5 EXPERT SYSTEM IMPLEMENTATION ........................................... 95
5.1 Introduction ................................................................. 95
5.2 Selection Of The Expert System Shell .................................. 95
5.3 Structure of the Expert System .......................................... 98
5.4 User Interface .............................................................. 99
5.5 Implementation Of Numerical Computation Engine .................. 102
5.6 Data Organization With Frame Knowledge Base .................... 105
5.6.1 Representation of Data in Frames Knowledge Base ............. 105
5.6.2 Routines for Frames Data Processing .............................. 107
5.7 Rule Knowledge Base And Inference Mechanism .................... 108
5.7.1 Test for the Improvement of a Design Step ......................... 111
5.7.2 Actions after a Bad Design Step .................................... 112
5.7.3 Complete Rule Firing Control ........................................ 116
5.8 Communication with External Programs ....................................................... 118
5.9 Summary ........................................................................................................ 120

6 NUMERICAL TECHNIQUES FOR MULTIOBJECTIVE OPTIMIZATION ............122
6.1 Introduction ........................................................................................................ 122
6.2 The Method of Inequalities .............................................................................. 123
    6.2.1 Moving Boundary Process ........................................................................ 126
    6.2.2 $H^\infty$ Loop Shaping Design with the Method of Inequalities ............ 127
6.3 Genetic Algorithms .......................................................................................... 130
    6.3.1 Motivation .................................................................................................. 130
    6.3.2 Multi-Objective Genetic Algorithm (MOGA) ........................................ 132
    6.3.3 MOGA with LSDP Controller Design Technique .................................. 138
    6.3.4 Multi-Layer Multi-Objectives Genetic Algorithm (MLMOGA) .......... 141
    6.3.5 MLMOGA with LSDP Controller Design Technique ......................... 142
    6.3.6 MLMOGA with TDF-LSDP Controller Design Technique ................... 144
6.4 Design Example .............................................................................................. 147
    6.4.1 Plant Models ............................................................................................ 147
    6.4.2 Specifications ........................................................................................... 149
    6.4.3 Closed-loop Performance Functionals .................................................... 149
    6.4.4 Controller Designs and Simulations ......................................................... 150
    6.4.5 Design Analysis and Conclusions ............................................................ 154
6.5 Summary and Discussion .................................................................................. 155

7 DESIGN EXAMPLES .........................................................................................162
7.1 Introduction ....................................................................................................... 162
7.2 The CH-47 System ........................................................................................... 163
    7.2.1 LSDP One Degree of Freedom Design ................................................. 165
    7.2.2 LSDP Two Degree of Freedom Design ................................................ 169
7.3 The Bell-205 Helicopter ................................................................................... 172
    7.3.1 LSDP One Degree of Freedom Design ................................................ 176
Chapter 1

INTRODUCTION

1.1 A General Overview

In this thesis we examine the interaction between two important areas namely the development of an integrated expert system and the design of robust control systems. An expert system which is capable of designing a robust controller for a linear multivariable plant is the result of this work. The aim of this project is to provide a control system design environment which supports industrial control engineers.

Control systems are in most cases subjected to the presence of parameter variations, external disturbances, measurement noise and unmodelled system dynamics. The robust control problem is to design a controller such that the closed-loop system is stable and simultaneously meets a set of performance criteria in the presence of such uncertainties. In the past two decades, there have been significant advances both in theory and in practice of the robust multivariable feedback control system design. At the same time, a wide range of Computer Aided Control Systems Design (CACSD) software environments has emerged to alleviate the use and the understanding of these state of the art control system design approaches. The major aim of developing the CACSD software environments is to make such a specialized field of knowledge easily applicable and sharable within the community of control practitioners. The combination of advanced controller design techniques and software tools offers a valuable and powerful design framework for performing control system design. Nevertheless, to apply control system design techniques successfully using the available software tools, the user is still
required to be reasonably familiar with the relevant theory and the formal syntax of the CACSD packages. Therefore, a gap is yet to be filled between the design framework and the user. It is our belief that expert systems have a key role to play in bridging the gap between the powerful design framework and its potential users.

An expert system which possesses sufficient knowledge of advanced robust control system design techniques in a knowledge base, together with some appropriate functions and procedures to perform numerical computations, can mimic the behaviour of a human expert in the robust control system design.

![Diagram](image)

**Figure 1.1: The Role of An Expert System For Robust Control System Design**

The practising control engineer should be able to access the advanced control system design techniques via the user interface of the expert system as illustrated in Figure 1.1. With the knowledge base overseeing the procedures of the control system design software for performing numerical computations, the expert system can relieve practising control engineers from remembering the complicated formal syntax of control system design software and any other complex procedures involved in using the functions of the software. With the expert system, the control engineer can concentrate on the trade-offs
Chapter 1. Introduction

in the design which are crucial in applying any design techniques successfully to control system problems.

On the other hand, the procedures and design approaches used by the expert system must be coherent with the familiar principles used by the control system designer on the task of controller design. In order to achieve this key attribute of coherency, the design approaches used by the expert system should be based on an appropriate technique which is familiar to its user. With this key factor in mind, this work emphasizes the recent results from the $H_\infty$ loop-shaping design procedure (LSDP) [46]. This design methodology allows the design of multivariable control systems by initially shaping the open-loop singular values of the nominal plant, and an optimal $H_\infty$ controller is then synthesized to robustly stabilize the shaped plant based on its normalized coprime perturbation.

In this chapter, we present the relevant introductory materials of expert systems, robust control design methods and CACSD environments. We begin with a brief history of artificial intelligence and expert systems in Section 1.2, which is followed by an overview of the recent progress in robust control theory in Section 1.3. A history of successful implementations of expert system techniques for control system design is presented in Section 1.4. The contribution and the structure of the thesis are described in Section 1.5. The notation which will be used throughout the thesis is listed in Section 1.6.

1.2 Artificial Intelligence and Expert Systems

Although the digital computer was initially intended to be a high speed data processor for performing numerical computations, in the early days of its creation, there was a small group of computer scientists devoted to create algorithms and software which attempted to emulate some of the activities of the human mind. In 1956, the members of this group attended a summer workshop sponsored by IBM at Dartmouth College to discuss ways of directing their work to develop both hardware and software which could mimic human reasoning. The workshop marked the emergence of a new branch of the study of computer science, Artificial Intelligence (AI), the field of creating intelligent machines which are capable of reasoning in a manner similar to human be-
Chapter 1. Introduction

The initial focuses of AI were on theorem proving and general problem solving. AI researchers started to develop computer programs based on complex search algorithms which have good general problem solving capabilities, independent of any specific problem domain. At that stage, researchers believed that intelligent behaviour was primarily reliant on smart reasoning techniques. Nevertheless, most of the earlier programs failed to produce a meaningful system that solved real-world problems. It was not until the mid 60s, when researchers began to realise that search techniques alone would not be enough to produce an intelligent program, that the research attention was shifted towards the quest for methods to represent knowledge and to code it into the computer in symbolic forms. In the late 60s, a program named DENDRAL [7, 6] was developed at Stanford University at the request of NASA to perform chemical analysis of the Martian soil. The system was developed by E.A. Feigenbaum and B.G. Buchanan in conjunction with an organic chemist, J. Lederberg. They captured the heuristics from many knowledgeable chemists into their program for recognizing molecular structures of unknown compounds. DENDRAL was the first program whose success was attributed to heuristic knowledge representation, rather than complex search techniques. The emphasis on knowledge led to the concept of Knowledge Based Systems (KBS) or Expert Systems (ES).

Two major lessons were learnt from the work on DENDRAL to develop expert systems in the early 70s. The first was the importance of having a rich source of knowledge embedded within the knowledge bases of the expert system. The second lesson was concerned with the scope of the knowledge in an expert system. A knowledge base which was encoded with well focused knowledge scored far better than one that addressed broad problems. These two key factors naturally lead the programmer towards a crucial source when developing an expert system, an expert on the problem. An individual is regarded as an expert if he or she possesses a superior understanding of the problem. The expert gains skills through experience to enable him or her to solve the problem effectively and efficiently. The task of the knowledge engineer is to capture the expertise of the expert into the expert system.
1.3 An Overview of Robust $H^\infty$ Control

In the 60s, the multivariable control system design was developed with an emphasis on achieving good performance but not on robustness. Most of these design techniques were based on linear quadratic performance indices and Gaussian disturbances i.e. the Linear Quadratic Gaussian (LQG) method. The application of this method had proved to be successful in the field of aerospace. The main reason for the success was that accurate mathematical descriptions of aerospace vehicles could be derived and the assumption of external disturbances based on white noise was appropriate in such applications. However, Doyle [11] later showed that LQG design can exhibit arbitrarily poor robustness properties and stability margin when applied to other industrial problems. This motivated the development of other control theories to explicitly address the issue of robustness in the multivariable feedback design.

A frequency domain theory known as the $H^\infty$ optimization technique was originally formulated by Zames and Francis [81] in the early 80s. Within the $H^\infty$ optimization framework, model uncertainty and performance specification in the frequency domain can be incorporated into the controller design procedure. A constraint optimization is then performed to optimize the robust stability and performance specification of the closed-loop system with the constraint being that the feedback system is internally stable. The performance and robustness objectives for these designs are generally taken to be the minimization of the weighted sensitivity function $S$ and the weighted complementary sensitivity function $T := I - S$. The transfer function $KS$ can also be weighted to penalize the control energy, with $K$ being the $H^\infty$ controller. These design approaches are thus commonly referred to as the $S/T/KS$ mixed sensitivity techniques.

The $H^\infty$ optimization technique has proved to be an attractive framework for analyzing robustness as well as an effective tool for designing control systems with unstructured uncertainty. Design using the $H^\infty$ optimization approach was further alleviated by the emergence of various control system design software packages such as MATLAB$^\text{1}$.

In addition, an efficient state space solution of the $H^\infty$ optimization was developed by Doyle et al.[12]. It only requires the solution of two algebraic Riccati equations, and

---

$^1$MATLAB is a registered trademark of The MathWorks, Inc.
resulted in a controller of state dimension equal to or less than that of the weighted plant. The major disadvantage of this approach is that, there is no clear methodology for selecting the weights in the problem formulation stage. The appropriate selection of the weights to reflect the desired closed-loop performance objectives and the robustness against model uncertainty is generally not straightforward and often requires a thorough understanding of the plant. This may not be simple for a novice practical control engineer to achieve. As a result, the $H^\infty$ optimization design approach is mostly used by the researchers from academia for industrial case studies. Furthermore, pole-zero cancellation may occur between the plant and the controller. All the stable poles of the open-loop plant are cancelled by the controller and the closed-loop poles may include the mirror image positions (in the imaginary axis) of all the unstable poles of the plant.

To overcome the drawbacks of the mixed sensitivity approach, an optimization Loop Shaping Design Procedure (LSDP) was developed by McFarlane and Glover [46, 47]. Using the LSDP, a robust controller can be synthesized systematically based on the multivariable generalization of the classical loop shaping ideas. The open-loop singular values of the plant are shaped by selecting some appropriate weighting functions. The shaped plant is then robustly stabilized by an $H^\infty$ controller and the final controller can be constructed by combining the $H^\infty$ controller with the weights. With the LSDP approach, the loop shaping process is done without the need to explicitly consider the closed-loop stability requirement and the $H^\infty$ controller can be synthesized without having to specify any frequency weight once the plant is shaped. The LSDP approach also overcomes the problem of pole-zero cancellation between the plant and the controller, except for a special class of plants which contains stable all-pass factor.

The LSDP approach was further developed into a two degree of freedom scheme by Limebeer et al.[41]. The Two Degree of Freedom LSDP (TDF-LSDP) allows the design of a controller which guarantees a prescribed level of robust stability while simultaneously forcing the closed-loop transfer function to approximate a given model. With the TDF-LSDP, time domain specifications can be incorporated into the design procedure. In an alternative development direction, a mixed optimization design procedure based on the LSDP and the Method of Inequalities (MOI) framework was developed by Whid-
Chapter 1. Introduction

borne et al.[76]. In the mixed optimization approach, the robust design problem is transformed into a constrained optimization problem with the weighting functions as design parameters. All the design constraints are formulated as a set of inequalities which need to be satisfied for a successful design. Closed-loop time domain specifications (such as rise time, settling time and overshoot) as well as the functionals of frequency domain specifications (such as bandwidth) can be included as the set of constraints. Solutions which satisfy all the inequalities can be found by using simple hill-climbing parameter search algorithms such as Moving Boundary Process (MBP). In general, hill-climbing search algorithms work well with a simple problem with few local minima. They are also dependent upon being provided with a good starting point. However, the constrained optimization problem is generally non-convex, non-smooth and multi-objective. To find the global solution to such optimization problem is often a difficult task. For problems which have many local minima, algorithms with probabilistic transition rules can offer a better choice of finding the global minimum point. One of such approaches uses the Genetic Algorithms (GAs) technique.

The GAs apply operators inspired by the mechanics of natural selection such as crossovers, mutations and reproductions, to a population of binary string encoding of the parameters set which can be non-differentiable and non-convex. By simultaneously evaluating a number of population points in the parameters set, GAs can effectively reach many local minima and thereby increase the likelihood of finding the global minimum point.

A Multi-Objective Genetic Algorithms (MOGA) as a modification of the standard genetic algorithms at the selection level has been developed to solve the multi-objects optimization problems [9]. Although computationally demanding, the combination of MOGA and LSDP has been found to be very useful in searching for the parameters of a fixed structure controller to meet a set of stringent closed-loop performance objectives. An improvement of MOGA, the Multi-Layer Multi-Objective Genetic Algorithms (MLMOGA) is developed by Goh et al.[24]. The MLMOGA differs from the MOGA in its hierarchical chromosome structure. Each chromosome in the MLMOGA has two levels of genes. The top level consists of the control genes which contain information of the order and the type of the weighting functions. The bottom level consists of the data...
Chapter 1. Introduction

genes which define the parameters of the weighting functions. Genetic operations can be applied to both levels of chromosomes so that both the order and the parameters of the weighting functions can be optimized.

Despite the rapid development of advanced control techniques, practising control engineers rarely use them for practical industrial problems. A thorough understanding of the theory is crucial for a successful application of advanced control techniques. Moreover, it is very difficult for industrial engineers to keep up with the continuous development of software for control system design. The usefulness of expert systems in assisting practising control engineers to perform control system design is becoming apparent. There has been a rapidly growing interest in applying the technology developed in the study of Artificial Intelligence (AI) to the field of control system design. In the next section, we present a brief history of some efforts from the control society to build expert systems for Computer Aided Control System Design (CACSD).

1.4 History of Expert System in CACSD

The research of the application of expert system techniques to CACSD was initiated in the early 80s when Taylor et al. [36] presented an overview of the application of expert systems to control engineering. They outlined the wide range of activities which should be addressed while designing a control system. Meanwhile, Birdwell [4] addressed some important issues in the development of a CACSD software package.

Taylor and Frederick [35] later demonstrated the use of an expert system for control system design by building a rule knowledge base expert system for performing SISO control system design. Their expert system focused on the use of classical frequency domain controller design techniques and used the subroutines from the Cambridge Linear Analysis and Design Program [15] for numerical computations. James et al. [37] presented an expert system which automatically designed a compensator for a given plant, leaving the user with little scope for interaction during the controller design process. During the same period, Birdwell et al. [16] designed an expert system which was capable of performing analysis and design for linear multivariable system using linear quadratic regulator (LQR) theory and Kalman-Bucy filter (KBF) design techniques to satisfy the
frequency-domain measures of performance. Their expert system employed the loop
transfer recovery (LTR) technique to maintain the overall performance characteristics
in the final design.

Trankle et al. [68] attempted to build an expert system for control system design
by using the ideas of a planning system to organize the process of controller design
into two levels. The higher level concentrated on developing an overall strategy by
building a list of goals to be accomplished. The lower level was made up of specific
computer-aided control system design functions for achieving these goals. Nolan [52]
constructed an expert system to be an assistant to a control system designer. His expert
system was used in parallel with a control system design package such as MATLAB for
numerical computation required for analysis and synthesis of control systems. Pang
et al. [55, 56, 54] built an expert system called MAID to assist the user to design a
controller for a linear multivariable system by executing a structured set of multivariable
frequency domain design techniques.

In the late 80s and the early 90s, expert systems were developed in a growing extent
to aid users of various levels of sophistication to design controllers. Lewin and Morari
[39] presented an expert system for robust controller synthesis. Their expert system can
help the novice users in designing controllers for processes which only have approximate
models. Tebbutt [67] implemented an expert system for the design of linear multi­
variable control systems which operated on a personal computer. It helped the user in
formulating an achievable design specification, and in dealing with conflicting design
constraints.

As the application of expert systems to CACSD becomes more mature, most of the
recent research emphasizes the implementation issues of expert systems and the data
organization within the knowledge representation framework. Moreover, an Intelligent
Front End (IFE) has also been recognized as important in transforming the expert sys­
tems to be more user-friendly. Pang [53] had initiated the pioneering work on developing
an IFE for control system design and analysis packages. An IFE to assist the user to
systematically design an $H^\infty$ controller for a control system was later reported by Pang
et al.[57]. The object-oriented and structural approach to organize data and knowledge
used for the control system design is beginning to gain popularity as the scope of control problems becomes larger. Barker et al. examined the application of object-oriented database management systems to computer-aided control engineering [1].

This thesis describes research in the University of Leicester which explores the application of the expert system techniques to aid practising control engineers to design robust controllers. This approach is introduced in [58] and the architecture of the expert system is presented in [17]. Details of internal data organization within the knowledge bases are presented in [27]. An application of the expert system for robust controller design is presented in [28].

1.5 Structure of Thesis

The main contribution of this thesis is the development of an expert system package for robust controller design which can benefit practising control engineers from industry, as well as academic researchers. The software package is used for the analysis and design of robust controllers. In the software development process, we also explore the approximation of the closed-loop performance specifications from the open-loop frequency domain requirements. The relationship between the closed-loop specifications in the time/frequency domain and the open-loop frequency response is essential in any frequency response design techniques for feedback control systems. A systematic controller design procedure which comprises one degree of freedom and two degree of freedom $H^\infty$ loop shaping techniques is also shown in the thesis to handle generic robust control problems. Several numerical optimization techniques have been incorporated into the $H^\infty$ loop shaping techniques to design controllers which satisfy a set of closed-loop performance specifications. An improved genetic algorithm is also developed for handling multi-objective optimization criteria in the controller design.

Chapter 2 introduces some useful concepts about the expert system techniques and design formulation. Attention is focused on two important classes of expert systems: Rule-Based Systems and Structural Knowledge Bases (more commonly known as Frame Systems). All the basic concepts presented in this chapter will be used later to build an expert system for performing robust controller design. It begins by stressing
the importance of choosing the knowledge representation scheme, and proceeds with a
detailed discussion of the general architecture of the rule knowledge base and the frame
knowledge representation.

Chapter 3 lists the preliminary mathematics for robust controller design theories based
on the well known $H_{\infty}$ optimization approach. It starts with a summary of the different
approaches for modelling plant uncertainty and the corresponding robust stabilization
problems. A brief introduction to designing robust controllers using the standard
$H_{\infty}$ optimization framework is then given. Finally, two approaches of $H_{\infty}$ Loop Shaping
Design Procedure based on the normalized coprime plant uncertainty representation are
presented.

Chapter 4 describes the development of the expert system for robust controller design.
It begins by giving the definition of the structural data representation for handling the
control system design data. The heuristic of the singular values loop shaping design is
given, followed by the outlines of systematic controller design procedures using two $H_{\infty}$
loop shaping approaches. Based on the two systematic controller design procedures,
rule knowledge bases are used to capture the controller design heuristic. Finally, an
overall design flow is then discussed to show the linkage of all the rule knowledge bases
and to devise a generic systematic design procedure.

Chapter 5 discusses the practical issues of the implementation of the expert system for
robust controller design using an expert system shell. A guideline for the selection of
an expert shell is presented, and the structure of the expert system is then defined.
The components of the expert system are then discussed in detail from the aspect of
practical implementations and the philosophy behind the motivation of building and
linking up each component. Representation of data in the frame system is presented
with the classification of control related data and routines for frames data processing.
The implementation issues of rule knowledge base along with the inferences mechanism
are also examined whereby attentions are focused on the rule firing control and the test
for the improvement of each design step. As a result, a fuzzy type of cost evaluation for
the rule firing mechanism is presented.

Various numerical techniques for multi-objective optimization are presented in Chapter
Chapter 1. Introduction

6. An effective optimization technique known as the Method of Inequalities is discussed, followed by two multi-objective parallel optimization approaches employing Genetic Algorithms. Two parallel optimization techniques are incorporated into the $H_\infty$ loop shaping approaches to form a mixed optimization approach for robust controller design. The 1993 IFAC world congress benchmark problem is used to test the three optimization approaches and a comparative study is made.

Several case studies are examined in Chapter 7 using the expert system, in order to demonstrate the feasibility of this controller design package.

Finally, Chapter 8 draws a conclusion of the thesis and provides views on possible extensions to the work presented in the thesis.

1.6 Notations

$a \in S$ $a$ is an element of set $S$; $a$ belongs to $S$

$\exists a \in S$ there exists an element $a$ of set $S$

$\forall a \in S$ for every element $a$ of set $S$

$S_1 \cap S_2$ intersection of sets $S_1$ and $S_2$

$S_1 \cup S_2$ union of sets $S_1$ and $S_2$

$S_1 \subseteq S_2$ set $S_1$ is contained in set $S_2$

$p \implies q$ $p$ implies $q$

$p \equiv q$ $q$ implies $p$

$p \iff q$ $p$ if and only if $q$; equivalently, $p$ implies $q$ and $q$ implies $p$

$A := B$ $A$ is equal to $B$ by definition

$A ::= B$ $B$ is equal to $A$ by definition

$a \ll b$ $a$ is much less than $b$

$a \gg b$ $a$ is much greater than $b$

$a \approx b$ $a$ is approximately equal to $b$

$\forall$ for all

$\exists$ such that

$C$ space of complex numbers

$\mathbb{R}$ space of real numbers
Chapter 1. Introduction

$C^n$ space of complex n-dimensional vectors

$\mathbb{R}^n$ space of real n-dimensional vectors

$C^{n \times m}$ space of $n \times m$ complex matrices

$\mathbb{R}^{n \times m}$ space of $n \times m$ real matrices

$j = \sqrt{-1}$; sometimes an index, as in $x_{ij}$

$\Re(z)$ real part of complex number $z$

$\Im(z)$ imaginary part of complex number $z$

$\bar{z}$ complex-conjugate of complex number $z$

$|z|$ modulus of complex number $z$

$\angle z$ argument of complex number $z$

$I$ identity matrix of appropriate dimension

$I_n$ $n \times n$ identity matrix

$A^{-1}$ inverse of the square matrix $A$

$A^T$ transpose of the matrix $A$

$A^*$ complex-conjugate transpose of the matrix $A$

$|A|$ nonnegative matrix; comprising the moduli of the entries of the matrix $A$

$A > 0$ matrix $A$ is positive definite

$A \geq 0$ matrix $A$ is semi-positive definite

$a_{ij}$ the $(i, j)$ element of the matrix $A$

$\text{diag}[x_1, x_2, \ldots]$ diagonal matrix with diagonal elements $x_1, x_2, \ldots$

$\det(A)$ determinant of the square matrix $A$

$\text{tr}(A)$ trace of the square matrix $A$

$c(A)$ condition number of the matrix $A$

$\rho(A)$ spectral radius of the square matrix $A$

$\lambda_i(A)$ ith eigenvalue of the square matrix $A$

$\lambda(A)$ maximum eigenvalue of the square matrix $A$

$\Delta(A)$ minimum eigenvalue of the square matrix $A$

$\sigma_i(A)$ ith singular value of the matrix $A$

$\delta(A)$ maximum singular value of the matrix $A$
Chapter 1. Introduction

\( \sigma(A) \) minimum singular value of the matrix \( A \)

\( \Delta \) model uncertainty

ess sup essential supremum

sup supremum, i.e. least upper bound (l.u.b.)

inf infimum, i.e. greatest lower bound (g.l.b.)

max maximum

min minimum

\( \mathcal{L}[g(t)] \) Laplace transform of \( g(t) \)

\( \|x\|_p \) \( p \)-norm of the vector \( x \), \( 1 \leq p < \infty \)

\( \|x\|_\infty \) \( \infty \)-norm of the vector \( x \)

\( \mathcal{L}^\infty(\mathcal{R}) \) set of essentially bounded measurable functions

\( T : A \to B \) \( T \) is an operator (or a function) mapping from \( A \) to \( B \)

\( \|T\|_1 \) induced norm of the operator \( T \)

\( \|T\|_1 \) induced 1-norm of the operator \( T \)

\( \|T\|_2 \) induced 2-norm of the operator \( T \)

\( \|T\|_\infty \) induced \( \infty \)-norm of the operator \( T \)

\( \|M\|_F \) Frobenius norm of the matrix \( M \)

\( \|G\|_\infty \) := \( \sup_{\omega \in \mathbb{R}} \sigma[G(j\omega)] = \|T\|_2 \), the \( \infty \)-norm of the transfer function matrix \( G(s) \) of the system operator \( T \)

\( H^\infty \) set of stable matrix-valued functions \( G(s) \) with \( \|G\|_\infty < \infty \)

\( RH^\infty \) set of real-rational functions in \( H^\infty \)

\( F(M, \Delta) \) lower linear fractional transformation on \( M \) by \( \Delta \)

\( F_u(M, \Delta) \) upper linear fractional transformation on \( M \) by \( \Delta \)

\( \dot{x}(t) \) := \( \frac{dx(t)}{dt} \)

\( \frac{dx}{dt} \) the differentiation with respect to \( t \)
### Chapter 1. Introduction

**Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e.g.</td>
<td>for example</td>
</tr>
<tr>
<td>i.e.</td>
<td>that is</td>
</tr>
<tr>
<td>resp.</td>
<td>respectively</td>
</tr>
<tr>
<td>r.m.s.</td>
<td>root-mean-square</td>
</tr>
<tr>
<td>BIBO</td>
<td>Bounded-Input Bounded-Output</td>
</tr>
<tr>
<td>FDLTI</td>
<td>Finite-Dimensional, Linear and Time-Invariant</td>
</tr>
<tr>
<td>I/O</td>
<td>Input-Output</td>
</tr>
<tr>
<td>LFT</td>
<td>Linear Fractional Transformation</td>
</tr>
<tr>
<td>LHP</td>
<td>Left-Half Plane</td>
</tr>
<tr>
<td>LLFT</td>
<td>Lower Linear Fractional Transformation</td>
</tr>
<tr>
<td>MIMO</td>
<td>Multi-Input Multi-Output</td>
</tr>
<tr>
<td>NP</td>
<td>Nominal Performance</td>
</tr>
<tr>
<td>NS</td>
<td>Nominal Stability</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional plus Integral plus Derivative</td>
</tr>
<tr>
<td>RHP</td>
<td>Right-Half Plane</td>
</tr>
<tr>
<td>RP</td>
<td>Robust Performance</td>
</tr>
<tr>
<td>RS</td>
<td>Robust Stability</td>
</tr>
<tr>
<td>SCC</td>
<td>Standard Compensation Configuration</td>
</tr>
<tr>
<td>SISO</td>
<td>Single-Input Single-Output</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>ULFT</td>
<td>Upper Linear Fractional Transformation</td>
</tr>
</tbody>
</table>
Chapter 2

EXPERT SYSTEMS

2.1 Introduction

The aim of this chapter is to introduce some useful concepts about expert systems (ES). Various characteristics of expert systems are summarized. Attention is focused on two important classes of expert systems: the Rule-based System and the Frame System. All the basic concepts presented in this chapter will be used in Chapters 4 and 5 to build an expert system for performing robust controller design.

The organization of this chapter is as follows. The importance of the knowledge representation is introduced in Section 2.2, followed by the general architecture of the rule-based system in Section 2.3. Frame systems will be discussed in Section 2.4. Finally, a comparison is drawn between the two methods of representing problem solving knowledge which are described in Section 2.3 and Section 2.4.

2.2 Knowledge Representation

Expert system technology is derived from the discipline of Artificial Intelligence (AI). Barr and Feigenbaum [2] defined the study of Artificial Intelligence as a branch of computer science concerned with designing intelligent computer systems that exhibit characteristics we associate with intelligence in human behaviour such as understanding a language, learning, reasoning, general problem solving and so on. A person is considered as an expert in a specific field when he or she possesses superior and specialized knowledge about the domain. In the study of the expert systems, this type
of knowledge is known as domain knowledge. An expert system can be considered as a computer program which is able to mimic the intelligent behaviour of an expert in performing a particular task. It reasons with the domain knowledge acquired from the human expert with a view to solving problems or providing useful advice. In the early 60s, most of the expert systems relied heavily on very complicated but smart search algorithms. The search for a solution to a particular problem was often very time consuming. None of the early expert systems proved useful in solving real world problems until the discovery of the need to encode the problem solving knowledge into the expert system.

Since the early 70s, there has been growing conviction that the power of a problem solver lies in the explicit representation of knowledge that the problem solver can access, rather than in a sophisticated inference mechanism for the knowledge. Cognitive psychologists have performed research on theories to explain how human beings solve their problems with the aim of uncovering the type of knowledge which is commonly used by human beings and how this knowledge can be organized to solve problems efficiently. Artificial intelligence researchers used results from these studies to develop techniques for representing the different knowledge in the context of symbolic programming.

When building an expert system, one attempts to capture all the human expert's knowledge of a well-focused topic from the subject area. In the field of expert systems, this kind of knowledge is often referred to as domain-specific knowledge. After acquiring domain-specific knowledge from the expert, the next step is to find a method to organize and to structure the knowledge into the expert system so that it can be used to solve problems in a manner similar to that followed by the expert. The various methods which are used to encode knowledge into the knowledge base of an expert system are formally known as Knowledge Representation methods.

The problem of designing an expert system is therefore seen primarily as a problem of building a knowledge base that represents the domain-specific knowledge of an expert. The general problem of knowledge representation is to develop a sufficiently precise formal method for representing a particular domain-specific knowledge. Despite the existence of numerous knowledge representation schemes, the problem of adopting a
knowledge representation method remains as a difficult one. Some domains of knowledge, for example mathematical knowledge, are relatively straightforward to cope with, due to the precise definition of the knowledge involved. Unfortunately, a majority of domain specific knowledge areas do not have simple definition of objects like those found in mathematics. It is therefore very important to consider the types of objects and their relationships in the problem domain when choosing a knowledge representation scheme to capture the problem solving knowledge.

One way to identify different classification of knowledge representation schemes is to consider the problem domain to contain a set of individual objects with a collection of relationships existing between them. The collection of all objects and their relationships at any one time constitutes a state. There can be state transformations that cause the creation or destruction of objects, or that can change the relationship among them. Depending on whether the key starting point for a representation scheme is the objects and their relationships or the state transformations, leads to a frame or production scheme respectively. The remainder of this chapter describes these two knowledge representation schemes in more detail.

2.3 Rule-Based Systems

2.3.1 Introduction

The initial work on the modelling of the human problem solving process to form useful information processing techniques was studied by Newell and Simon [50]. They classified the memory stored in the human brain into two categories: the long-term memory and the short-term memory. The long-term memory can be moulded by a set of situation-action rules which are called production rules. The short-term memory consists of numerous situations or specific information and facts about a particular problem to be solved. While solving a problem, human beings apply production rules repeatedly to the situations in the short-term memory and derive new facts by carrying out the action parts of the production rules continuously changing the state of the short-term memory until the problem is solved. In short, human beings reason by inferring new
information based on existing facts until the state of the short-term memory matches the state of a solution. This sort of reasoning process which invokes production rules to change the state of the short-term memory is commonly known as the production cycle.

The model that mimics the human problem-solving process as a series of production cycles is referred to as the production system. Figure 2.1 illustrates a model of a typical production system.

The concise and effective principles of the production system form the basis of the rule-based expert system. In the literature of expert systems, production rules are sometimes called condition-action rules or situation-action rules. This is because they are usually used to encode empirical associations between the patterns of data presented to the system and the actions that the system should perform as a consequence. In general, rule-based systems have three main components to artificially represent the problem solving environment of the production system:

1. Working Memory
2. Rule Knowledge Base
3. Inference Engine

In the rule-based system, the rules represent the long-term memory of the production system. The situations in the short-term memory will be represented by a set of princi-
tive facts which in the context of expert system refer to the simplest independent form of facts. The inference engine is used to mimic the reasoning behaviour of the human expert. It compares the facts from the working memory with the premises of the production rules to see which rule can be fired. Those rules that can be fired will have their conclusions added to the working memory as new facts and this process continues until no other rules have premises that match the facts contained in the working memory.

Most of the commercially available rule-based expert system shells normally have other subsystems which would be more convenient for the programmer to build his or her problem specific expert system. Generally, a more complete expert system shell will have the following additional subsystems:

1. User Interface
2. Interface To External Programs
3. Programmer Interface
Chapter 2. Expert Systems

The architecture of a general rule-based expert system shell is illustrated in Figure 2.2. The following subsections will describe the functions of the individual component of the rule-based system in more detail.

2.3.2 Working Memory

The working memory is the central memory area used to store objects that represent facts about the problem which are generally referred to as primitive facts. Primitive facts can be provided by the user or inferred from other facts by triggering rules from the rule knowledge base. Generally, primitive facts represent the working hypotheses to the problem domain, and a set of primitive facts can uniquely define a state of the problem solving process. Primitive facts are used by the inference engine to trigger rules in the rule knowledge base, and in turns they can be modified by the rules. The absence or presence of certain fact elements in the working memory will activate some rules to which the pattern of their premises are matched. Primitives facts can be added, modified or deleted from the working memory in the light of subsequent information and direct the problem solving process to progress to another state towards the goal.

2.3.3 Inference Engine and Rule Knowledge Base

The inference engine is a knowledge processor which mimics the expert’s reasoning process. Essentially the function of the inference engine is to draw conclusions based on all available information coupled with the expert’s knowledge stored in the form of rules. It searches through the rule knowledge base for a rule to be fired by matching the premises of the rules and the facts in the working memory. Therefore, the rule knowledge base contains rules that govern the inference behaviour of the production system. There are at least two ways of controlling the inference behaviour in a production system:

- Forward Chaining
- Backward Chaining

Without regarding the chaining direction, rules in a rule-based system can be written schematically as:
Chapter 2. Expert Systems

\[ A_1, \ldots, A_m \rightarrow C_1, \ldots, C_p \]

with the meaning of

the consequences of \( C_1, \ldots, C_p \) can be implied if antecedence \( A_1 \) and \( \ldots \) and \( A_m \) are true.

The following subsections will examine the two inferencing mechanisms from the point of view of the inference engine and the rule knowledge base.

2.3.4 Forward Chaining

Forward chaining is a fact-driven search approach in which the reasoning process begins with some known facts in the working memory. It deduces conclusions based on rules whose premises match the known facts and derives new facts to be inserted into the working memory. The process continues until a solution is found or no additional rules can be fired. The syntax of forward chaining rules is generally in the form of:

\[ S_1, \ldots, S_m \rightarrow A_1, \ldots, A_p \]

with the meaning of

if situations \( S_1, \ldots, S_m \) exist then carry out actions \( A_1, \ldots, A_p \)

In a higher level expert system shell, these rules are normally written as

\textbf{Rule the name of the forward chaining rule}

\textbf{IF situations THEN actions}

Situations of forward chaining rules govern the premises for the firing of a particular rule by defining the pattern to be matched against the primitive facts in the working memory. The actions of the rules generally consist of procedures or functions to perform modification, retraction and addition of primitive facts in the working memory. The resultant state of the working memory will be matched against the situations of all the rules in the next cycle of rule firing selection. The behaviour of the inference engine for a forward chaining rule-based system can be described by the recognized-resolution-act cycle which consists of the following steps:

1. Match all the conditions of the rules against facts in the working memory.

2. Pick all the rules that have conditions matched with the facts in the working memory and put it in a conflict set, \( S_{\text{conflict}} \).
Chapter 2. Expert Systems

3. If the set $S_{\text{conflict}}$ contains more than one rule, apply conflict resolution strategy to choose a rule from $S_{\text{conflict}}$ to be fired.

4. Perform the action part of the chosen rule and update the working memory.

5. Repeat Step 1 to 4 until the state of the working memory matches the goal state or there is no rule from the rule knowledge base with conditions that match the state of the working memory.

Figure 2.3 shows a flow chart of the inference process of a forward chaining production system. Some of the typical conflict resolution strategies which can be found in most of the expert systems are:

1. First Come First Serve Strategy: The first rule that matches the state of the working memory will be selected.

2. Highest Priority Strategy: A priority can be attached to each rule and the rule which possesses the highest priority in the conflict set will be selected.

3. Specificity Strategy: A rule is more specific than the others if it has more conditions. Therefore the rule from the conflict set which satisfies the greatest number of conditions will be selected.

4. Recency Strategy: Select the rule with conditions that match the facts which have been most recently added to the working memory. This strategy allows the search to focus on a single line of reasoning.

5. Fire Once Strategy: Every rule in the rule knowledge base can only be fired once.

6. Refraction Strategy: Once a rule has been fired, it will not be fired again until the facts in the working memory that match its conditions have been modified. This strategy is normally used for preventing unwanted looping within the rule knowledge base.

Despite the numerous choices of conflict resolution strategies, most expert systems use the simple strategies in order to increase the speed of the search process.
Chapter 2. Expert Systems

Yes

conflict set empty?

No

apply conflict resolution strategy and choose a rule to be fired

match the conditions of rules with facts from the working memory

place all the rules which have premises match with the state of the working memory in the conflict set

start forward chaining production system

initial known facts from the working memory

add new facts into the working memory

is the goal state reached?

END

Figure 2.3: Forward Chaining Inference Process
The search mechanism of the forward chaining rule-based system starts with a set of known primitive facts. New facts are derived to progress towards the aim of reaching a state where all the primitive facts in the working memory are enough to support the goal state. The search direction of the forward chaining rule-based system which initiates from the bottom of the search space and moves toward the goal at the top, as depicted in Figure 2.4, results in this search approach being known as the bottom-up approach.

2.3.5 Backward Chaining

Backward chaining is a goal-driven search approach in which the reasoning process starts from an expectation of what is to happen (hypothesis), and evidence is sought to support or to contradict the expectation. The backward chaining rules can be written as:

\[ CO_1, ..., CO_m \rightarrow C_1, ..., C_p \]

with the meaning of
if conditions \( CO_1 \) and ... \( CO_m \) are true then conclusions of \( C_1 \) and ... \( C_p \) can be concluded

and their usual form in the higher level expert system is normally formulated as

- **Rule the name of the backward chaining rule**
- IF conditions THEN conclusion

![Search Direction Of The Top-Down Approach](image)

A backward chaining expert system begins with a goal to be proven. The backward chaining inference engine will first check to see if the goal exists in the working memory. If the goal has not been proven, the inference engine will search through all the rules to find one which contains the goal in its conclusion part. This particular rule is generally referred to as the goal rule. The inference engine continues to prove the premises of the goal rule by matching them against the contents of the working memory. The premises which do not exist in the working memory become the new sub-goals to be proven by the inference engine. The sub-goals may in turns be supported by other rules in the rule knowledge base, and the inference engine will continue the goal proving process until it
finds premises which constitute the primitive facts in the working memory. Due to the nature of its goal proving inferencing mechanism, the backward chaining is often called the top-down approach as depicted in Figure 2.5.

2.3.6 Programmer Interface

All expert system development packages offer different facets to the programmer to build their problem specific expert system. A good programmer interface will provide an easier path for the programmer to encode all the problem specific knowledge elicited from the human expert to build rules in the rule knowledge base and characterize the behaviour of the inference engine to mimic the human expert reasoning heuristic. Generally, the programmer is provided with various ways to incorporate and to develop problem domain knowledge into the expert system shell, either by the traditional 'source code entry and compilation' approach or through a smart question and answer editor by an intelligent front end. In general, the latter approach is more user-friendly. However, the former approach is more widely adopted due to its flexibility in using the lower level commands and, consequently, the programmer can gain better insight into the functions and capabilities of the expert system shell.

2.3.7 User Interface

Expert systems provide a symbolic enriched language for problem-oriented communication between the user and the computer. This communication is, in most cases, supplemented by graphical means. The user interface serves the purpose of acting as a communication platform between the user and the expert system shell, where data can be passed back and forth between the user and the expert system. The interaction between the user and the expert system is normally a bi-directional communication throughout the problem solving process. The user may supply useful data which describes the problem to be solved while the expert system continuously updates the user by displaying the intermediary states during the problem solving process.
2.3.8 External Software Interface

Due to the formal syntax of the rules in the rule knowledge base to cater for the flexibility in symbolic manipulations, numerical computations needed for solving numerical-rich and algorithmic problems are commonly not well supported by the rule-based expert system. To cope with such deficiency in the expert systems, most of the commercially available expert system shells offer an open architecture that allows the programmer to interface with external software packages. The programmer is able to write a simple interface program so that data and functions from some external software packages can be called within the rules in the rule knowledge base. This capability adds to the utility of the expert system, so that problem specific information which can be processed and stored naturally in the external programs can be easily accessed and used by the expert system.

2.3.9 Characteristics of Rule-Based Expert Systems

To conclude this section on the rule-based systems, a brief subsection is presented that draws on some characteristics of the general rule-based expert system.

The rules appear to be very similar to the conditional statements in conventional programming languages, such as the logical if-then statement in C-language. Nevertheless, they are used in two totally different programming environments and hence have different characteristics.

Two major differences distinguish rules from conventional programming statements:

1. The condition part of the rule is expressed as a complex pattern to be matched against facts from the working memory rather than just a set of boolean expression to be satisfied.

2. The flow of control does not pass from one rule to the next rule in the lexical sequence but depends separately on the inference engine.

The second distinction is an important feature of the production system. It allows separation of the knowledge from the control of how the knowledge is applied. The work of Newell and Simon [50] on the model of production system suggests that the
expert’s knowledge about a particular problem was separated from the reasoning with
that knowledge. A rule-based system mimics this behaviour of the human expert by
separating the knowledge in the rule knowledge base from the reasoning performed
by the inference engine. This feature is not unique to the production systems but it
is essentially a common characteristic to all expert systems. It allows the knowledge
engineer to change the knowledge of the expert system in the rules knowledge base and
control the inference behaviour separately.
Each rule is an independent chunk of knowledge, the knowledge engineer can review and
modify it easily. Furthermore, since the rules are independent of each other, new rules
can also be added to the existing knowledge base to increase the level of intelligence
of the production system concerning the problem domain. The formal structure of the
rule simplifies the task of consistency checking. All the rules can be checked to ensure
that rules with the same conditions do not pose conflicting actions. Therefore, prior
to adding a new rule to the rule knowledge base, the condition part of the new rule
should be matched with all the existing rules in the rule knowledge base to perform the
consistency checking. However, the more the number of rules being used, the longer
the executing time of the expert system will be and the more laborious the task of the
consistency checking becomes.
The production rule is only one of many choices to represent the knowledge of problem
solving process within an expert system. Despite its numerous advantages, it might not
be the best knowledge representation of the problem domain. Examples of other knowl-
edge include semantic networks, frames, scripts, objects and many more. A structural
knowledge representation is considered in the next section.
2.4 Frame Systems

The production rule is a good representation of linking conditions with actions. However, in some circumstances, the inter-relationship of complex objects in the knowledge domain is crucial in finding the solution to a problem. It is not very convenient to represent knowledge about objects, events and relationships between them in the limited form of rules. The rule formalism is also not very efficient in terms of how the knowledge is stored and accessed. This section explores a more natural representation of structural knowledge known as the Frame System. The use of the frame system to represent knowledge about the structured problem solving domain continues to gain popularity as the scope of the problem solving process gets larger.

Minsky [49] described frames as 'data structures for representing stereotype situations' to which various types of information were being attached. Such information includes the types of objects and events in the situation, as well as embedded procedures which will use the information stored in the frame. Minsky's idea was to use a single knowledge structure to capture all the knowledge that was relevant to a class of objects, rather than distributing the knowledge among smaller structures such as the production rules. He proposed to use frame-based systems to unify both the procedural and declarative expression of knowledge. In a frame-based system, all the knowledge relevant to a concept is stored in a complex entity of structural representation which provides a formalism for grouping explicitly all the knowledge concerning the properties of an individual or a class of objects.

2.4.1 Definition of a Frame

A frame is a generic object-oriented data structure containing any desired number of categories of information called a slot. A frame has a unique name to identify the object it represents. Each frame defines a semi-independent body of knowledge, which can be linked together to form a hierarchical classification of domain knowledge to allow for inheritance. A slot is a component of knowledge which may have various types of entities. These entities can either be a primitive fact, a procedure or another frame depending on which type of slot they are residing in. Generally, there are three basic...
types of slots:

1. Inheritance slots
2. Procedural slots
3. Attribute slots

The inheritance slot is filled by the name of another frame to indicate the inheritance relationship. It is used to form the taxonomy of the frame structure. A set of frames which represent the knowledge in the domain of interest are linked together hierarchically in what is called a frame taxonomy. A frame taxonomy is often depicted graphically as a down opening tree. All the nodes of the tree represent a frame and every branch denotes an is-a link between two frames.

![Diagram of a tree-like taxonomy](image)

An example of a tree-like frame taxonomy is shown in Figure 2.6. The highest node of the structure is the most generic frame, which in this case is the general transfer functions frame. Within the class of transfer functions, subclasses of open-loop transfer functions and closed-loop transfer functions can be defined. Furthermore, the plant transfer function, controller transfer function and open-loop gain transfer function can
be attached to the frame of open-loop transfer functions. By constructing such a hierarchal structure, relevant data and knowledge can be built up neatly to the most specific level required for describing the problem in hand. In addition, the expansion of the frame structure can be done with the addition of levels to every frame in the structure.

The structural taxonomy of frame systems forms the basis of a method of automated reasoning called inheritance. Using the inheritance relationship, frames can be defined as specializations of more general frames by means of the is-a link. Furthermore, frames which are lower in the hierarchy can access the contents of the slots from the frames at the higher level if the slots do not exist in the lower hierarchy frame. The inheritance strategy can be defined as follows:

- if a slot for the data sought exists in the current frame being processed, then the information contained therein is used.
- otherwise, the information in the corresponding slot in the frame at the higher level is accessed.
- if necessary, the inheritance can be pursued up to the highest frame in the same branch of the entire frame hierarchy.
- the information about a particular slot is considered unavailable if the slot cannot be found even in the highest order frame.

The procedural slots define how the information required by the frame should be obtained and what actions should be taken if the designer selects that frame.

The attribute slots are related to the declarative part of the frame knowledge. They normally contain primitives specific to a particular frame which owns the slots. At the bottom of the frame hierarchy, we have primitive frames which only have attribute slots containing primitives but inherit all the procedural and other attribute slots from frames above the hierarchy.

The structure of the frame should be pre-defined to the greatest possible extent. The knowledge engineer attaches various object instances to the object classes of the pre-defined frame structure and fills out the problem- or case-specific knowledge content.
This approach will greatly ease the gathering of knowledge and ensure the consistency of the knowledge base.

2.4.2 Inferencing With Frames

Reasoning with frames is generally more complicated than reasoning with rule-based systems. Frame-based reasoning normally involves confirmation or fulfilment of expectation in certain attribute slot values. All the attribute slots are loosely attached by default values which may be replaced by values, subject to certain conditions, that meet certain assignment rules. The principal idea is that problem-solving computations occur largely as a side effect of the flow of data into and out of the nodes in the frame structure. The simplest method of filling in the values of the slots is by the inheritance relationship defined with the taxonomical structure of the frame.

Demon procedures can be defined and attached to the frame systems for information processing among the slots in the frames. A demon is generally a procedure which will be invoked at a particular time during the manipulation of the frame in which it has been attached. The condition under which a demon is activated depends on the type of the demon. The most popular demons which can be found in the frame systems are:

1. **if-needed demon** which is activated the moment an attribute value is needed but not yet known for the attributes that it is attached to.

2. **if-added demon** which is activated after a value is entered into the value of the attribute slot concerned.

3. **if-removed demon** which is invoked when a value is removed from the attribute slots it is attached to.

To illustrate the usage of the demons, consider a frame knowledge base which is running in real time to control an ongoing process plant. Assume that the plant has several measurements which indicate the current status of the process. Frames can store the values of the process measurements to be used for other manipulations. An if-needed demon can be attached to all these frames to read off the measurements whenever they are required. An if-added demon can also be defined to give further instructions to
manipulate the measurements. Finally, the if-removed demon might contain data-saving instructions before the data is removed and replaced with a new set of data.

This particular method of integrating the procedural and declarative knowledge is known as procedural attachment. The frame formalism with procedural attachment forms a powerful way of knowledge representation. By incorporating demons into the frame system, it is possible to control local state changes due to symbolic computations. The exact state of the frame before a procedure is activated can be defined as the condition for the activation of the procedure.

2.5 Comparison Study Of The Production System and The Frame System

Generally, both of the knowledge representation methods have good features in capturing knowledge from the real world. The use of simple and unstructured rules in production systems offers good modularity in knowledge representation. The uniform syntax of if-then rules is flexible and consistent to be symbolically processed by the inference engine. This modular and uniform representation was at one time held to be a positive advantage. On the other hand, the frame system is capable of capturing stereotyped situations when the syntax of the production system is difficult to cope with. Frames partition knowledge into discrete structures having individual properties, which can then be linked together hierarchically according to the generalization and specification of objects to aggregate an extensive structural knowledge representation. The structural representation of frames enables the exploitation of inheritance mechanisms and subsequently provides a formalism which can be easily visualized and mapped onto real world concepts. The explicit ways of knowledge and data organization ease the extendibility of frames and the maintenance of the knowledge base. In addition, the inheritance property of frames provides a means of knowledge distribution without unnecessary duplication within the entire knowledge base, thus providing an efficient manner of memory usage. With demons and procedures attached to the slots of frames, it becomes a form of dynamic knowledge organization which is able to respond to different situations of the problem solving process. Hence it is an invaluable complement
to any data-driven programming such as the forward chaining production system. To conclude, the frame systems are more expressive than the production systems which can only represent monotonic knowledge.

The use of production systems exploits the separation of knowledge and control, which gives better extensibility, enabling the knowledge engineer to concentrate on the acquisition and organization of knowledge without worrying how to control it in the initial phase of constructing the rule knowledge base. Forward chaining of the production system can be employed to explore new conclusions and the simple backward chaining mechanism can be used to focus on the establishment of facts to derive conclusions. Both chaining mechanisms are simple to implement and understand. In a production system, it is easy to keep track of the rule firing sequence for generating the history of the problem solving process. The rule firing record can be used for understanding the problem solving heuristics and for debugging purposes. This makes the production system excellent in terms of its tractability.

Considering the positive characteristics of the two knowledge representation methods, we conclude that frame systems are invaluable for domain specific data organization and production systems are more suitable for representing the heuristic of performing the problem solving process. Integrating the frame system into the production system can form a powerful tool for the modelling of sophisticated problem solving process. Thus in the application of the integrated expert system for the controller design, we propose to use the frame system to explicitly handle all the related data for the control system design, and rules will be used to capture the heuristics of performing the controller design.

2.6 Summary

In this chapter, various important characteristics of the expert systems were summarized. The concept of knowledge representation was discussed. Two classes of expert systems, i.e. the production system and the frame system were examined, and a comparative study was made. The frame system was proposed for the organization of the data involved in the control system design, and the rule knowledge base can be employed...
to represent the heuristics of performing the control system design.
Chapter 3

ROBUST CONTROL SYSTEM DESIGN

3.1 Introduction To Robust Multivariable Control

Over the past three decades, there has been a number of advances in the development of multivariable linear control theory. This has been mainly motivated by the fact that a growing number of technologically advanced fields such as aerospace and robotics have started to use the multi-input multi-output (MIMO) models with each input possibly having significant effect on all the outputs. These cross coupling effects exposed the deficiencies of classical single-input/single-output control design methods when applied to MIMO systems and led researchers to explore new controller design approaches. For the first two decades, linear optimal and state space methods dominated the literature of multivariable control theory. The major problem of these techniques was their inability to maintain robust stability and performance in the face of model uncertainties. This led researchers to seek systematic design methods that blended classical control insights and techniques with the state space computational algorithms and theorems that had emerged from this early work. During the past decade, a renewed emphasis on design methods based on the frequency domain was seen in parallel with developments in multivariable systems. Developments in frequency domain control design have been greatly induced by the introduction of the $H^\infty$ control theory. The development was to a great extent influenced by the celebrated work of Zames [80]. Since the emergence of $H^\infty$ optimization design framework, many design approaches have been developed based on this optimization method. One of the most widely adopted $H^\infty$ approaches by
practitioners in industry is the Loop Shaping Design Procedure method which is based on the robust stabilization of coprime factor uncertainty and incorporates the generalization from the classical loop shaping principles. This chapter summarizes the main results developed to design robust controllers within the $H^\infty$ optimization framework, ranging from the standard $H^\infty$ optimization method to the more intuitive $H^\infty$ Loop Shaping Design Procedure.

Mathematical preliminaries are listed in Section 3.2 which will be used for the rest of this thesis. Section 3.3 summarizes the different approaches for various representations of model uncertainty and the corresponding robust stabilization problems. We give a brief of designing the robust controller using the standard $H^\infty$ optimization framework in Section 3.4. Two approaches of $H^\infty$ Loop Shaping Design Procedure will be presented in Section 3.5.

3.2 Mathematical Background

3.2.1 Norms for Signals

Let $X$ be a linear space over the field $F$ (typically, $F$ is the field of real numbers $\mathbb{R}$, or complex numbers $\mathbb{C}$). Then a functional $\|\cdot\| : X \rightarrow \mathbb{R}$ that maps $X$ into the real numbers $\mathbb{R}$ is a norm on $X$ iff

1. $\|x\| \geq 0$, $\forall x \in X$ (nonnegativity) \hspace{1cm} (3.1)
2. $\|x\| = 0 \iff x = 0$ (positive-definiteness) \hspace{1cm} (3.2)
3. $\|\lambda x\| = |\lambda| \cdot \|x\|$, $\forall \lambda \in F, \forall x \in X$ (homogeneity with respect to $F$) \hspace{1cm} (3.3)
4. $\|x + y\| \leq \|x\| + \|y\|$, $\forall x, y \in X$ (triangle inequality) \hspace{1cm} (3.4)

Given a linear space $X$ there might be many possible norms on $X$. For a given norm $\|\cdot\|$ on $X$, the pair $(X, \|\cdot\|)$ is called a normed space.

Let the linear space $X$ be continuous or piecewise continuous, scalar-valued time signals $x(t)$, $t \in \mathbb{R}$. The $p$-norm of a signal $x$ is defined by

1-norm. $\|x\|_1 := \int_{-\infty}^{\infty} |x(t)| \, dt$\hspace{1cm} for $p = 1$ (3.5)
Chapter 3. Robust Control System Design

\[ p\text{-norm. } \|x\|_p := \left( \int_{-\infty}^{\infty} |x(t)|^p \, dt \right)^{1/p}, \text{ for } 1 < p < \infty \]  
\[ \infty\text{-norm. } \|x\|_\infty := \operatorname{ess} \sup_{t \in \mathbb{R}} |x(t)|, \text{ for } p = \infty \]  

The average power of \( x(t) \) is the average over time of its instantaneous power, it is defined as

\[ \lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{T} x(t)^2 \, dt \]  

The signal \( x(t) \) will be called a power signal if the above limit exists and is finite. The expression of (3.8) has all the properties of a norm except property (2), therefore it is not a norm. The square root of expression (3.8) is well-known as the r.m.s. (root-mean-square) value of \( x(t) \).

3.2.2 Singular Value

Some of the common norms for a MIMO system can be expressed in terms of the singular values of the \( n \times m \) transfer function \( H \). The singular values of matrix \( H \in \mathbb{C}^{n \times m} \) are defined by

\[ \sigma_i(H) = (\lambda_i(H^*H))^{1/2} \quad i = 1, \ldots, \min(n, m). \]  
where \( \lambda_i(\cdot) \) denotes the ith largest eigenvalue. The largest singular value (i.e. \( \sigma_1 \)) is denoted by \( \sigma \) and the smallest singular value is denoted by \( \sigma_\infty \). The plot of \( \sigma_i(H(j\omega)) \) over the frequency domain is called the singular value plot and is analogous to the Bode magnitude plot of a SISO transfer function.

3.2.3 Norms of MIMO Systems

We consider systems which are linear, time-invariant and finite-dimensional. In the time domain, the relationship between the input and output signals of such a model is given by

\[ y(t) = \int_{-\infty}^{\infty} g(t - \tau) u(\tau) \, d\tau \]  

where \( g(t) \) is the impulse response matrix of the system. Let \( G(s) \) denote the Laplace transform of \( g(t) \), be a proper transfer function matrix with no pole on the imaginary axis, the norms of the transfer function \( G(s) \) are defined as
Chapter 3. Robust Control System Design

2-norm. \[ \| G(s) \|_2 = \left( \frac{1}{2\pi} \int_{-\infty}^{\infty} \text{tr}[G(j\omega)G^T(j\omega)]d\omega \right)^{1/2} \] (3.11)

\( \infty \)-norm. \[ \| G(s) \|_\infty := \sup_{\omega} |\sigma(G(j\omega))| \] (3.12)

It can be shown that \( \| G(s) \|_2 \) and \( \| G(s) \|_\infty \) both satisfy all the properties of norms and furthermore, the \( \infty \)-norm also satisfies for

\[ \| G(s)H(s) \|_\infty \leq \| G(s) \|_\infty \| H(s) \|_\infty \] (3.13)

If \( G(s) \) is a SISO system and it is stable (i.e. it is analytic in the closed right half plane) then by Parseval’s theorem,

\[ \| G(s) \|_2 := \left( \frac{1}{2\pi} \int_{-\infty}^{\infty} |G(j\omega)|^2 d\omega \right)^{1/2} = \left( \int_{-\infty}^{\infty} |G(t)|^2 dt \right)^{1/2} \] (3.14)

3.2.4 Induced Norms

The system represented by the transfer function \( G(s) \) is an operator which maps the function of the input signals into the functions of the output signals, therefore, the norms of \( G(s) \) are often referred to as operator norms. Generally, these norms measure the amplification of this mapping. Consider the input-output mapping of a system which is defined as \( y(s) = G(s)u(s) \), where \( G(s) \) is stable and proper with no poles on the imaginary axis. \( \| G(s) \|_2 \) gives the precise information about the power gain of \( G(s) \) when the input is a white stochastic process. On the other hand, if the input signal \( u(t) \) has finite energy as measured by

\[ \| u(s) \|_2 = \left( \int_{-\infty}^{\infty} |u(t)|^2 dt \right)^{1/2} < \infty \]

then the induced 2-norm of the operator \( G(s) \), \( \| G(s) \|_2 \) is defined as [70]:

\[ \| G(s) \|_2 = \sup_{u \neq 0} \frac{\| y(s) \|_2}{\| u(s) \|_2} = \| G(s) \|_\infty \] (3.15)

From the singular value plot of \( G(s) \), \( \| G(s) \|_\infty \) is simply the peak value of \( \sigma(G(j\omega)) \).
3.3 Robust Stabilization of Uncertain System

The modelling of a physical system for the controller design inevitably involves trade-offs between simplicity of the model and its accuracy in matching the true behaviour of the actual system. In practice, a mathematical model is derived to represent the nominal behaviour of the actual system. Therefore, it is important to design controllers which are able to stabilize the whole family of systems which exist in the uncertain region around the nominal model, i.e. the possible actual systems.

The need to stabilize the uncertain plant motivates the concept of robust stability, that is to design a controller for the nominal plant such that the closed-loop system remains stable in the presence of modelling errors. Robust control system design problems also involve consideration for the performance of the system. In the robust performance problem, the closed-loop system is required to maintain an adequate performance level in the presence of various uncertainties. The robust performance problem is difficult to solve and still remains as an open question for the general case. However, some of the robust performance requirements may be transformed into a robust stabilization problem as will be seen later in this chapter.

During the past decade, the $H^\infty$ optimization approach has provided some promising results with respect to achieving closed-loop robust stability, particularly in the area of robust stabilization of plants with unstructured uncertainty. By unstructured uncertainty, it is meant that the only information available about the uncertainty, $\Delta$, is that it can be bounded by a frequency dependent scalar function $\delta(j\omega)$, i.e.

$$\delta(G(j\omega)) \leq \delta(j\omega) \quad \forall \omega$$

In $H^\infty$ optimization theory, uncertainties which might occur in different parts of the system are lumped into a single perturbation, $\Delta$. The perturbation can be modelled by a stable full transfer function matrix whose dimensions conform to the plant dimensions, in a multiplicative or additive way with respect to the nominal plant. Figure 3.1 to 3.3 illustrate some possible representations of unstructured uncertainties for the purpose of robust stability analysis.
Chapter 3. Robust Control System Design

Figure 3.1: The Additive Uncertainty Model

Figure 3.2: The Input Multiplicative Uncertainty Model

Figure 3.3: The Output Multiplicative Uncertainty Model
Let $G_\Delta(s) \in \Lambda$ be any member of the set $\Lambda$ of possible perturbed plants, and let $G_\circ(s) \in \Lambda$ denotes the nominal model of the physical plant. By using each configuration of unstructured uncertainty model, a class of $G_\Delta(s)$ in which the true plant might lie, can be derived as:

- For additive unstructured uncertainty
  $$G_\Delta = G_\circ(s) + \Delta \quad \sigma(\Delta(j\omega)) \leq \delta(j\omega)$$

- For input multiplicative unstructured uncertainty
  $$G_\Delta = G_\circ(s)(I + \Delta) \quad \sigma(\Delta(j\omega)) \leq \delta(j\omega)$$

- For output multiplicative unstructured uncertainty
  $$G_\Delta = (I + \Delta)G_\circ(s) \quad \sigma(\Delta(j\omega)) \leq \delta(j\omega)$$

Underlying most of the robust stability results is the celebrated Small Gain Theorem. It states that a feedback loop composed of stable operators will remain internally stable if the product of all the operator gain is less than unity. As an example, consider the feedback loop depicted in Figure 3.4.

The closed loop system is formed by two transfer functions $G$ and $K$ both belong to $RH_\infty$ which denotes the space of all real-rational transfer function matrices which have no poles in the closed right half plane. According to the small gain theorem, the closed-loop system is internally stable if

$$\|G(j\omega)\| \cdot \|K(j\omega)\| < 1 \quad \forall \omega$$  \hspace{1cm} (3.16)
The small gain theorem can be applied to yield sufficient condition for any additive and multiplicative representations of uncertainty. For an example, consider a feedback loop containing the plant $G$ and a feedback controller $K$ with a block $\Delta_m$ to represent the stable output multiplicative uncertainty associated with the nominal plant, as illustrated in Figure 3.5. If the uncertainty $\Delta_m$ is bounded in the sense that $\Re(\Delta_m(j\omega)) < \varepsilon \quad \forall \omega$, the small gain theorem states that the closed-loop system will remain internally stable provided that the transfer function from $w$ to $e$ is stable and have gain less than or equal to $1/\varepsilon$, i.e.

$$\Re \left( G(j\omega)K(j\omega)(I - G(j\omega)K(j\omega))^{-1} \right) \leq 1/\varepsilon$$

The small gain condition is in general sufficient but not necessary for stability, unless the unstructured uncertainty may happen in all directions. Therefore the small gain theorem is usually a conservative measure when taken as a guarantee for closed-loop stability.

### 3.4 $H_\infty$ Optimization Control Approaches

Since its emergence, the $H_\infty$ control theory has developed its own terminology, notations and paradigms. The classical block diagram has been modified to handle more general types of control problems. Standard shorthand notation has been introduced to simplify the lengthy design equation. In this section we present a brief introduction of the controller design using the standard $H_\infty$ optimization approach.
3.4.1 General Control Problem Formulation

Feedback control design problems can be cast as an $H^\infty$ optimization problem in numerous ways. Therefore it is very important to have a standard problem formulation in which any particular design problem can be manipulated. Figure 3.6 shows the most widely adopted design formulation for posing performance specifications in the $H^\infty$ optimization framework.

![Diagram](image)

Figure 3.6: The General $H^\infty$ Control Formulation

The diagram consists of two main blocks i.e. the plant $P$ and the controller $K$. The plant has two input vectors and two output vectors. The control input, $u$, is the output of the controller which becomes the input to the actuator for driving the plant. The exogenous input, $w$, is generally a collection of inputs which may include external disturbances, noise from the sensors and actuators as well as the command signals. The plant outputs are categorized into two groups. The first group, $y$, are signals which are measured and fed back forming the input to the controller. The second group, $z$, consists of all the signals we are interested in regulating. They are typically error signals, states and control signals.

The standard plant, $P(s)$ can be partitioned as:

$$P(s) = \frac{P_{11} \quad P_{12}}{P_{11} \quad P_{21}}$$

A transfer function representation of the system is given by

$$z = P_{11}w + P_{12}u$$
Chapter 3. Robust Control System Design

\[ y = P_{21}w + P_{22}u \]  
\[ u = Ky \]  
\[ (3.19) \]  
\[ (3.20) \]

The closed-loop transfer function matrix from the exogenous inputs, \( w \), to regulated outputs, \( z \), is given by the lower linear fractional transformation

\[ z = F_L(P, K)w \]  
\[ (3.21) \]

where

\[ F_L = P_{11} + P_{12}K(I - P_{22}K)^{-1}P_{21} \]  
\[ (3.22) \]

The state space representation of the plant is given by

\[ \dot{x} = Ax + B_1w + B_2u \]  
\[ z = C_1x + D_{11}w + D_{12}u \]  
\[ y = C_2x + D_{21}w + D_{22}u \]  
\[ (3.23) \]  
\[ (3.24) \]  
\[ (3.25) \]

Using the pack matrix notation, the state space representation of \( P \) is written as

\[ P(s) = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \]  
\[ (3.26) \]

The additive and multiplicative uncertainty model introduced in Section 3.3 can be added to the standard \( H^\infty \) control formulation as Figure 3.7.

In Figure 3.8, the perturbed plant model, \( G_\Delta \) can be written in the form of upper linear fractional transformation as:

\[ G_\Delta = F_U(P, \Delta) \]  
\[ (3.27) \]

with the standard plant \( P(s) \) associated with the three uncertainty model descriptions stated earlier as:

- For additive uncertainty

\[ P(s) = \begin{bmatrix} 0 & I \\ I & G_\alpha \end{bmatrix} \]  
\[ (3.28) \]
Chapter 3. Robust Control System Design

Figure 3.7: The General $H^{\infty}$ Control Formulation with Uncertainty

Figure 3.8: The Perturbed Plant Model
For input multiplicative uncertainty

\[ P(s) = \begin{bmatrix} 0 & I \\ G_o & G_o \end{bmatrix} \]  \hspace{1cm} (3.29)

For output multiplicative uncertainty

\[ P(s) = \begin{bmatrix} 0 & G_o \\ I & G_o \end{bmatrix} \]  \hspace{1cm} (3.30)

where \( G_o \) is the nominal model.

### 3.4.2 The Robust Stabilization Problem

In the design of a feedback system, an important objective is to ensure that the closed loop system is robustly stable to the model uncertainty. In the \( H^{\infty} \) framework, the model uncertainty is described as a norm bounded perturbation and the small gain theorem is used to guarantee the robust stability condition. We now state the sufficient condition for robust stability for the additive and multiplicative uncertainty models, which is a generalization of the result by Vidyasagar [70].

**Theorem 3.1** Let \( D \) be the set of all bounded perturbations, \( \Delta \), i.e. \( \| \Delta \|_{\infty} < \varepsilon \), such that the nominal plant \( G \) and the perturbed plant \( G_\Delta \) have the same number of closed right half plane poles.

A controller, \( K \) will stabilize \( F_D(P, \Delta) \) for all \( \Delta \in D \) and any standard plant \( P \) with minimal state space representation if and only if \( K \) stabilizes the nominal plant and \( \| F_D(P, K) \|_{\infty} \leq 1/\varepsilon \).

Nevertheless, in practice, robust stabilization on its own is not very useful as the designer is unable to specify the desired performance requirement. Fortunately, the \( H^{\infty} \) optimization framework is capable of mixing several design constraints into a single description to be optimized, for example, by stacking the robust stability requirement with a frequency domain performance function. This leads to the consideration of performance issues. These are discussed in the following subsection.
3.4.3 The $H^\infty$ Performance Specifications Formulation

With the standard $H^\infty$ control formulation, it is possible to combine several objective functions into the form of a single linear fractional transformation, $\mathcal{F}_L(P, K)$, which can then be minimized over all possible choices of the controller $K$. For completeness, we formally state the standard $H^\infty$ optimization problem as follows:

**Problem 3.1 The $H^\infty$ Optimization Problem**

Find the optimal controller, $K_{opt}$, which solves

$$\inf \| \mathcal{F}_L(P, K_{opt}) \|_\infty := \gamma_{\min}$$

where $K_{opt}$ is chosen over all controllers which internally stabilize the nominal plant.

To cast frequency domain performance specifications into the $H^\infty$-optimization framework, consider the standard feedback configuration illustrated in Figure 3.9.

![Figure 3.9: The Standard Feedback Configuration](image)

The majority of feedback control problems can be configured in this way which includes consideration of the effects of reference input $r(s)$, error signals $e(s)$, control effort $u(s)$, plant output disturbance $d(s)$ and measurement noise $m(s)$.

From the block diagram, the output signal $y(s)$ is derived as

$$y(s) = [I + G(s)K(s)]^{-1} d(s) + [I + G(s)K(s)]^{-1} G(s)K(s) (r(s) - m(s))$$  \hspace{1cm} (3.31)
Chapter 3. Robust Control System Design

It can further be rewritten as

\[ y(s) = S(s)d(s) + T(s)r(s) - T(s)m(s) \]  \hspace{1cm} (3.32)

where \( S(s) \) is called the sensitivity function

\[ S(s) = (I + G(s)K(s))^{-1} \]  \hspace{1cm} (3.33)

and \( T(s) \) is the complementary-sensitivity function

\[ T(s) = (I + G(s)K(s))^{-1}G(s)K(s) \]  \hspace{1cm} (3.34)

Another useful equation for the feedback loop which relates to the control input, \( u(s) \), is

\[ u(s) = K(s)S(s)r(s) - K(s)S(s)m(s) - K(s)S(s)d(s) \]  \hspace{1cm} (3.35)

Equations 3.32 and 3.35 play an important role in the feedback controller design. Particularly, the sensitivity transfer function \( S(s) \) and the complementary sensitivity transfer function \( T(s) \). \( S(s) \) is the gain from the reference signal \( r(s) \) to the tracking error \( e(s) \) or the gain from the disturbance signal \( d(s) \) to the plant output \( y(s) \). Therefore it is sensible to keep \( S(s) \) small for good disturbance rejection as well as small tracking error. On the other hand, \( T(s) \) determines the response of the plant output signal with respect to the reference signal. We will therefore try to keep \( T(s) \) as close to unity as possible so that output signal follows the reference signal well. Nevertheless, \( T(s) \) is also the transfer function from the measurement noise to the output signal as well as the stability robustness indicator for the plant output multiplicative uncertainty as we have seen from Equation 3.16. In practice, the measurement noise and the output multiplicative uncertainty generally exist as high frequency signals, therefore, \( T(s) \) should be kept small at the high frequency range to provide sufficient noise attenuation. The majority of the closed-loop performance objectives are centered around the aim of keeping all the effects of disturbance small and maintaining good tracking properties of reference signals while preserving the robustness of closed-loop stability. In the \( H^\infty \)-optimization framework, this can be achieved by posing an optimization problem with objectives to minimize the energy gain from the disturbance signals to the regulated signals by
suitably defining $w(s)$, $z(s)$ or equivalently $P(s)$. To illustrate this with an example, suppose that we want to achieve good disturbance-rejection performance as well as to maintain stability in the presence of unstructured multiplicative output perturbations, we should keep both the sensitivity $S$ and the closed loop transfer function $T$ small in magnitude. We can emphasize one or other objective at different frequency range by using frequency dependent weights $W_1(s)$ and $W_2(s)$ on $S$ and $T$ respectively. The $H^\infty$ optimization problem thus becomes the minimization of the function described by the lower linear fractional transformation of

$$ F_L(P, K) = \begin{bmatrix} W_1S \\ W_2T \end{bmatrix} $$(3.36)

Defining $G$ and $K$ as in Figure 3.9 we may formulate the standard $H^\infty$ optimization by choosing

$$ P_{11} = \begin{bmatrix} W_1 \\ 0 \end{bmatrix}, P_{12} = \begin{bmatrix} -W_1G \\ W_2G \end{bmatrix}, P_{21} = I, P_{22} = -G $$ (3.37)

Formulæ exist in the literatures [12, 21] for the sub-optimal solution of Problem 3.1, which is to find an internally stabilizing controller which satisfies

$$ \| F_L(P, K) \|_\infty < \gamma $$ (3.38)

where $\gamma$ is a real number greater than the optimal $\gamma_{\text{min}}$. By solving two matrix Riccati equations, a stabilizing controller which has a state dimension equal to or less than the state dimension of the weighted plant can be found. The solution of the optimal $H^\infty$ optimization problem is in general an approximation task, a $\gamma$-iteration is usually performed by starting with a high $\gamma$ and consequently reducing it to yield a $\gamma$ sufficiently close to the optimal $\gamma_{\text{min}}$. On the other hand, the selection of the frequency dependent weights to emphasize different design objectives which are often conflicting with each other requires considerable amounts of expertise and knowledge of the mathematics involved. The major drawback of this mixed optimization approach is the resultant cancellation of all the open-loop stable poles of the plant by the controller. If lightly damped resonant poles of the plant are cancelled by the controller, these modes will be
uncontrollable from the controller input and unobservable from the controller output, therefore they can be excited in response to a disturbance which will affect the control inputs and force the closed-loop system to exhibit undesirable behaviour. Another inflexibility of the $H^\infty$ optimization framework based on the additive or multiplicative description of model uncertainty is that only stable perturbations (either multiplicative or additive) are permitted so that they can be norm bounded. To accommodate these drawbacks, an alternative approach of $H^\infty$ optimization formulation based on the normalized coprime factors of plant uncertainty was developed [22, 46], and this will be discussed next.

3.5 $H^\infty$ Loop Shaping Design Procedure

The previous section motivated the use of $H^\infty$ optimization with additive and multiplicative uncertainty for designing robust stabilizing controllers. During the past decade, many formulations for the synthesis of $H^\infty$ controller have been proposed, the most widely adopted approach for real applications is the two blocks $S/KS$ and $S/T$ $H^\infty$ optimization. These advanced techniques are powerful in their inherent capability to deal with multivariable systems and guarantee a degree of robustness against a specified uncertainty structure. Nevertheless, there has been a very limited acceptance of these methods by practitioners from the industrial sectors. The problem with all these approaches as discussed in the last section is the difficulty in selecting the weights and the undesirable pole and zero cancellations between the plant and the controller. Another potential problem with these approaches is their emphasis on the plant output which may give rise to poor robustness properties at the plant input.

In this section we will examine a specific robust stabilization problem, where the plant perturbation assumed at the normalized left coprime factor of a nominal plant is considered. The controller is designed to allow the $H^\infty$ norm of the coprime factor perturbations to be as large as possible while preserving the internal stability of the closed-loop system. We will see that unlike the standard $H^\infty$ optimization approach, this robust stabilization approach has a particularly straightforward optimal solution, which does not need the $\gamma - interaction$. In addition the selection of weighting functions in the
trade-offs between performance specifications and stability robustness is much simpler yet intuitive than the other $H^\infty$ approaches.

3.5.1 Normalized Coprime Factor Plant Description

Firstly, we define the necessary and sufficient condition for two matrices to be left coprime:

**Definition 3.1** If $\tilde{M}, \tilde{N} \in RH_\infty$ have the same number of rows, then $\tilde{M}$ and $\tilde{N}$ are left coprime if and only if there exist $U, V \in RH_\infty$ such that

$$\tilde{MV} + \tilde{NU} = I$$

(3.39)

It is possible to represent any real-rational, proper transfer function in terms of a pair of asymptotically stable, real-rational, proper left coprime transfer functions which are called the Left Coprime Factorization (LCF) of a transfer function.

**Definition 3.2** The pair $(\tilde{N}, \tilde{M})$, where $\tilde{M}, \tilde{N} \in RH_\infty$, constitutes a Left Coprime Factorization (LCF) of $G$ if and only if

1. $\tilde{M}$ is square and $\det(\tilde{M}) \neq 0$
2. $G = \tilde{M}^{-1}\tilde{N}$
3. $\tilde{N}$ and $\tilde{M}$ are left coprime

Our particular interest is the class of the left coprime factorization of $G$ in which the factors $\tilde{N}$ and $\tilde{M}$ are normalized:

**Definition 3.3** The pair $(\tilde{N}, \tilde{M})$, where $\tilde{M}, \tilde{N} \in RH_\infty$, constitutes a Normalized Left Coprime Factorization (LCF) of $G \in R$ if and only if $(\tilde{N}, \tilde{M})$ is a LCF of $G$, and

$$\tilde{N}\tilde{N}^* + \tilde{M}\tilde{M}^* = I.$$

Let $G$ be the transfer function of the nominal plant model and $G_\Delta$ be the transfer function of the perturbed plant model, then $[\Delta_M, \Delta_N]$ represents the coprime uncertainty of the normalized left coprime factorization of $G = \tilde{M}^{-1}\tilde{N}$, if

$$G_\Delta = \tilde{M}_\Delta^{-1}\tilde{N}_\Delta = (\tilde{M} + \Delta_M)^{-1}(\tilde{N} + \Delta_N)$$

(3.40)
where $\tilde{N}_\Delta = (\tilde{N} + \Delta \tilde{N})$ and $\tilde{M}_\Delta = (\tilde{M} + \Delta \tilde{M})$ represent the left coprime factorization of $G_\Delta$ as illustrated in Figure 3.10.

We will now present the state-space representation of the normalized $LCF$ of $G$. For this purpose, we shall first consider two Riccati equations associated with the minimal state space representation of $G(A, B, C, D)$ which will also be used later for the state space formulation of the stabilizing controller for normalized $LCF$ of $G$.

The Generalized Control Algebraic Riccati Equation (GCARE) is defined by,

$$ (A - B S^{-1} D^* C)^* X + X (A - B S^{-1} D^* C) - X B S^{-1} B^* X + C^* R^{-1} C = 0, \quad (3.41) $$

The Generalized Filtering Algebraic Riccati Equation (GFARE) is defined by,

$$ (A - B S^{-1} D^* C) Z + Z (A - B S^{-1} D^* C)^* - Z C^* R^{-1} C Z + B S^{-1} B^* = 0, \quad (3.42) $$

where in both cases $R = I + DD^*$ and $S = I + D^* D$.

If $(A, B, C, D)$ is controllable and observable, then there exists a unique solution of $X = X^* > 0$ and $Z = Z^* > 0$, to the GCARE and GFARE respectively. If the control gain $F$ and the filter gain $H$ are defined to be

$$ F \triangleq -S^{-1}(D^* C + B^* X) $$

$$ H \triangleq -(B D^* + Z C^*) R^{-1} $$
then the eigenvalue \((A+BF)\) and \((A+HC)\) which correspond to the solution of GCARE and GFARE respectively, will have strictly negative real parts.

Vidyasagar [70] shows that the unique, positive definite solution to the GFARE, \(H\), can be used to form the state-space representation of the normalized LCF of the transfer function \(G = \hat{M}^{-1}\hat{N}\).

Let \((A, B, C, D)\) be a minimal state-space realization associated with the transfer function :

\[
G(s) = C(sI - A)^{-1}B + D = \begin{bmatrix} A & B \\ C & D \end{bmatrix}.
\]

If \([\hat{N}, \hat{M}]\) is the normalized LCF of \(G\), then the state-space realization of \([\hat{N}, \hat{M}]\) is given by:

\[
[\hat{N}, \hat{M}] \triangleq \begin{bmatrix} A + HC & B + HD & H \\ R^{-1/2}C & R^{-1/2}D & R^{-1/2} \end{bmatrix} \tag{3.43}
\]

### 3.5.2 Robust Stabilization of Normalized Left Coprime Factor Plant

To address the robust stabilization of the normalized LCF of \(G\), we define the perturbation class associated with the normalized coprime factor uncertainty as

\[
\mathcal{D}_0 \equiv \{ \Delta = [\Delta_{\hat{N}}, \Delta_{\hat{M}}]; \Delta \in RH_\infty, \|\Delta\|_\infty < \epsilon \} \tag{3.44}
\]

It can be shown that a controller \(K\) will robustly stabilize the perturbed plant in Figure 3.10 if and only if \(K\) stabilizes the nominal plant and

\[
\left\| \begin{bmatrix} K & I \end{bmatrix} (I - GK)^{-1}\hat{M}^{-1} \right\|_\infty < \epsilon^{-1}
\]

where \(\epsilon\) is the perturbation bound for \([\Delta_{\hat{M}}, \Delta_{\hat{N}}]\) as defined in (3.44)

The normalized LCF robust stabilization problem can then be stated as follow:

**Problem 3.2** Find the solution to

\[
\left( \inf K \right) \left\| \begin{bmatrix} K \\ I \end{bmatrix} (I - GK)^{-1}\hat{M}^{-1} \right\|_\infty^{-1} := \epsilon_{\text{max}} := (\gamma_{\text{min}})^{-1} \tag{3.45}
\]
where $\epsilon_{\text{max}}$, a positive real number, is the largest perturbation bound such that $G_\Delta = (\tilde{M} + \Delta \tilde{M})^{-1} (\tilde{N} + \Delta \tilde{N})$ can be stabilized by a single controller $K$ for all $\Delta \in \mathcal{D}_\epsilon$ and $K$ is chosen over all internally stabilizing controllers.

The solution to Problem 3.2 is given by McFarlane and Glover [46] as follows:

**Theorem 3.3** The optimal solutions to the normalized LCF robust stabilization problem is given by

$$
\begin{align*}
\left( \inf_K \left\| \begin{bmatrix} K \\ I \end{bmatrix} (I - GK)^{-1} \tilde{M}^{-1} \right\| \right)^{-1} &= \left( 1 - \left\| [\tilde{N}, \tilde{M}] \right\|_H^2 \right)^{-1/2} \\
\end{align*}
$$

where $\|\cdot\|_H$ denotes the Hankel norm. The maximum stability margin is given by

$$
\epsilon_{\text{max}} = \left( 1 - \left\| [\tilde{N}, \tilde{M}] \right\|_H^2 \right)^{-1/2} > 0
$$

It can be further shown [46] that

$$
\left\| [\tilde{N}, \tilde{M}] \right\|_H^2 = \lambda_{\text{max}} \left( ZX(I + ZX)^{-1} \right)
$$

where $X$ and $Z$ are unique stabilizing solutions of GCARE and GFARE respectively. From Equations (3.48) and (3.45) it is easy to show that

$$
\gamma_{\text{min}} = (1 + \lambda_{\text{max}}(ZX))^{1/2}
$$

Using state space descriptions, the controller which achieves $\gamma_{\text{min}}$ is given in [46] by

$$
K \triangleq \begin{bmatrix} A + BF + \gamma_0(Q^*)^{-1}ZC*(C + DF) & \gamma_0^2(Q^*)^{-1}ZC^* \\ B*X & -D^* \end{bmatrix}
$$

where $Q = (1 - \gamma_0^2)I + XZ$.

### 3.5.3 Loop Shaping Design Procedure

The solution to the robust stabilization problem using the normalized LCF plant description discussed in the previous section can be solved in a very straightforward manner. Therefore, it appears to be a much simpler approach to design controllers which achieve optimal robust stability when compared to the standard $H^\infty$ optimization.
method. However, the trade-off for this simplicity is the lack of scope for the designer to specify any performance related objectives. To circumvent this difficulty, a Loop Shaping Design Procedure (LSDP) is presented in [46] which enables performance requirements to be specified within the normalized LCF framework, so that the designer is able to trade-off between stability robustness and performance objectives. In this approach, the well known singular value loop shaping principles by [13] can be used to achieve performance specifications, while the normalized LCF robust stabilization method is used to guarantee the closed-loop stability.

Over the past decades, the loop-shaping procedure has been successfully generalized to multivariable design problem by using singular values as the appropriate measures of magnitude for matrix-valued transfer functions. By an analogy with classical loop shaping ideas of the SISO system, Doyle and Stein [13] showed that singular values of multivariable systems can be shaped to achieve the benefits from the feedback design in the presence of external disturbances and model uncertainties. Thus closed-loop frequency domain performance specifications can be translated into a set of open-loop singular values constraints to specify a desired shape for the open-loop singular values. Employing the LSDP method, the singular values of the nominal plant are shaped accordingly to give a specified open-loop shape by using a pre-compensator \( W_1 \) and/or a post-compensator \( W_2 \). The shaped plant \( G_s \) is formed by the nominal plant and the shaping functions \( W_1 \) and \( W_2 \), i.e. \( G_s = W_2GW_1 \). The weighting functions \( W_1 \) and \( W_2 \) have to be chosen such that \( G_s \) contains no hidden unstable modes.

\( \gamma_0 \) is then calculated, where

\[
\gamma_0 = \inf_{K \text{ stabilizing}} \left\| \begin{bmatrix} K & 0 \\ I & 0 \end{bmatrix} (I - G_sK)^{-1} \right\|_\infty
\]

(3.51)

and \( \tilde{M}_s \) and \( \tilde{N}_s \) define the normalized coprime factors of \( G_s \) such that \( G_s = \tilde{M}_s^{-1}\tilde{N}_s \), and \( \tilde{M}_s\tilde{M}_s^* + \tilde{N}_s\tilde{N}_s^* = I \). If \( \gamma_0 \gg 1 \), \( W_1 \) and \( W_2 \) have to be re-adjusted. Otherwise, select \( \gamma \geq \gamma_0 \) and synthesize an \( H^\infty \) feedback controller \( K_\infty \) to robustly stabilize the normalized left coprime factorization of the shaped plant, \( G_s \) as shown in Figure 3.11,
Chapter 3. Robust Control System Design

Figure 3.11: Robust Stabilization of The Shaped Plant

Figure 3.12: Construction of The Final Controller
which satisfies:

$$\inf_{K_{\text{stabilizing}}} \left\| \begin{bmatrix} K_{\infty} \\ I \end{bmatrix} (I - G_{1}K_{\infty})^{-1} \tilde{M}_{x}^{-1} \right\|_{\infty} \geq \gamma_{o}$$ (3.52)

The final feedback controller $K_{f}$ is then constructed by combining the $H^{\infty}$ controller $K_{\infty}$ with weighting functions $W_{2}$ and $W_{1}$ such that $K_{f} = W_{1}K_{\infty}W_{2}$, as depicted in Figure 3.12.

The measure of $\gamma_{o}^{-1}$ is the stability margin for the normalized coprime factor robust stability problem and therefore provides a robust stability guarantee for the closed loop system. In addition, $\gamma_{o}$ can be interpreted as an indicator of the success of the loop shaping stage. It is shown [46] that the re-organized controller does not significantly alter the specified loop shape. Specifically, at frequencies where $\sigma(W_{2}GW_{1}) \ll 1$ and $\sigma(W_{2}GW_{1}) > 1$ the deterioration of the loop shape due to $K$ will be minimal if the value of $\gamma_{o}$ achieved and the condition number of the chosen weights are sufficiently small. Therefore $\gamma_{o}$ can be used as an indicator of the effectiveness of the design to recover the final loop shape.

Although in practice, skill is needed to select the weight to shape the open-loop singular values of the plant, experience on real applications has shown that a robust controller can be designed systematically by following some simple guidelines, for example see [34]. Due to its effectiveness and simplicity, many control researchers are favourable to LSDP approach to design robust controllers. Successful applications of the LSDP have been reported [34, 59, 63, 60] since its emergence and are continuing in a growing extent. Advantages of LSDP over some other approaches in $H^{\infty}$ design will now be discussed.

First of all, using the LSDP approach, the loop shaping process is carried out without the need to explicitly consider the closed-loop stability requirements. The robust stabilizing controller can be built without having to specify any further frequency weight once the nominal plant is shaped. The optimal gamma can be found without iteration and the controller has an observer-state feedback structure [65]. Therefore it is obvious that the robust control system design can be performed in a more intuitive manner with relatively less effort. Unlike the other $H^{\infty}$ approaches, the designer can concentrate more on the trade-offs between performance specifications and robustness and the choice of weighting.
functions is much more clearly targeted.

The procedure can be applied to stable or unstable, minimum or non-minimum phase plants, provided there are no hidden unstable modes. Modelling uncertainty in the plant model in terms of perturbations on its normalized coprime factors has advantages over other representations (e.g. additive and multiplicative), because no restrictions on the number of right half-plane poles of the nominal and perturbed plant are imposed. It should be noted that by the definition of coprime factors, \( \hat{M}, \hat{N}, (\hat{M}+\Delta_M) \) and \( (\hat{N}+\Delta_N) \) are all stable, thus \( \Delta_M \) and \( \Delta_N \) will always be stable and hence the Small Gain Theorem can be applied to any perturbed plants. The major improvement of the LSDP approach over the standard \( H^\infty \) optimization method is that the problem of pole-zero cancellations is largely avoided except for a small class of plants which contain an all pass transfer function.

Furthermore, it can be shown that the robust stabilization objectives of Equation 3.2 can be interpreted as a standard \( H^\infty \) problem for the minimization of the \( H^\infty \) norm of the frequency weighted gain from the plant input and output disturbances to the controller input and output as follows \cite{47}:

\[
\begin{align*}
K_\infty & (I - G_s K_\infty)^{-1} \hat{M}_s^{-1} \\
& = \begin{bmatrix}
I \\
K_\infty
\end{bmatrix} (I - G_s K_\infty)^{-1} \begin{bmatrix}
I & G_s
\end{bmatrix} \\
& = \begin{bmatrix}
W_2 \\
W_1^{-1} K_f
\end{bmatrix} (I - G K_f)^{-1} \begin{bmatrix}
W_2^{-1} G W_1
\end{bmatrix} \\
& = \begin{bmatrix}
I \\
G_s
\end{bmatrix} (I - K_\infty G_s)^{-1} \begin{bmatrix}
I & K_\infty
\end{bmatrix} \\
& = \begin{bmatrix}
W_1^{-1} \\
W_2 G
\end{bmatrix} (I - K_f G)^{-1} \begin{bmatrix}
W_1 G W_1^{-1}
\end{bmatrix}
\end{align*}
\]

(3.53)  
(3.54)  
(3.55)  
(3.56)

It can clearly be seen that various important closed-loop transfer functions relating to the input and the output of the shaped plant have been incorporated into the optimization design framework, for example the input and the output sensitivity functions.
(I - KfGf)^{-1} and (I - GfKf)^{-1}, the gain from the input disturbance to the plant output (I - GfKf)^{-1}Gf and the norms of additive, input and output multiplicative plant perturbations Kf(I - GfKf)^{-1}, Kf(I - GfKf)^{-1}Gf and Gf(I - KfGf)^{-1}Kf respectively. Therefore, from equation (3.53 ) and (3.56), it can be seen that solving the normalized LCF robust stabilization problem is equivalent to simultaneously achieving good feedback properties at both points of the shaped plant.

In addition, Freudenberg [20] has shown that the robust performance problem is synonymous to the simultaneous uncertainty problem where the plant output disturbance rejection specification has to be maintained in the presence of the plant input multiplicative uncertainty. It is apparent that the two criteria have been incorporated into Equations (3.53 - 3.56). Therefore the LSDP approach can be regarded as having a certain level of performance robustness.

McFarlane and Glover [47] have further shown that the LSDP is able to ensure that all the closed-loop objectives are norm-bounded and well-behaved by:

\[
\begin{bmatrix} K_{\infty} & I \\ \end{bmatrix} (I - GfK_{\infty})^{-1} \tilde{M}_{f}^{-1} \leq \gamma
\]

then

\[
\bar{\sigma}(Kf(I - GfKf)^{-1}) \leq \gamma \bar{\sigma}(\tilde{M}_{f}) \bar{\sigma}(W_{1}) \bar{\sigma}(W_{2}) \tag{3.57}
\]

\[
\bar{\sigma}((I - GK)^{-1}) \leq \gamma \bar{\sigma}(\tilde{M}_{f}) \kappa(W_{2}) \tag{3.58}
\]

\[
\bar{\sigma}(K(I - GK)^{-1}G) \leq \gamma \bar{\sigma}(\tilde{N}_{f}) \kappa(W_{2}) \tag{3.59}
\]

\[
\bar{\sigma}((I - GK)^{-1}G) \leq \frac{\gamma \bar{\sigma}(\tilde{N}_{f})}{\kappa(W_{1}) \kappa(W_{2})} \tag{3.60}
\]

\[
\bar{\sigma}((I - KG)^{-1}) \leq 1 + \gamma \bar{\sigma}(\tilde{N}_{f}) \kappa(W_{1}) \tag{3.61}
\]

\[
\bar{\sigma}(G(I - KG)^{-1}K) \leq 1 + \gamma \bar{\sigma}(\tilde{M}_{f}) \kappa(W_{2}) \tag{3.62}
\]

where \(\kappa(\sigma)\) denotes the frequency dependent condition number (ie \(\kappa(\sigma) = \bar{\sigma}(\sigma)/\sigma(\sigma)\)).

Equations (3.57),(3.59),(3.62) are robustness measures of the bounds on the additive, input multiplicative and output multiplicative plant perturbations respectively, while Equations (3.58),(3.60),(3.61) give bounds on various performance measures.

Because \([\tilde{N}_{f}, \tilde{M}_{f}]\) is the normalized LCF of \(G_{f}\), \(\bar{\sigma}(\tilde{M}_{f}) \leq 1\) and \(\bar{\sigma}(\tilde{N}_{f}) \leq 1\), it follows that if \(\gamma\) is small and the weights are well-conditioned, then all the above closed-loop
objectives will be bounded in magnitude. Moreover, it can be seen that these objectives can be shaped at low and high frequency regions by using an appropriate choice of weighting functions $W_1$ and $W_2$. Therefore it can be concluded that the LSDP is capable of ensuring good feedback properties both at the input and at the output of the nominal plant.

The major difficulty in using the LSDP method is that users have to convert time domain specifications to open-loop singular values loop shaping constraints at different frequency ranges. Nevertheless, the LSDP approach of robust controller design, offers a number of attractive features mainly due to its simplicity and intuitive generalization from the classical loop shaping method. Therefore, it is served as an initial design method in the expert system and the guidelines for using this method to design robust controller will be studied further in the next chapter.

In the following subsection, a two degree of freedom LSDP design configuration is examined which permits time response specifications to be incorporated into the controller design process.

### 3.5.4 Two Degree of Freedom $H^\infty$ Loop Shaping Approach

The LSDP has been developed further into a two degree of freedom (TDF) scheme [41]. The Two Degree of Freedom LSDP (TDF-LSDP) allows the design of a controller which guarantees a prescribed level of robust stability while forcing the closed-loop transfer function to approximate a given time response model simultaneously. Figure 3.13 shows the structure of the TDF-LSDP formulation.

The controller can be partitioned as $K = [K_1, K_2]$ where the feedforward controller $K_1$ is used as a pre-filter to meet time response specifications and $K_2$ is the feedback controller which is constructed as the single degree of freedom LSDP to meet robust stability and disturbance rejection criteria. The control signal is given by:

$$ u = \begin{bmatrix} K_1 & K_2 \end{bmatrix} \begin{bmatrix} \beta \\ y \end{bmatrix} $$

The purpose of the feedforward controller $K_1$ is to ensure that:

$$ \|R_{u0} - M_0\|_\infty \leq \gamma \rho^{-2} $$
where $R_{\phi\theta} = (I - GK_2)^{-1} GK_1$ is the closed loop transfer function mapping from $\beta$ to $y$. The reference model $M_0$ is used to represent the ideal time-domain response, which the closed-loop system is desired to follow. The controllers are synthesized to robustly stabilize the shaped plant and simultaneously enforce a robust model-matching property between the closed-loop system and the reference model. The scaling factor $\rho$ is used to weight the relative importance of robust stability as compared to robust model-matching.

To use the TDF-LSDP for controller synthesis, the starting point is to formulate the TDF-LSDP configuration as the generalized $H^\infty$ control problem as in Figure 3.6. If the exogenous signals ($w$) are defined as $\{r, \phi\}$, the control input ($u$) to be $\{u\}$, the regulated signals ($z$) to be $\{z, y, u\}$ and the controller input ($y$) to be $\{\beta, y\}$, then the closed loop transfer function matrix from the exogenous signals ($w$) to the regulated signals ($z$) can be written as the following equation:
Chapter 3. Robust Control System Design

\[
\begin{bmatrix}
  z \\
  y \\
  u \\
  \beta
\end{bmatrix} =
\begin{bmatrix}
  \rho^2 (I - G_s K_2)^{-1} G_s K_1 - M_0 & \rho (I - G_s K_2)^{-1} \tilde{M}_s^{-1} \\
  \rho G_s (I - K_2 G_s)^{-1} K_1 & (I - G_s K_2)^{-1} \tilde{M}_s^{-1} \\
  \rho (I - K_2 G_s)^{-1} K_1 & K_2 (I - G_s K_2)^{-1} \tilde{M}_s^{-1}
\end{bmatrix}
\begin{bmatrix}
  r \\
  \phi
\end{bmatrix}
\] (3.63)

Therefore if \( \|\Delta_M \Delta_N\|_\infty < \gamma^{-1} \) then the loop will remain stable for all \( \Delta_M, \Delta_N \in RH_\infty \) provided that

\[
\begin{bmatrix}
  \rho^2 (I - G_s K_2)^{-1} G_s K_1 - M_0 & \rho (I - G_s K_2)^{-1} \tilde{M}_s^{-1} \\
  \rho G_s (I - K_2 G_s)^{-1} K_1 & (I - G_s K_2)^{-1} \tilde{M}_s^{-1} \\
  \rho (I - K_2 G_s)^{-1} K_1 & K_2 (I - G_s K_2)^{-1} \tilde{M}_s^{-1}
\end{bmatrix}
\leq \gamma
\] (3.64)

and \([K_1 K_2]\) stabilizes the nominal plant. The \((1,1)\) partition of Equation 3.64 is associated with model matching. Immediately from Redheffer’s theorem, the closed loop has the guaranteed robust model matching property of \(\| (I - G_s K_2)^{-1} G_s K_1 - M_0 \|_\infty \leq \gamma \rho^{-2}\) for all perturbed plant generated by \(\Delta_M, \Delta_N \in RH_\infty\) such that \(\|\Delta_M, \Delta_N\|_\infty \leq \gamma^{-1}\). The \((2,2)\) partition is associated with robust stabilization. From the Small Gain theorem, the closed loop system will remain stable for all \(\Delta_M, \Delta_N \in RH_\infty\) such that \(\|\Delta_M, \Delta_N\|_\infty \leq \gamma^{-1}\). The overall aim of Equation 3.64 is therefore to provide robust model matching with robust stabilization in the face of the normalized LCF uncertainty \([\Delta_M, \Delta_N]\). The scaling factor \(\rho\) is used to weight the relative importance of robust stability as compared to robust model matching. In addition, the TDF-LSDP problem reduces to the single degree of freedom LSDP if \(\rho\) is set to zero.

From Figure 3.13, the generalized plant for the \(H^\infty\) regulator framework is given by

\[
\begin{bmatrix}
  z \\
  y \\
  u \\
  \beta
\end{bmatrix} =
\begin{bmatrix}
  P_{11} & P_{12} \\
  \beta & P_{22}
\end{bmatrix}
\begin{bmatrix}
  r \\
  \phi
\end{bmatrix}
\] (3.65)
Chapter 3. Robust Control System Design

Let the shaped plant $G_s$ and the reference model $M_o$ have state-space realizations of

$$
G_s = \begin{bmatrix}
A & B \\
C & D
\end{bmatrix}
$$

$$
M_o = \begin{bmatrix}
A_o & B_o \\
C_o & D_o
\end{bmatrix}
$$

A state space realization of the generalized plant $P$ is given by

$$
P = \begin{bmatrix}
\bar{A} & B_1 & B_2 \\
C_1 & D_{11} & D_{12} \\
C_2 & D_{21} & D_{22}
\end{bmatrix}
$$

The optimization problem and thus the synthesis of the controller can then be solved by a matrix environment control system design software package, e.g. Robust Control Toolbox of MATLAB [8].

For plants with additional feedback variables in which the number of measurements exceeds the number of outputs to be controlled, Walker [71] suggested an alternative TDF-LSDP for the strictly proper shaped plants by introducing a 'flat' output selection matrix $F$, which causes only the controlled outputs to be selected in the model matching.
part of the optimization. The formulation of the generalized plant $P$ becomes

$$ P = \begin{bmatrix} \hat{A} & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} = \begin{bmatrix} A & 0 & 0 & (BD' + ZC') & B \\ 0 & A_0 & -B_0 & 0 & 0 \\ \rho F_1 C & -\rho^2 C_0 & -\rho^2 D_0 & \rho F_0 & \rho D \\ C & 0 & 0 & I & D \\ 0 & 0 & 0 & 0 & I \\ 0 & 0 & \rho I & 0 & 0 \\ C & 0 & 0 & I & D \end{bmatrix} \tag{3.67} $$

Robust controller can be designed using the TDF-LSDP approach with some simple guidelines presented by Limebeer et al. [41]:

1. Shape the singular values of the open-loop plant with pre and post compensators as in the LSDP to meet the desired closed-loop performance specifications.

2. Choose a simple reference model $M_o$ to be followed by the closed-loop system. The chosen reference model must be realistic to avoid the resultant poor robust stability of the closed-loop system and/or excessive control effort.

3. Find the minimal value $\gamma_{\text{min}}$ for the robust stabilization problem using Equation (3.49).

4. Select a value of $\rho$ to reflect the weight between the properties of robust stabilization and robust model matching.

5. Find the optimal value of $\gamma$ through $\gamma$-iterations and construct the optimal controller, by solving the standard $H_{\infty}$ optimization problem with $P$ defined in (3.67).

6. Pre and post multiply the optimal controller with $W_2$ and $W_1$ respectively to give the final feedback controller.

7. Rescale the feedforward controller, $K_1$, by replacing it with $\rho^{-2}K_1$ to achieve perfect steady model matching.
This procedure will be used as an enhancement of the single degree of freedom LSDP in the Expert System; therefore the guidelines and heuristic of designing the robust control system using this approach will be discussed in the next chapter.

3.6 Summary

In this chapter, a comprehensive treatment of robust controller design via the $H^\infty$ optimization methods was given. Various representations of model uncertainty and the corresponding robust stabilization problem were presented. Robust controller design using the standard $H^\infty$ optimization framework was introduced. The concept of normalized coprime factor perturbation was introduced. An $H^\infty$ loop shaping design procedure was presented and its potential advantages over the standard $H^\infty$ optimization approach was discussed. Finally, a two degree of freedom extension of the $H^\infty$ loop shaping design procedure was presented.
Chapter 4

DEVELOPMENT OF AN EXPERT SYSTEM FOR ROBUST CONTROLLER DESIGN

4.1 Motivation

With the increasing complexities of modern plants and the emergence of various advanced design methodologies, the tasks of performing control systems analysis and controller design require judicious decision, extensive experience and heuristics. In the absence of a unified approach to the design formulation of a wide spectrum of systems and uncertainty configurations, considerable expertise becomes necessary to suitably apply the recent theoretical developments of controller design to solve real industrial problems. Although the LSDP design approaches discussed in Chapter 3 offer systematic and powerful robust controller design methods, they only form a small step in the overall robust controller design process. Many important issues related to the heuristics of using the design techniques need to be addressed in order to apply it successfully. In particular, practising control engineers from the industry often have difficulty in following the theoretical development of the design method mainly due to the complexity of mathematics involved, short product development cycles and the nature of their working environment. It is in these situations that the expert system will be invaluable to provide assistance to the industrial engineers.

An expert system approach can combine the heuristic of controller design with available
control systems analysis and design packages to form an integrated environment thereby making modern control techniques accessible to users from the industrial sector. Broadly speaking, the expert system can assist users through an intelligent user-interface by coordinating loop shaping design procedures to find a robust controller which achieves some prescribed level of performance. It helps the designer to formulate a set of comprehensive design specifications and to deal with multiple design constraints placed on the controller design. In doing so, the expert system can effectively extend the ease of application of the LSDP approach.

The LSDP approach requires the design specifications to be specified in terms of frequency weighted magnitude constraints on the open-loop singular values plots of the plant. The design cycle comprises setting a set of open-loop performance bounds to shape the singular values of the plant before the $H^\infty$ optimization technique is employed to synthesize a controller which robustly stabilizes the shaped plant. Furthermore, if the designer has only a set of closed-loop time domain specifications, then the open-loop frequency domain requirements have to be approximated based on the closed-loop time domain specifications before the LSDP technique can be used. In general, it is obvious that this design approach is inherently iterative, whereby the open-loop specifications are engineered in stages based on the closed-loop time domain requirements, with specifications being added or removed, tightened or relaxed at each step, until a satisfactory closed-loop performance is achieved.

The main function of the expert system is to provide a user-friendly and intuitive environment for the user to formulate and refine various design specifications and provides essential assistance in the synthesis of a controller which meets the design specifications.

4.2 Data Organization For Control System Design

In building an expert system for robust controller design, before deciding on the heuristic of the control design methodology, it is important to study the types of control design related data to be manipulated, and to consider how to organize them effectively within a framework so that the data can be processed efficiently by the expert system. In general, the control system design can be qualified as an attempt to satisfy a set of design
objectives. Typically, some of the objectives define levels of desirable performance, e.g. the rise time of the closed-loop system, whereas others specify limitations which must be enforced by all designs, e.g. the limits on the control signal. Generically, this set of objectives is referred to as the design specification which includes all the desirable as well as the essential characteristics to be achieved.

In this thesis, the problem of linear controller design is considered and the specifications here fall into two categories, i.e. the time domain and the frequency domain. Time domain specifications describe the goals of reference signals tracking, desired characteristics of disturbance rejection and the limitations of control efforts. Specifications in frequency domain describe all the characteristics and the shapes of open-loop and closed-loop singular values. Each specification can be further decomposed into a set of attributes, each of them defines a specific requirement and characteristic of the specification.

After careful consideration of the design of linear controllers, a list of essential constraints and attributes has been identified to be used by the design specification. Table 4.1 lists all the constraints and attributes, respectively, which will be used in the linear controller design. With the definitions of the specifications and their attributes, all the design requirements in the time domain or the frequency domain can be precisely characterized.

Consider the frequency plot in Figure 4.1, where three constraints on the open-loop singular values are shown. Let the first constraint be the crossover frequency constraint which is required to be at the point $C$. Using the definition of specification attributes, the crossover frequency constraint can be represented by the combination of the following attributes:

req\_value : $C$
type : equal\_to
extent : point
tolerance : $\text{To}_1$

Note that the variable $\text{To}_1$ can be any real value which is determined by the designer. This attribute is particularly useful when the designer is not sure about the exact con-
### Table 4.1: Constraint And Attributes For Linear Controller Design

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Domain</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>req.value</td>
<td>Time/Frequency</td>
<td>The required value of the specification</td>
</tr>
<tr>
<td>actual.value</td>
<td>Time/Frequency</td>
<td>The actual value of the constrained item</td>
</tr>
<tr>
<td>error</td>
<td>Time/Frequency</td>
<td>req.value - actual.value</td>
</tr>
<tr>
<td>type</td>
<td>Time/Frequency</td>
<td>The bound characteristic of the specification : min/max/equal_to</td>
</tr>
<tr>
<td>extent</td>
<td>Frequency</td>
<td>The extension of the specification over the entire frequency spectrum</td>
</tr>
<tr>
<td></td>
<td></td>
<td>point/left_region/right_region</td>
</tr>
<tr>
<td>at_frequency</td>
<td>Frequency</td>
<td>The effective frequency point</td>
</tr>
<tr>
<td>tolerance</td>
<td>Time/Frequency</td>
<td>Fractional permitted error</td>
</tr>
<tr>
<td>in_tol</td>
<td>Frequency</td>
<td>The status of the specification whether it is satisfied within the tolerance</td>
</tr>
<tr>
<td>fuzzy_error</td>
<td>Time/Frequency</td>
<td>The normalized error ($\frac{\text{error}}{\text{req.value}}$)</td>
</tr>
<tr>
<td>priority</td>
<td>Time/Frequency</td>
<td>The integer type ranking of importance for each constraint</td>
</tr>
</tbody>
</table>
Chapter 4. Development of An Expert System For Robust Controller Design

Figure 4.1: The Open-loop Frequency Constraints

For instance, if the crossover frequency C must be set within the range of 10 rad/s < C < 20 rad/s, then we can specify the req.value to be 15 rad/s and the tolerance to be 5 rad/s.

For the low frequency constraint, we require the open-loop singular values to be larger than L(w), for the frequency range of 10^{-\infty} < w < W_L in the logarithmic scale. It can be characterized by the following attributes:

req.value : L(w)
type : min
at.frequency : W_L
extent : left.region

Specifying min for the type of the constraint implies that the singular values must be above the req.value. The combination of at.frequency and extent stretches the constraint to the left of the logarithmic scale (towards 10^{-\infty}) from frequency W_L.

Similarly, for the high frequency constraint, we require the open-loop singular values to be smaller than H(w) for the frequency range of W_H > w > 10^{\infty} in the logarithmic scale. It can be characterized by the following attributes:

req.value : H(w)
type : max
at.frequency : W_H
On the other hand, time domain specifications can also be characterized in a similar manner to the frequency domain specifications. To illustrate the characterization of the time domain specifications, let us consider the unit step response which is bounded by the thicker lines, shown in Figure 4.2. Three constraints are sufficient to characterize the bounded area, i.e. the overshoot, the rise time and the settling time.

![Figure 4.2: A Time Domain Step Response](image)

The overshoot constraint can be represented by the following attributes:
- `req.value`: OS
- `type`: max

The rise time constraint can be represented by the following attributes:
- `req.value`: RT
- `type`: max

Finally the settling time can be represented by the following attributes:
- `req.value`: ST
- `type`: max

The attribute of `actual.value` is used for recording the current value of the item (e.g. singular values or step responses) bounded by the constraint. The attributes of `error` and `fuzzy.error` will be useful to give a picture of how far the `actual.value` is away from the `req.value`. With the `priority` attribute, the designer can weigh the relative...
importance of different specifications. For instance, the control signal limitation may always weigh higher than any performance related specifications.

After reaching a systematic formalism for representing design data, attention will be switched to design heuristics in the next section, which can be employed in parallel to our data organisation to form a systematic design procedure.

4.3 Design Heuristic Of Singular Values Loop Shaping

By using the LSDP approach, the $H^\infty$ stabilising controller can be synthesized directly based on the shaped plant. Therefore, the heuristics of the LSDP approach boil down to the rules of thumb of shaping the open-loop singular values of the plant to reflect the importance of various closed-loop design objectives. The main difficulty in arriving at an appropriate shape of the open-loop singular values is how to relate the design objectives, which are often given as constraints in some closed-loop transfer functions, with the desirable shape of the open-loop singular values. In this section, the fundamental relation between the closed-loop and open-loop singular values in the MIMO system will be examined to devise a generic way of shaping the open-loop singular values. A systematic design procedure of the LSDP approach can then be formed by joining the heuristic of shaping the plant with the convenience of synthesizing the $H^\infty$ controller from the LSDP approach.

4.3.1 Shape of the Closed-Loop Singular Values

Major developments in the mathematical theory of multivariable linear time invariant feedback systems over the past decades have shown how classical SISO traditions of the feedback design methodologies can be generalized to the MIMO systems. The stability and performance criteria can be derived in a similar manner like their SISO counterparts. Singular value functions used in the statements of MIMO design play a similar role as the Bode magnitude function in classical loop-shaping design. In Chapter 3, it was seen that many performance and robust stability objectives can be formulated as requirements on the maximum singular values of the particular closed-loop transfer functions. The principal idea of loop shaping is that the singular values of closed-loop transfer functions
over the appropriate frequency range can be conveniently determined by the open-loop
singular values, \( G(s)K(s) \). To illustrate the relationship between common closed-loop
design objectives and open-loop singular value requirements, we consider some common
closed-loop design objectives and show how each objective can be approximated, over
a specific frequency range, by the open-loop singular values.

From Chapter 3, for the standard feedback configuration shown in 4.3, the following
fundamental closed-loop relation can be derived:

\[
y(s) = S(s)d(s) + T(s)r(s) - T(s)m(s) \tag{4.1}
\]

\[
S(s) = (I + G(s)K(s))^{-1} \tag{4.2}
\]

\[
T(s) = (I + G(s)K(s))^{-1} G(s)K(s) \tag{4.3}
\]

\[
u(s) = S(s)K(s)r(s) - S(s)K(s)m(s) - S(s)K(s)d(s) \tag{4.4}
\]

Figure 4.3: The Standard Feedback Configuration

\( S(s) \) is the sensitivity function and \( T(s) \) is the complementary sensitivity function (also
known as the closed-loop transfer function). The term complementary sensitivity for
Chapter 4. Development of An Expert System For Robust Controller Design

T(s) follows from the identity:

\[ S(s) + T(s) = I \]  \hspace{1cm} (4.5)

Essentially, \( S(s) \) and \( T(s) \) are the transfer functions from the plant output disturbance \( d(s) \) and reference signal \( r(s) \) respectively, to the plant output \( y(s) \).

For the SISO feedback system, closed-loop performance specifications can be easily converted into requirements on the gain of the open-loop Bode diagram. In the multivariable feedback system, closed-loop requirements can be represented by the corresponding open-loop singular values in a similar manner to the SISO counterpart.

To summarize the closed-loop requirements, we first list down some common important closed-loop objectives which illustrate the trade-offs in the feedback system:

1. To assess the disturbance rejection properties of the closed-loop system, we are interested in keeping the sensitivity function, \( S(s) \), as small as possible.

2. To avoid the propagation of measurement noise, we should keep the complementary sensitivity function, \( T(s) \), small.

3. To track reference signal properly, \( T(s) \) is required to be \( I \).

4. To minimise the control signal, \( K(s) \) should be kept as small as possible, especially at the high frequencies to avoid unwanted peak at transient period.

5. The bandwidth of the closed-loop transfer function should not be too large to avoid amplification of unwanted high frequency noise.

Due to the identity of Equation (4.5), it can be seen immediately that requirements (1), (2) and (3) conflict with each other, illustrating the classical trade-offs between maintaining good tracking performance and having effective disturbance rejection. Fortunately, the conflicting design requirements generally occur across different frequency ranges. It will be shown later that these closed-loop requirements can be met, by appropriate adjustment of the singular values of the open-loop gain.
4.3.2 Relationships Between Open-Loop And Closed-Loop Design Objectives

To illustrate the relationship between common closed-loop design requirements and open-loop singular value constraints, consider the first three requirements on the closed-loop system mentioned above, i.e. constraints on sensitivity function and complementary sensitivity function. It will be shown that each of them can be approximated by the open-loop singular values over a specific frequency range.

First consider the minimization of the sensitivity function, \( S(s) \). To reduce the effect of output disturbances on the plant output at the low frequency region, \( \sigma[(I + GK)^{-1}] \) is required to be small:

\[
\sigma[(I + GK)^{-1}] = \frac{1}{\sigma[I^GK]}
\]

\[
\leq \frac{1}{\sigma[GK] + 1}
\]

\[
\approx \frac{1}{\sigma[GK]} \text{ at frequencies when } \sigma[GK] \gg 1 \quad (4.6)
\]

Therefore, by keeping the minimum singular value of the open-loop gain, \( G(s)K(s) \), high enough at low frequency region, the feedback system can effectively reject the disturbances that appear at the plant output.

To minimize the effect of measurement noise on the plant output which normally appear as high frequency signals, and also to maximize the robust stability due to the output multiplicative plant uncertainty, it is necessary to minimize the singular values of the complementary sensitivity. Suitable manipulation of the transfer function leads to:

\[
\sigma[(I + GK)^{-1}GK] \leq \frac{1}{1 + \sigma[(GK)^{-1}]}
\]

\[
\approx \sigma[(GK)] \text{ at frequencies when } \sigma[(GK)] \ll 1 \quad (4.7)
\]

Therefore, to reject measurement noise, the singular values of the loop-gain should possess a high roll off rate at high frequency region and maintain low magnitude thereafter to maximize robustness of the feedback system in the face of plant uncertainty.

To track the reference signal properly, the complementary sensitivity function is required to be near unity and according to Identity (4.5), the sensitivity function should be very
small. By the definition of induced norm in Section (3.2.4), it is required that
\[ \frac{\| S(s)r(s) \|}{\| r(s) \|} \ll 1 \text{ for all } r(s) \]
or equivalently,
\[ \sigma[S(s)] = \sigma[(I + GK)^{-1}] \ll 1 \]
\[ \Leftrightarrow \sigma[(I + GK)] \gg 1 \]
\[ \approx \sigma[(GK)] \gg 1 \] (4.8)

From 4.6, \( \sigma[(GK)] \gg 1 \) typically happens at the low frequency region, coinciding with the closed-loop performance requirement which is typically crucial at lower frequency range. Summarizing (4.6)-(4.8), it is obvious that although conflicting trade-offs exist between robust stability (i.e. require \( \sigma[(GK)] \) to be small) and performance objectives (i.e. require \( \sigma[(GK)] \) to be large), an acceptable compromise can be achieved at different frequency regions. As performance is typically more important at relatively low frequencies, it is desirable that \( \sigma[(GK)] \ll 1 \) in the low frequency range. To reduce the effects of high frequency noise and disturbances and to accommodate for robust stability requirements, it is required that \( \sigma[(GK)] \ll 1 \) at high frequency region.

Figure 4.4 illustrates graphically how the closed-loop requirements constrain the shape of the open-loop singular values in the feedback system design.

The open-loop singular values must be adjusted to avoid the design boundaries of \( L(w) \) (in the low frequency region) and \( H(w) \) (in the high frequency region). In other words,
\[ \sigma[(GK)] > L(w) \text{ for } 10^{-\infty} < w < W_L \]
and
\[ \bar{\sigma}[(GK)] < H(w) \text{ for } W_H < w < 10^{\infty} \]

The crossover frequency of the loop gain should not be too large to avoid the unwanted excitation of the unmodelled high frequency plant dynamics as well as the amplification of high frequency measurement noise. Therefore the frequency range over which performance objectives (bounded by \( L(w) \)) can be met is directly constrained by various
uncertainties. It is also obvious that $L(w)$ is severely constrained by the slope of open-loop singular values near the crossover frequency. The steeper these singular values roll off, the wider the frequency range over which $L(w)$ can be extended towards the crossover frequency. However, the MIMO linear time invariant transfer functions usually behave in such a manner that increasing the roll-off rate at the crossover frequency results in a large resonance peak in the sensitivity and complementary sensitivity functions, as a result of the so called 'water bed effect' [38]. This condition will be even more severe if the plant is non-minimum phase and unstable. Therefore, unwanted characteristics of the plant can contribute to major limitations on the achievable performance of the feedback system.

4.4 A Systematic Controller Design Using The One Degree Of Freedom Loop Shaping Approach

This section considers the Loop Shaping Design Procedure (LSDP) proposed by McFarlane and Glover [46] which was introduced in Section (3.5). Incorporating the simple performance and robustness trade-offs outlined in the previous section to perform loop shaping of the open-loop singular values into the LSDP approach, results in a systematic
approach of robust controller design for a nominal plant based on the principle of classical loop-shaping methodology. The systematic approach extends the ease of performing the analysis of the design process, and the design can be assessed by the bounds on the open-loop singular values plot.

The design objectives are achieved by the manipulation of the open-loop singular values which in turn are approximated from the closed-loop transfer function requirements. The well-defined and systematic nature of the approach will also facilitate the development and organization of the rule knowledge base which captures the general controller design heuristic.

In conjunction with the LSDP approach, the systematic design of feedback controllers can be divided into four distinct phases as follows:

1. Loop Shaping Of Open-loop Singular Values
2. Robust Stabilization
3. Construction Of The Final Controller
4. Formulation Of the Closed-loop Feedback System

The following three sub-sections will be presented with details of these design phases.

### 4.4.1 Loop Shaping of Open-Loop Singular Values

The loop shaping problem of the open-loop singular values can be partitioned into four sub-tasks, each of them serves to satisfy a certain level of closed-loop performance and robustness requirements:

1. Low Frequency Constraint
2. High Frequency Constraint
3. Crossover Frequency Constraint
4. Slope Constraint Of The Crossover Frequency
4.4.2 Robust Stabilization

Robust stabilization of the shaped plant is carried out using the formulae from chapter 3. At this point, the maximum stability, \( \gamma_{\text{min}} = \frac{1}{j_{\text{max}}} \) is calculated using formula (3.47) to check the compatibility of the loop shape. If \( j_{\text{max}} \) is found to be too big (> 5), the weights should be modified until a satisfying \( \gamma_{\text{min}} \) is achieved. Select \( \gamma > \gamma_{\text{min}} \) by about 5% to 10% and synthesize a suboptimal controller using formula (3.50).

4.4.3 Construction of the Controller and the Closed-Loop System

When implementing the controller, the configuration shown in Figure 4.5 has been found to be more useful than the conventional setup shown in Figure 3.12 [34]. The advantage of this configuration is that the reference \( r(s) \) does not directly excite the dynamics of \( K_{\infty} \), which can result in large overshoot if there is any derivative term in \( K_{\infty} \). The constant pre-filter which consists of the DC gain of \( K_{\infty} \) and \( W_2 \) in series, ensures a steady state gain of 1 between \( r(s) \) and \( y(s) \), assuming integral action exists in \( W_1 \). Extra dynamic weights can be inserted into the post-compensator, \( W_2 \), if needed by the designer.

\[
\begin{array}{c}
\text{\( r(s) \)} \quad K_{\infty}(0)W_2(0) \quad \text{+} \quad W_1 \quad u(s) \quad G \quad y(s) \\
\text{\( K_{\infty} \)} \quad \text{\( W_2 \)}
\end{array}
\]

Figure 4.5: Construction Of The Final Controller And The Closed-Loop System

4.5 A Systematic Controller Design Using Two Degree Of Freedom Loop Shaping Approach

Having obtained the shaped plant, the two degree of freedom scheme of the LSDP can be readily employed to fine tune the time response tracking requirements of the
Chapter 4. Development of An Expert System For Robust Controller Design

closed-loop system. A reference model can be generated based on specifications of the reference tracking between $r(s)$ and $y(s)$ in Figure 3.13. γ-iteration is then performed to find the optimal $H^{\infty}$ controller based on the LSDP two degree of freedom design framework. The reference model can be modified to improve the time domain tracking specifications.

Figure 4.6: Block Diagram Of The Reference Model Based On A First Order Transfer Function

Using the specifications of tracking the reference signal, a reference model based on a first order transfer function can be derived analytically. Consider the first order transfer function shown in Figure 4.6, the time domain description of $Y(s)$ based on a step function input at $R(s)$ can be written as the inverse Laplace transform of $Y(s)$ as

$$y(t) = \mathcal{L}^{-1}\{Y(s)\}$$

$$= \mathcal{L}^{-1}\{\frac{A}{s + A}R(s)\} \quad \text{where } R(s) = \frac{1}{s} \text{ for unit step input}$$

$$= \mathcal{L}^{-1}\{\frac{1}{s} - \frac{1}{s + A}\}$$

$$= 1 - e^{-At}$$

(4.9)

Where $A$ is the time constant of the first order transfer function.

An approximation of the rise time is the time taken for $y(t)$ to reach a magnitude of 0.9. If, for example, the reference model is required to have a rise time of 3 seconds, by direct substitution we will have the following equation:

$$0.9 = 1 - e^{-3A}$$

(4.10)

From Equation 4.10, it is obvious that the time constant $A$ can be calculated directly.

A MATLAB m file was written to determine a suitable reference model based on the
closed-loop tracking specifications on the rise time. With a given open-loop shape of the singular values, the performance of the closed-loop system can be improved by reducing the rise time of the reference model. However, past experience indicated that this might result in an unavoidable increment of the magnitude of the control signals and their rates. Therefore, the constraints on the control signals and their rates need to be checked whenever the rise time of the reference model is reduced.

4.6 Rule Knowledge Bases For The One Degree Of Freedom LSDP Design

In parallel to the systematic approach discussed in the previous section, rules are grouped into different rule knowledge bases. Each rule knowledge base aims to satisfy a set of design constraints and each rule within addresses a particular design constraint. Initially, three basic rule knowledge base modules have been identified:

1. the open-loop frequency domain design module
2. the closed-loop frequency domain design module
3. the closed-loop time domain design module

The function of each rule knowledge base is examined in the following subsections.

4.6.1 Open Loop Frequency Domain Design Rule Knowledge Base

The main function of the open-loop frequency domain design rule knowledge base is to assist the user in the task of shaping the open-loop singular values. Therefore, it is essentially a collection of basic loop shaping heuristics incorporated into the syntax of rules. The most important rules in this module, with their functions, are listed in Table 4.2.

The condition number of the open-loop gain at the desired crossover frequency point will be checked by each rule of the rule knowledge base. A procedure named set_the_crossover_frequency is used by each rule to set the loop gain to the specified crossover frequency if the condition number of the plant is low at the desired crossover frequency. The procedure
### Table 4.2: Rules From The Open-Loop Frequency Domain Design Module And Their Functions

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Function Of The Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>integrators</td>
<td>Add an integrator to the weight if integrating action is needed and the plant's DC gain is not high enough.</td>
</tr>
<tr>
<td>compensate_low_freq_cons</td>
<td>To add a band-pass filter in the weight if a low frequency gain constraint is not satisfied.</td>
</tr>
<tr>
<td>compensate_high_freq_cons</td>
<td>To add an extra pole in the weight if a high frequency gain constraint is not met.</td>
</tr>
<tr>
<td>compensate_for_slope</td>
<td>To add a zero to the weight if the slope near the crossover frequency exceeded the specified requirement.</td>
</tr>
</tbody>
</table>

uses the `align` function from the MATLAB multivariable toolbox [19] which implements the result given by Hung et al.[33] based on the work of Edmunds et al.[14]. Otherwise, for plants with a high condition number at the crossover frequency, the `align` function might fail to align the plant at the crossover frequency properly, a constant gain will be added instead to the weight for shifting the loop nearer to the crossover frequency.

```plaintext
rule integrators
if open_loop_integrating_action_is_required
   and open_loop_integrating_action_is_not_present
then add_integrator_to_open_loop_gain
   and set_the_crossover_frequency_to_the_desired_value.
```

**Figure 4.7: Rule For Inserting An Integrator**

Figure 4.7 is an example of a rule in the open-loop frequency domain design module for adding an integrator to the open-loop gain. The condition part of the rule is checked by
Chapter 4. Development of An Expert System For Robust Controller Design

the inference engine by backward chaining to see if the integrating action is present in the open-loop gain and hence, whether it is required by the open-loop specification. If all the conditions have been verified to be true, the action part of the rule will be fired, i.e. for this rule, an integrator will be inserted to the open-loop gain and the crossover frequency will be set to the desired value.

4.6.2 Closed-Loop Frequency Domain Design Rule Knowledge Base

The main function of the closed-loop frequency domain design module is to assist the user in shaping the closed-loop singular values in the frequency domain. The type of constraints treated by the rules are tabulated in Table 4.3.

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Function Of The Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>comp.s_peak_by_crossover</td>
<td>Reduce the peak magnitude of the sensitivity function by reducing the crossover frequency</td>
</tr>
<tr>
<td>comp.s_peak_by_slope</td>
<td>Reduce the peak magnitude of the sensitivity function by reducing the slope at the crossover frequency</td>
</tr>
<tr>
<td>comp.t_peak_by_crossover</td>
<td>Reduce the peak magnitude of the complementary sensitivity function by reducing the crossover frequency</td>
</tr>
<tr>
<td>t_bandwidth</td>
<td>Increase/decrease the bandwidth of the complementary sensitivity function by increase/decrease the open-loop crossover frequency</td>
</tr>
<tr>
<td>s_mag_constraint</td>
<td>Compensating the magnitude constraint at the low frequency region by modifying the low frequency constraint of the open-loop singular values</td>
</tr>
</tbody>
</table>

Table 4.3: Rules From The Closed-Loop Frequency Domain Design Module And Their Functions

Based on the relationship between open-loop and closed-loop design objectives presented
in Section 4.3.2, each rule in this module will change the constraint value of the open-loop gain and fire the open-loop design module to satisfy the modified open-loop constraint set. Once the open-loop singular values have been reshaped, the new $H^\infty$ controller will be recalculated to reconstruct the new closed-loop system. The new closed-loop system will be evaluated to determine whether or not it is an improvement on the previous design. If the modified open-loop constraints do not lead to an improvement of the closed-loop, the modification will be cancelled and the original open-loop constraint will be restored by backtracking from the design history data. Otherwise, the resultant changes will be added to the design history data. The design on improvement of the closed-loop frequency domain constraint will stop if no more rules can be applied or all the rules fail to yield an improvement.

```
rule comp_s_peak_by_crossover
if in_tol of reso_peak of cl_freq_spec.S is no
then reduce_the_crossover_frequency
and fire_open_loop_design_module
and build_cinf_and_closed_loop_system
and check_for_improvement_or_retract_the_design_step.
```

Figure 4.8: Rule For Reducing The Peak Magnitude Of Sensitivity By Reducing The Crossover Frequency

Figure 4.8 shows the listing of a rule from the closed-loop frequency domain design module. The function of this rule is to compensate for the peak magnitude of the sensitivity function by reducing the crossover frequency. Once the condition part (the resonance peak of the sensitivity function is not in tolerance) has been verified through backward chaining, the action part of the rule will be fired to perform the relevant task to reduce the peak of the sensitivity as described in the previous paragraph.
4.6.3 Closed Loop Time Domain Design Rule Knowledge Base

The closed-loop time domain design module can assist the user in modifying the open-loop singular values to satisfy the desirable specification of reference signal tracking form $r(s)$ to $y(s)$ in Figure 4.5. All the rules with their corresponding functions in this module are listed in Table 4.4.

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Function Of The Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>risetime.compensation</td>
<td>Reduce the rise time of the response from $r(s)$ to $y(s)$ by increasing the crossover frequency</td>
</tr>
<tr>
<td>settling_time.compensation</td>
<td>Reduce settling time of the response from $r(s)$ to $y(s)$ by increasing the crossover frequency</td>
</tr>
<tr>
<td>overshoot.compensation</td>
<td>Reduce the peak magnitude of the response from $r(s)$ to $y(s)$ by reducing the crossover frequency</td>
</tr>
</tbody>
</table>

Table 4.4: Rules From The Closed-Loop Time Domain Design Module

In a similar manner to the rules in the closed-loop frequency design module, each rule modifies the crossover frequency constraint and reshapes the open-loop singular values using the open-loop design module. A new $H^\infty$ controller will be calculated based on the new shaped plant to construct a new closed-loop transfer function from $r(s)$ to $y(s)$. The new closed-loop system will be evaluated to determine whether it is an improvement of the previous design. Each design step will be cancelled if the constraint which has been treated by the rule does not improve. For each successful step, the resultant changes will be added to the design history data.

Figure 4.9 shows the listing of a rule from the closed-loop frequency domain design module. The function of this rule is to compensate for the rise time of the responses from $r(s)$ to $y(s)$ by increasing the crossover frequency. Once the condition part (at least one of the rise time is not satisfied) has been verified through backward chaining, the action part of the rule will be fired to perform the relevant task to improve the rise time as described in the previous paragraph.
rule risetime.compensation
if unsatisfied.risetime.of.r.to.y.exist
then increase.the.crossover.frequency
and fire.open.loop.design.module
and built.cinf.and.r.to.y.closed.loop.system
and check.for.improvement.or.retract.the.design.step.

Figure 4.9: Rule For Reducing The Rise Time Of Responses From r(s) To y(s)

4.7 Rule Knowledge Bases For The Two Degree Of Freedom LSDP Design

The rule knowledge base for the two degree of freedom LSDP design can be viewed as an extension of its one degree of freedom counterpart. The major advantage of the two degree of freedom LSDP design is the scope for the user to incorporate the closed-loop time domain specification directly into the process of designing the controller. As mentioned in Section 4.5, a MATLAB function can be coded for finding a suitable first order transfer function which reflects the closed-loop time domain specification. Furthermore, we recall that the modification of the reference model can cause excessive control signal to the plant input. The main function of the expert system is to help the user to tune the reference model so that all the closed-loop time domain requirements of the tracking of reference signals can be satisfied without violating the allowable magnitude of the control signals. The rules for designing the two degree of freedom controller can be classified into 3 major groups:

1. Time Domain Control Signal Tuning Rule Knowledge base
2. Closed-loop Time Domain Design Rule Knowledge base
3. Closed-loop Frequency Domain Design Rule Knowledge base
The functions of each rule knowledge base will be examined more closely in the following subsections.

4.7.1 Time Domain Control Signal Tuning Rule Knowledge Base

The function of this rule knowledge base is to assist the user in reducing the magnitude and the rate of the control signals i.e. the step responses from \( r(s) \) to \( u(s) \) in Figure 3.13. The rule of thumb to decrease the rate and the magnitude of the control signal is to decrease the crossover frequency. The rules in this knowledge base are listed in Table 4.5.

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Function Of The Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>tdf_control_signal_reduction</td>
<td>Reduce the magnitude of the response from ( r(s) ) to ( u(s) ) by decreasing the crossover frequency</td>
</tr>
<tr>
<td>tdf_control_signal_rate_reduction</td>
<td>Reduce rate of the response from ( r(s) ) to ( u(s) ) by decreasing the crossover frequency</td>
</tr>
<tr>
<td>tdf_control_signal_reduction_fail</td>
<td>To give a warning message to the monitor if at least one of the control signal constraint (rate or magnitude) is unsatisfied and the rule knowledge base fail to bring it to satisfaction</td>
</tr>
</tbody>
</table>

Table 4.5: Rules From The Time Domain Control Signal Tuning Rule Knowledge Base

The third rule serves as a warning whenever the expert system fails to reduce the control signal to a satisfactory level if the reduction of the open-loop singular values crossover does not reduce the magnitude or rate of the control signal. At this point, the user may have to relax the specification or modify the control configuration to continue the controller design process.

Figure 4.10 is the syntax of the rule for reducing the control signal by decreasing the crossover frequency of the open-loop singular values.
Chapter 4. Development of An Expert System For Robust Controller Design

4.7.2 Closed-Loop Time Domain Design Rule Knowledge Base

The closed-loop time domain design rule knowledge base was built to aid the user in tuning the reference model of the two degree of freedom LSDP framework in Figure 3.13, so that the specification of the closed-loop reference signal tracking from \( r \) to \( y \) can be met. The design heuristic involved is to reduce the rise time of the reference model for the closed-loop system to follow. Therefore, a faster reference model is used at each design step until the specifications of reference tracking are satisfied with the control signals remaining within the bounds. In addition, past experience suggests that, in the case of MIMO system, cross channel interactions can be tuned by changing individual elements in the \( \rho \) matrix. For example, in the case of a two input and two output system, the interaction effect of the induced signal from the second channel when a step function is applied to the first input can be reduced by increasing the value of the first element in the \( \rho \) matrix. Table 4.6 lists all the rules grouped to the closed-loop time domain design rule knowledge base.

The rule for compensating the rise time is listed in Figure 4.11. The rule verifies the existence of an unsatisfied risetime constraint by backward chaining. If there exists at least one unsatisfied rise time constraint, the action part of the rule will find the relevant channel and assign it to the variable \texttt{CHANNEL}. The rise time of the reference model in the corresponding channel will be speeded up for the synthesis of the new controller. A

```
rule tdf.control.signal_reduction
if unsatisfied_control_signal_exist
then decrease.the.crossover_frequency
and fire.open_loop.design.module
and built.cinf.and.r.to.y.closed.loop.system
and check_for.improvement.or.retract.the.design.step.
```

Figure 4.10: Rule For Reducing The Control Signals
Table 4.6: Rules From The Closed-Loop Time Domain Design Rule Knowledge Base

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Function Of The Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>tdf_interaction_compensation</td>
<td>Reduce the magnitude of the cross coupling response by increasing the value of the relevant element of $\rho$ matrix</td>
</tr>
<tr>
<td>tdf_risetime_compensation</td>
<td>Improve the rise time specification for the responses from $r(s)$ to $y(s)$ by decreasing the crossover frequency</td>
</tr>
<tr>
<td>tdf_settletime_compensation</td>
<td>Improve the settling time specification for the responses from $r(s)$ to $y(s)$ by decreasing the crossover frequency</td>
</tr>
</tbody>
</table>

new $r$ to $y$ transfer function is built and tested for improvement, and consequently the decision to keep or retract the design step will be made.

```
rule tdf.risetime.compensation
if unsatisfied.risetime.specification.exist
then find.the.channel.which.the.risetime.is.unsatisfied(CHANEL)
and decrease.the.rise.time.of CHANNEL
and built.tdf.cinf.and.r_to_y.closed_loop.system
and check.for.improvement.or.retract.the.design.step.
```

Figure 4.11: Rule For Improving The Rise Time In Two Degree Of Freedom Design

4.7.3 Closed-Loop Frequency Domain Design Rule Knowledge Base

The main purpose of this rule knowledge base is to tune the closed-loop frequency domain constraints of two degree of freedom closed-loop system. This rule knowledge
base only handles the constraints treated by the one degree of freedom closed-loop frequency domain design rule knowledge base described in Section 4.6.2. In fact, the same set of rules is used for tuning the closed-loop frequency domain constraints by adjusting the open-loop singular values. A flag is attached to the rule firing procedure to indicate the current design framework so that different sets of design data (one or two degree of freedom design) are manipulated by the rules. The purpose of setting up this link to the rule knowledge base is to provide a way for the user to tune the closed-loop frequency domain constraints when the two degree of freedom LSDP approach is used directly without trying the one degree of freedom design initially. In some cases, it has been found useful to set up bounds for the closed-loop frequency constraint to ensure good performance in the time domain closed-loop transfer functions.

4.8 Overall Design Flow

In the last two sections (Sections 4.6 and 4.7), individual rule base for treating the constraints in specific domain using different degrees of freedom LSDP approaches were developed. The final and the most important task is to link together all the rule knowledge base to devise a systematic design flow, so that a more generic way of performing robust controller design can be reached. Figure 4.12 shows how this task is fulfilled. Typically, the designer will start with a plant model and a brief description of the design specifications. Based on the design specifications, a set of frequency domain open-loop singular value constraints can be estimated and the time domain specification can be outlined. The open-loop singular value constraints will be used for shaping the open-loop singular values of the plant model. With the shaped plant, a one degree of freedom $H^\infty$ robust stabilizing controller can be synthesized, and subsequently, all the closed-loop transfer functions can be built. At this point, the limitations on the allowable magnitude and the rate of the control signal are checked to ensure that they are within the specified limits. If the control signal is found to be excessive, then the crossover frequency constraint is reduced and the open-loop singular values have to be reshaped. Once the control signals are within their limits, the tracking performance of the closed-loop transfer function is determined. If the tracking performance meets the
Figure 4.12: The Overall Design Flow

specifications, the design ends with the achieved controller. However, if the closed-loop system is found to be too sluggish, the crossover frequency is increased after allowing for the control signal limits. This iterative design tuning process continues until a controller is found to satisfy all the closed-loop time domain specification, or after a fixed number of iterations, such a controller is found to be non-existent. In the latter case, the design that produces the best tracking performance which simultaneously satisfies the limitations on the control signals will be available to the designer.

Alternatively, the two degree of freedom design framework can be employed to tune for the tracking performance. The first task in the two degree of freedom design framework is to build a reference model for specifying the model matching property in the design of the two degree of freedom $H^\infty$ controller. A first order reference model is synthesized...
Chapter 4. Development of An Expert System For Robust Controller Design

analytically according to the specifications of reference signal tracking as discussed in Section 4.5. With the shaped plant and the reference model, a two degree of freedom $H^\infty$ controller is synthesized for building the closed-loop reference tracking transfer function. The control signals are then checked to see whether they are within the required limits. In a similar manner to the one degree of freedom design, the control signal is brought within limits by reducing the crossover frequency of the open-loop singular values. Once the control signals are within bounds, the closed-loop reference tracking performance is tuned by modifying the reference model as described in Section 4.7.2. The two degree of freedom design process continues until all the reference tracking specifications are satisfied with the control signals staying within their limits, or no further improvement is possible by modifying the reference model.

4.9 Summary

In this chapter, the development of an expert system for robust controller design had been described. A method for organizing the time and frequency domain data involved in the generic control system design was explored. Control system design heuristics based on the shaping of open-loop singular values were examined. A method of relating the closed-loop design objectives to the required shape of the open-loop singular values was sought and the resultant loop shaping heuristics were then incorporated into systematic robust controller design approaches based on both the one degree of freedom and the two degree of freedom LSDP techniques introduced in Chapter 2. The rule knowledge bases used for capturing the systematic design heuristic were then described. The rule knowledge bases were then linked together to devise a systematic design flow to perform the robust controller design.
Chapter 5

EXPERT SYSTEM IMPLEMENTATION

5.1 Introduction

This chapter discusses the implementation of an expert system for computer-aided controller design using the facilities provided by an expert system shell (flex), an object-oriented window programming language (PWXPCE), a CACSD package (MATLAB) and the C programming language. It begins by listing the important criteria to be considered for the selection of the expert system shell in Section 5.2, and this is followed by the overall structure of the expert system in Section 5.3. Discussions of various components of the expert system which concentrate on the aspect of practical implementation and co-ordination of all the software packages follow in Sections 5.4, 5.5, 5.6 and 5.7. In particular, emphasis is placed on Section 5.6 which describes the formulation of the design specification and the manipulated data, and on Section 5.7 which presents the rule knowledge bases and their inferencing mechanisms that assist the user in dealing with conflicts in the performance constraints.

5.2 Selection Of The Expert System Shell

Expert systems can be developed using logic programming languages such as PROLOG, C++ or Lisp. Nevertheless, the development of an expert system from scratch using these languages is generally laborious and requires a great deal of man-power and time. Fortunately, there are numerous powerful expert system development toolkits, generically known as expert system shells, available to be purchased, which provide a more
productive and inexpensive way to develop a task-specific expert system. The justifications for using an expert system toolkit, and customizing it to implement a controller design expert system are summarized as follows:

I. An expert system development toolkit can provide a quick means of developing a working prototype which can then be refined over time for improvement.

II. With the development toolkit, the minimum requirement of programming capability is required and only little prior experience in building a complete expert system is needed.

III. The pre-defined inference engine and the formal syntax of the knowledge base, allows the user to concentrate on developing the domain-specific (in this case, the controller design) knowledge base.

IV. A well written expert system toolkit is generally reliable, resulting in shorter debugging time whilst normally being sophisticated enough for the complexity of the task.

V. An expert system development toolkit generally provides extensive support facilities for debugging and has sufficient built-in functions which are crucial for rapid-prototyping as described in I.

VI. The resulting system will be easy to maintain and expand.

In choosing a suitable expert system shell, the following criteria must be matched with the primary expectations for implementing an intelligent design system:

1. The shell should provide a multi-paradigm programming environment which includes frame system, inference engine with forward and backward chaining and a pre-defined structure of the rule knowledge base.

2. The structure of the rule knowledge base must be extensive and sophisticated enough to represent the complete control system design problem. Yet, it should also be flexible, in the sense that it can be customized easily to meet the specific needs of defining the abstractions involved in the controller design problem.
Chapter 5. Expert System Implementation

Therefore, an expert system toolkit which is written and executed on top of a well-known logic programming language such as PROLOG or C++ is preferred.

3. Communications with external software should be relatively easy, in particular with MATLAB which will play a key part as the numerical computation engine for the control system design. Nevertheless, the shell should also support a certain level of simple arithmetic computations by itself.

4. The shell should have flexible user interface facilities. It should possess some standard built-in functions to build a window based graphical interface. As a significant part of the expert system is devoted to interaction with the user, a comprehensive display facility is crucial to the success in implementing an expert system.

5. The shell should be able to support a large knowledge base with a modest requirement of memory space. Control system design spans a complex and huge search space, and thus requires a considerable knowledge base. Memory requirement for an expert system is conventionally large due to the large knowledge base supported. However, it will be impractical for others to adopt the developed expert system if the memory space needed is too large.

6. The shell should be able to run under a common operating system (such as UNIX-OS, Solaris, Mac-Format, MSDOS or Microsoft Windows) and a computer hardware (such as SUN SPARC workstation, Apple Computer and IBM-PC or compatible personal computer).

7. The expert system developing environment should provide a means to generate an executable version of the developed expert system. This feature is very important, so that the end product can exist as a stand alone software and be independent of the developing environment which normally occupies a large disk space and is license-restricted.

An expert system toolkit, flex, was chosen as the expert system development shell as it was the only one found to satisfy all the requirements listed above. It is written in
Quintus PROLOG and runs on a UNIX SPARC workstation under the Sun operating system version SunOS 4.1.13_U1 and onwards. The requirement of the memory size is modest, and it was found that it is more than sufficient to have a memory space of 24Mb for the expert system to execute with acceptable speed.

5.3 Structure of the Expert System

![Diagram of the Expert System Structure]

The implemented expert system is called the Robust Controller Design Expert System (RODEX) [23]. In the early stage of the project, a prototype of RODEX was developed as an aid to gain experience in improving the expert system. The prototype was extended and improved during the entire software development life cycle to a fully operational package to design robust controllers for multivariable systems. The structure of the expert system is shown in Figure 5.1.

On the expert system side, flex has been used as the expert system development tool. MATLAB has been used to provide control system analysis and design facilities. An
external program interface which is written in C-language has been built to facilitate
the data communication between the two packages. The same interface can be used
for exporting and importing data through a graphical interface via a set of external
data files. The graphical interface was built to enable the creation of a window-based
user interface. The window based user interface is mainly written in a UNIX Open
Window environment development language, PWXPCE, which is loosely coupled with
Quintus PROLOG. Therefore the majority of the components that constitute RODEX
are sitting on top of the Quintus PROLOG platform which gives the advantage of
efficient and fast data transfer among the individual components. Frame knowledge
base is used to provide a hierarchical data organization. Rule knowledge base is used
to represent all the heuristics related to controller design. The main purpose of having
external data files is to save design related data in the ASCII format. All data saved
in the files can be transferred to other machines and can be reloaded in different design
sessions. We will examine each of these components in more details in the next sections
in the aspect of practical implementation and philosophy behind the motivations of
building up each component.

5.4 User Interface

An expert system designed for robust control system design is generally much more
demanding in terms of graphical user interface requirements than many conventional
expert systems because:

1. Most of the analysis and simulation of control systems require graphical output,
   therefore, the conventional question and answer format of the user interface will
   not be sufficient.

2. Multiple windows are desirable for different types of data available to the user for
   conveying different types of information. For instance, the user may find it useful
   to be able to look at the MATLAB command window to inspect the procedural
   operation of the control system design process. Additionally, the user may also
   wish to check the currently displayed data from the command window of the
expert system shell, to monitor the search methods used by the expert system in
designing a controller.

3. Different windows are also useful for different purposes. For example, the user
should be able to input data and check the display data in separate windows so
that he/she can type while the output is being displayed.

4. Inputting the numerical data necessary for control system design, is in general
cumbersonome, especially when huge data structures involving large matrices and
polynomials are involved. Thus, the user interface requires means to access data
from external sources. The most convenient way to do so is by reading from
and writing to external ASCII files where matrices can be saved and reloaded as
required over the entire control system design session. The same difficulty will
also be encountered when displaying data to the user. The available size of the
display window is normally not sufficient to display a large dimension matrix or
vector. Therefore, scrolling through the output will be very useful to see the entire
output.

Novice and experienced users of expert systems will have different requirements for
an user interface. Experienced users will normally wish to have access to detailed
information on the state of the design and have the complete freedom to choose from all
the possible design alternatives. Novice designers tend to be overwhelmed by too much
information given and when too many possibilities are offered in the available design
methods. Therefore, assistance is required in interpreting the given information and
guidance is needed in selecting the options of design technique. A beginner may even
require help with the language or functions used to perform control systems analysis
and design.

The user interface of RODEX is largely menu-command driven to facilitate the use of
the expert system by all levels of users. The user interface has been designed with the
following features:

1. It is window-based and mouse driven such that all the functions of the expert
   systems are displayed clearly to the user and can be executed by clicking on the
window buttons or menu.

2. Windows for displaying data are separated from the data input window. Separate windows are used for displaying different information. For instance, the user can click a button in the user interface to switch between PROLOG and MATLAB command windows.

3. The user can load pre-defined ASCII format data files which may contain the plant model or specifications directly from the user interface by selecting the appropriate menu item.

4. All the windows for data display have scrolling facilities so that window contents can be scrolled to view.

5. The user interface is directly linked to the MATLAB to display graphical output from the MATLAB for graphical plots in time and frequency domain. Moreover, MATLAB commands can also be entered and executed through the user interface.

The implemented user interface is shown in Figure 5.2. The expert system gives the experienced users almost direct access to all the functions of the LSDP design approach, with comprehensive and detailed descriptions of steps taken during the entire design process.

To assist beginners and novice users, there are commands which perform the design with various levels of automation ranging from semi-interactive to fully automatic. For the novice user, the expert system allows the user to choose the degree of freedom of the controller and the option to tune the controller in frequency domain and/or in time domain. For beginners, the user interface offers assistance throughout the whole controller design process, from loading the model, defining various important specifications, designing the controller to construct all the essential closed-loop transfer functions and performing the necessary system analysis.
5.5 Implementation Of Numerical Computation Engine

Most of the problems solved by expert systems are qualitative and non-numeric in nature (e.g. medical diagnosis, mineral analysis, chess playing and etc) which only require symbolic manipulations and very moderate numerical computation. As we have seen in Chapter 3, robust control system design is a numerical intensive domain due to the following computational requirements:

- Large quantities of matrix computations,
- Complex algebraic manipulations and numerical calculations,
- Finding roots of polynomials,
- Plotting frequency and time domain responses.

PROLOG and the expert system shell have a very limited provision for performing mathematical computations due to their insufficiency in supporting the burden of design.
calculations listed above. It would be very unwise and extremely resources demanding (both time and financial) to develop our own software for all the design calculations. Many commercial packages have been developed to meet these demands. To gain maximum benefit from the available software packages in our research group to facilitate the mathematical computation routines, it is wiser to choose a numerical library, which can provide us with the maximum computational efficiency, to be integrated into the PROLOG environment.

In this project, an interactive mathematical package, MATLAB, has been adopted to be the numerical computation engine for the expert system. The selection of MATLAB to be used for this purpose is a natural first choice after considering the following factors:

- It is available to members of our research group for some time, therefore, by using it, we can cut the cost of purchasing extra numerical software packages.
- At the same time, MATLAB is getting increasingly popular among the practitioners in the control engineering community. This will not only increase the chance of the targeted users to be able to use the developed expert system, but also help to build up their confidence in using the expert system.
- The syntax of MATLAB is gaining recognition to be the quasi standard programming language for control design within the control engineering community, due to the growing extent of control literatures at present which include the MATLAB scripts.
- MATLAB is powerful in numerical calculations particularly for matrices manipulation and has good graphical simulation facilities which are very essential in the control system design.
- It can be expanded easily and has an efficient use of the program segment. The user can expand or change the MATLAB m file which is separated from the environment of the expert system. Further m files can be added to suit the special purpose of individual application. In addition, data transfer can be made within the MATLAB
Chapter 5. Expert System Implementation

environment through external files. This helps to ease the load of the expert system from maintaining a huge quantity of data.

- Another very desirable feature is that, up to a certain extent, the data structure in MATLAB is treated in a similar manner to the tree and list data organization of PROLOG. This feature alleviates the speed of data processing when data is transferred between MATLAB and PROLOG.

The objective of integrating MATLAB into the expert system environment is to provide the following features:

- To be able to invoke MATLAB whenever it is required by the user. Once the MATLAB is invoked, it should be kept on standby such that whenever a request is made by the expert system, it will perform the requested computation.

- MATLAB is used as a background processor, thus its command window should be hidden from the user. Nevertheless, graphical windows for displaying the simulation responses must be fully accessible by the user when a new graph is plotted.

- Data should be able to transfer to and from MATLAB in a pre-defined format.

- Direct communication between MATLAB and other components of the expert system is required especially for the rule knowledge base, user interface and frame knowledge base.

MATLAB version 4.0 onwards with its Engine Library [45] enables the user to exchange data to and from the MATLAB environment. The MATLAB Engine Library contains a set of subroutines which allow the user to call MATLAB from other programs, thereby employing MATLAB as a computation engine. The MATLAB engine operates by running in the background as a separate process from other processes, therefore, instead of linking the whole of MATLAB into the expert system, only a small engine communication library is encapsulated by an external interface written in C program. With the engine library, communication with the MATLAB can be done by using direct UNIX pipe command or remote shell (rsh) for remote execution.
Chapter 5. Expert System Implementation

5.6 Data Organization With Frame Knowledge Base

All the design related data in the expert system are hierarchically organized by a frame knowledge base. The frame structure can be viewed as an inverted tree with the top node as a general object and branches out with levels of objects with increasing specialization and more defined properties. The frames should be carefully designed, with the goal of achieving a small set of general frames. From these frames, other instances can be derived as specializations or collections of the basic frame set; such an approach aids clarity and maintenance by reducing the number of independent definitions in the system.

5.6.1 Representation of Data in Frames Knowledge Base

The most important data to be captured by the frame knowledge base is the set of constraint-type data which has been discussed in detail in Section 4.2. The constraint-type frames are used for defining constraints of the control system specifications in the time and frequency domain. All the control system specifications are first organized in a structural framework by the frame knowledge base. The top node of the specification frame has been named in the most generic term as the control system specifications, from which two distinct branches can be identified: time domain specification and frequency domain specification. From these two domain specification frames, instances of specialized specifications can be attached accordingly. The open-loop $L$, closed-loop $S$, closed-loop $T$ and closed-loop $KS$ are instances of the frequency domain specification frame, whereas, the instances of references to outputs, references to control signals and disturbances to outputs can be grouped under the time domain specification frame. Instances of constraints can be attached to the specification instances to individually characterize the specifications. For example, the constraint instances of settling time, rise time and overshoot can be attached to the specification instance of references to outputs. An example of the specification frame is illustrated by Figure 5.3.

Within the various instances of constraints, suitable attributes can be attached to uniquely define the instances. For example, attributes of req.value, actual.value, error, in_tol and type sufficiently define the constraint instance of risetime.
In addition, since our control system design methodology involves matrix manipulation, essentially, matrix-type data has to be represented in the frame knowledge base. Matrix-type frames are created to define matrices, state space systems and transfer function matrices. Data relating to these entities are stored in both PROLOG and MATLAB. In MATLAB, matrices are stored in the natural way of MATLAB syntax, whereas, in PROLOG, the dimensions and the name of the MATLAB objects are stored as attributes which are attached to matrix-type frames and instances.

\[
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6
\end{bmatrix}
\]

Then in PROLOG, an instance of the matrix frame is created with slot values:

- `name_in_matlab` : 'a'
- `number_of_rows` : 2
- `number_of_columns` : 3

The maintenance of this information is essential to perform operations with the matrix.
Chapter 5. Expert System Implementation

from PROLOG. The matrix element values are stored in MATLAB only; a similar situation exists for state space systems and transfer function matrices.

5.6.2 Routines for Frames Data Processing

In considering the type of processing needed to interact with the frames, it is useful to use the same grouping as the frame types. A set of procedures was built to interact with constraint-type frames, which are named as indicators, and another set to interact with matrix-type frames, which are named as modifiers. Furthermore, there is a set of procedures for miscellaneous frame manipulation. Figure 5.4 indicates the architecture of the data flow within the expert system. The frame processing procedures are further classified into three levels.

![Diagram of the data flow within the expert system](image)

Figure 5.4: Architecture Of The Data Flow Within The Expert System

The lowest level consists of functions written in MATLAB command language which deals with numerically-oriented operations such as manipulating state space systems and retrieving values from vectors. Hence, most of the procedures in the lowest level
Chapter 5. Expert System Implementation

are written and stored as the MATLAB function file. In addition, the procedures which call for the operations to be performed in MATLAB will be used to invoke the procedures in this level.

The top procedure level defines functions which are directly used by the rule knowledge base, and thus they are usually synthetically sugared to make the rule base more readable. For example, the following rule gives a clear statement of a rule for adding integrators to the open-loop system:

\[
\text{if open loop integrating action is required}
\text{and open loop integrating action not present}
\text{then add integrators to the open loop}
\]

In this rule, a top level procedure effectively replaces the first condition with

\[
\text{if integrating_required of integrating of open_loop_spec is equal to yes}
\]

The intermediate level of procedure handles interaction among all the frames. Essentially, the procedures in this level perform as an interface between the top and the lowest level procedures. The procedures reside in this level manipulate the data of the instances from matrix-type and constraint-type frames. The data from the instances is retrieved and processed by intermediate level procedures to be used by the top and the lowest level procedures. For example, the rule knowledge base uses a procedure which makes a copy of a state space system using the call of:

\[
\text{ss_copy(system_new, system_old)}
\]

The \text{ss_copy} procedure extracts the names of the matrices of \text{system_old} and \text{system_new} and performs assignment operations in MATLAB. The intermediate level therefore provides a language for frame manipulation within the expert system environment.

5.7 Rule Knowledge Base And Inference Mechanism

The entire design rule knowledge base can be partitioned into a set of modules:

1. Base Module
Chapter 5. **Expert System Implementation**

2. One Degree of Freedom Module

3. Two Degree of Freedom Module

![Diagram of Rule Knowledge Base]

Each module has its specific role to contribute to the overall control system design process. The function of the base module is to setup the model and design specification for controller design. The other two modules stand as controller design modules with different degrees of freedom as described in the previous chapter. There is a meta-level rule knowledge base in each of the design module to monitor and organize the controller design process. The meta rules of the base module monitor and coordinate the design process as a whole. The structure of the entire rule knowledge base is illustrated in Figure 5.5.

Rules in each rule knowledge base have been developed individually according to the outlines described in Sections 4.6 and 4.7. Modulation of the design process makes the rule knowledge bases easier to design and control since the rule selection algorithm is only presented with those rules which are relevant to the part of the design, on which
Chapter 5. Expert System Implementation

it is necessary to focus on. Furthermore, common knowledge is grouped together to make the syntax of the rule knowledge bases more readable. Each rule knowledge base has an associated rule-set and some constraint instances to be monitored. The forward chaining inferencing mechanism is used to fire the rules from each rule knowledge base. Rules are grouped into a rule-set to form a list of rules to be presented to the forward chaining mechanism for potential firing. Only one rule from the rule-set is allowed to be fired at one time based on the general first come first served rule conflict resolution strategy. Generally, there is at least one rule aimed at solving each monitored design constraint where the rule knowledge base is concerned. For example, the open-loop frequency rule knowledge base has rules which define how to achieve a given crossover, low frequency gain, roll-off rate at the crossover frequency and attenuation at the high frequency region. A rule is fired in order to modify its associated constraints from being out of tolerance to be in tolerance; hence if all the constraints monitored by the rule knowledge base are in tolerance, no rule will be fired.

The modularisation of the rule knowledge base divides the whole control system design process into several sub-problems. For each sub-problem, a set of constraints is defined in order to ensure the design improves after each step in the solution of the sub-problem. The selection of a constraint set as a function of the sub-problem is adopted to clarify our design objectives and improve the efficiency. The set of constraints is chosen with care so that each design step taken by firing the rules will progress towards the improvement of the specifications which are tackled by the rule knowledge base.

In contrast, constraints which are omitted from such a constraint set, are those which are known, a priori, will be constant during a rule base operation. For example, during the solution procedure for the open-loop constraints, the closed-loop constraints are not monitored by the open-loop frequency domain rule knowledge base system since they are not part of the problem which the open-loop frequency domain rules address.

Having a proper formulation of the sub-problem with appropriate attachments of constraints to be monitored, the next step is to work towards defining an algorithm for rule selection with each rule-set. The aim is to provide a means of selecting a rule such that the design improves after each rule firing. The successful implementation of rule firing
selection algorithms depends on the following four key factors:

- How does the expert system judge that the design is being improved?
- If the step taken results in a worse design, how is this step retracted to the state before the last design step was carried out?
- If a design step is retracted, how is it ensured that an alternative design step can be taken?
- How to stop the firing of rules which oscillate between two or more rules, each of which results in a worse design?

The following subsections will serve to resolve these questions which we have posed earlier. Before proceeding, the notations which will be used throughout the next few subsections will be defined. Recall that the complete design problem is to find a controller which satisfies all the constraints to be in tolerance. Our approach of modularisation encapsulates design tasks related to a specific design formulation. Furthermore, each module is internally organized to solve several particular canonical sub-problems of the controller design, each of which will be indicated by [SUBP], where SUBP will be replaced by a mnemonic for the sub-problem name. Specifically, let us consider the one degree of freedom design module which solves the sub-problems of open-loop frequency domain constraints [OLF], closed-loop frequency domain constraints [CLF], closed-loop time domain performance constraints [CLT] and closed-loop time domain control signals constraints [CLC]. For each [SUBP], a set of constraints, C[SUBP], will be defined to be monitored to ensure that the design improves after each step in the solution of the sub-problem.

5.7.1 Test for the Improvement of a Design Step

A combined numerical cost function and logical constraint priority test is used to determine whether the design is 'better' after a design change (due to a rule firing).

The condition for an improved design, $D_{\text{improved}}$, is given by:

$$D_{\text{improved}} \text{ is TRUE if :}$$
Chapter 5. Expert System Implementation

\[ J_{\text{new}} < J_{\text{old}} \]

AND

\[ D_{\text{priority-ordering}} \text{ is TRUE} \]

where \( J \) is simply a weighted sum of normalized errors of the monitored constraints \( C[\text{SUBP}] \) in the current design sub-problem which is defined as follows:

\[ J = \sum_{\forall c_k \in C[\text{SUBP}]} FUZ(c_k) \times PRI(c_k) \quad (5.1) \]

for \( FUZ() \) to be the normalized (or 'fuzzy') error of \( c_k \), and \( PRI() \) to be the weighting (or 'priority') of \( c_k \), both of them are listed in Table 4.1.

A logical component of \( D_{\text{improved}} \) is required to ensure that constraints with a higher priority than the one for which we are currently compensating take precedence. The weighted sum \( J \) ensures that only the cost decreases overall; however this could be at the expense of a more important constraint becoming out-of-tolerance.

There are two steps in defining \( D_{\text{priority-ordering}} \):

(a) Before a design step which attempts to solve a constraint \( c_k \), a temporary set \( C_h \) of constraints is constructed which:

- are in the current set of monitored constraints \( C[\text{SUBP}] \)
- have a higher priority than \( c_k \)
- are in-tolerance

(b) After the design step, each member of \( C_h \) is tested; if all of its members are in-tolerance then \( D_{\text{priority-ordering}} \) is TRUE, otherwise it is FALSE

5.7.2 Actions after a Bad Design Step

When the design progresses with \( D_{\text{improved}} \) to be TRUE after each design step, the rule base operation is quite straightforward. The system simply forward chain the current set of rules until no further rules can be fired; this may be because all of the specifications are satisfied, or because we cannot satisfy all the specifications and a limit on rule firing is reached which will be discussed later. However, when \( D_{\text{improved}} \) is FALSE, then the additional requirement for the 'state' of the design process should be known.
Chapter 5. Expert System Implementation

Let the state of the design in sub-problem [SUBP] be $S_{[SUBP]}$. This is defined to have 2 parts: information about the current design parameters, $S^P_{[SUBP]}$, and information about the last rule fired and its effect on the design, $S^R_{[SUBP]}$. $S^P_{[SUBP]}$ contains all (and preferably the minimal) information required to return the design to a previous point; $S^R_{[SUBP]}$ contains information which allows control of the firing of the current rule set. The history of a state as a list containing previous values of the state; the history of $S^P_{[SUBP]}$ is given by $S^P_{[SUBP]}$ and of $S^R_{[SUBP]}$ by $S^R_{[SUBP]}$.

The design parameters for sub-problem [SUBP] are given by $S^P_{[SUBP]}$.

To return the controller design to a previous point, previous values of $S^P_{[SUBP]}$ have to be stored; this is the role of $S^P_{[SUBP]}$; strictly it is only required to store the last state, but this has been extended to a scheme which stores all the past good designs, to allow provision of a design history 'trace' from the system. Design parameters which gave worse designs are not stored. By design parameters, we mean those variables which are modified by the rule base of the current design sub-problem. They are, therefore, not necessarily the same as the controller parameters.

To illustrate this point, consider a SISO problem which involves the design of a controller gain $K$ and time constant $\tau$ in order to satisfy a given closed-loop frequency specification. The closed-loop constraints are to be solved by manipulating the open-loop ones, hence 2 rule bases are defined; one to solve each sub-problem.

Then from the viewpoint of [OLF], the design state $S^P_{[OLF]}$ is the controller parameter list $\{K, \tau\}$ which it manipulates in order to solve the open-loop constraints. This sub-problem therefore maintains the history $S^P_{[OLF]}$ which will have the form $\{\{K_1, \tau_1\}, \{K_2, \tau_2\}, ...\}$.

However, from the viewpoint of [CLF], the design state $S^P_{[CLF]}$ is the open-loop constraint list itself, and hence this sub-problem maintains a history of these constraint values, which may take the form of a crossover and a minimum gain at the low frequency region.

In addition to the design parameter history, it is necessary to know which rules have been fired and whether they give improved designs in order to fully describe the state of the design. For sub-problem [SUBP], $S^R_{[SUBP]}$ is a 2-element list of the form $\{<name>, ...\}$.
<design reference>} where <name> is the name of the rule most recently fired and the <design reference> is an integer defined as follows:

If <name> fired and the design improved,

Then <design reference> is an index into $S_i^{R}[\text{SUBP}]$ i.e. it points to the list of good parameters which the rule <name> produced.

Otherwise <design reference> is the negative constant value, $-1$, implying a bad design from the firing of the rule <name>.

Therefore, $S_i^{R}[\text{SUBP}]$ is a list of lists. For example, for the closed-loop performance subproblem [CLP], we may have:

$S_i^{R}[\text{CLP}] = \{(\text{initial}, 0), (\text{risetime}, 1), (\text{overshoot}, -1), (\text{settling}, 2)\}$

indicating that the design started with an initial 'rule' (this is a constant identifier which is inserted at the start of all such lists to refer to the design starting point; a rule named 'initial' does not exist). A rule named 'risetime' is then fired which gives an improved design, and has index 1 in the associated $S_i^{R}[\text{CLP}]$ design parameter history. Thereafter a rule named 'overshoot' is fired, but gives a worse design as indicated by the $-1$ index value. At this point $S_i^{R}[\text{CLP}]$ is returned to the state it was in just after the 'risetime' rule was fired. The last rule, settling, is then fired to give an improvement to the design, and it has index 2 in the $S_i^{R}[\text{CLP}]$ design parameter history.

The rule history list allows the definition of a rule disabling algorithm which will stop rules from re-firing when they produce a worse design from the current state.

Let $\text{rule name}(S_i^{R}(I))$ be the rule name of the $I$th element of $S_i^{R}$ and $\text{ref}(S_i^{R}(I))$ be the corresponding design reference value; the 'rule disabling algorithm' is shown in Figure 5.6.

Its effect is to disable a rule if it was previously fired in some state with a design parameters, for instance, $S_i^{R}(1)$, to give a worse design, and the current design state is still $S_i^{R}(1)$, after retracting from the bad design state. Therefore, this algorithm prevents the rule firing process from cycling in an infinite rule firing loop.

The same algorithm also provides tolerance of the rule ordering being incorrect; this is useful since the order in which rules must be fired in order to provide a solution is
Chapter 5. Expert System Implementation

Figure 5.6: Algorithm For Rule Disabling

problem dependent. As an example, if three rules have been defined in the order of $r_1, r_2, r_3$; the forward chaining mechanism will select the rules in this order (from the first rule) on every iteration. To solve a design problem it may require the firing of the rules in the order $r_2, r_3, r_1$. Starting with design parameter state $S^p(0)$ and with $S^p_{tmp}$ a temporary (discarded) parameter state, the sequence illustrated by Table 5.1 is obtained.

The re-enabling of $r_1$ is a direct result of the state changing; disabling of a rule only occurs for the current $S^p$ state; once that state changes (i.e. the design improves, since bad $S^p$ values are not stored), any disabled rules are re-enabled. Thus the correct solution is obtained even when the rules are defined in the wrong order for a particular problem.

To retract a design step, the last element in $S^p_{SUBP}$ is retrieved; since this history
Table 5.1: An Example of Rule Firing Sequence Which Shows The Effect of Rule Disabling Algorithm

contains only good parameter states, the last element is the best parameter set found so far for \( \text{SUBP} \). The retrieved parameter state is then used to reformulate the open or closed-loop for which \( \text{SUBP} \) is designing. In Table 5.1, retraction is implicitly occurring to replace the bad states given by in rows 1 and 3, thereby returning the state to \( S^{(0)} \) in row 1 and to \( S^{(1)} \) in row 3.

The first two cases above follow from the rule disabling algorithm. The last case reflects an additional condition which is needed to define the resources available for solving each constraint. For example it may be found that adding a pole-zero compensator to the open-loop transfer function provides an improved design; however we need to limit the number of such compensators. In another case there are rules such as ‘to reduce control signals, multiply crossover by \( \frac{3}{2} \)’. Clearly it is necessary to define how many times such an action can be performed.

Although in these cases, another constraint may become out-of-tolerance and thus contribute to a worse overall design, this is not guaranteed; hence rule firing limits are an insurance that the rule base will terminate at some point.

5.7.3 Complete Rule Firing Control

Given the previous definitions, an algorithm for rule selection and firing within each sub-problem is now defined; this is illustrated in Figure 5.7.
The rule disabling algorithm is an important part of this complete operation; it both provides the rule order insensitivity as mentioned above and ensures that the rule base stops firing at some point. There are three possible reasons for the rule base stopping:

- All of the sub-problem constraints are satisfied and hence none of the rules fire.
- For all of the rules which address the remaining out-of-tolerance constraints, there is a \(-1\) entry in \(S^R\), that is, all of these rules give a worse design.
- All of the rules which address the remaining out-of-tolerance constraints have reached their firing limits.
Chapter 5. Expert System Implementation

5.8 Communication with External Programs

To facilitate the transfer of data and passing the execution of functions among software packages in the expert system, an external program interface is constructed. An efficient interface to external software is essential for a high performance computer aided controller design system. The main difficulty in building the interface is due to the different formats in data storing and processing of each component of the expert systems, which are listed in Table 5.8. Detailed considerations of data handling of packages which form the expert system have to be carried out before coding the external program interface. This is important not only for the compatibility of the data transfer between the software packages, but also to ensure the ease of its future expansion to accommodate additional software packages in the expert system.

<table>
<thead>
<tr>
<th>Packages</th>
<th>Data Handling Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB</td>
<td>Matrices &amp; Vectors</td>
</tr>
<tr>
<td>PROLOG</td>
<td>Lists &amp; Atoms</td>
</tr>
<tr>
<td>PWXPCE</td>
<td>Objects</td>
</tr>
<tr>
<td>Flex</td>
<td>Lists &amp; Atoms</td>
</tr>
<tr>
<td>External Data Files</td>
<td>ASCII text format</td>
</tr>
</tbody>
</table>

Table 5.2: Data Handling Format For Various Packages Of The Expert System

Furthermore, the success of the external program interface will not only rely on the transferability of data between packages, the speed of data transfer is also of our main concern.

Based on Figure 5.1, the main functions of the external program interface are:

1. To allow access of matlab data from the graphical interface and the rules from the rule knowledge base.

2. To allow access of matlab functions from the graphical interface and the rules from the rule knowledge base.
Chapter 5. Expert System Implementation

3. To enable the user to write and read data to and from the external data files via the graphical interface.

PROLOG and MATLAB allow the user to access external programs via customizing C-language or FORTRAN codes. The C-language interface has been chosen to build the data transfer interface between Prolog and MATLAB. The C codes receive and check the data format from a software platform before the data are converted to the appropriate format and transfer them to another software platform accordingly. Most of the data conversion code is tedious but straight-forward. The advantage of using the C-language for this task attributes to the ease of conversion between pointer data format to MATLAB vector data format and the list data format of Flex and Prolog. In addition, the PWXPCE package provides the essential data conversion functions from object data to pointer format which facilitates the data transfer between PWXPCE and C codes. Therefore the data transfer between MATLAB and the expert system is done via a MATLAB data transfer procedure which is coded in C-language. Nevertheless, the command for executing a MATLAB function within the Prolog environment is not as straight-forward as the transfer of data between them. The call of a MATLAB function involves not only the execution of the function itself which is normally stored as a m file, but also the coordination of input arguments to be processed and the return arguments from the function after the execution of the calling process. To overcome this difficulty, several Prolog predicates to call MATLAB functions via the C-language, with the capability of data passing and return were written. MATLAB input arguments are collected as a list in the Prolog and Flex environment to be accepted by the C-language, which will be converted to appropriate strings of characters, before passing them to the MATLAB function as input arguments. The name of the MATLAB function return arguments is pre-defined in the Prolog environment and attached to the MATLAB call function whenever a MATLAB function is executed from Prolog. Finally, a MATLAB function calling procedure via Prolog has been written to enable the user to call MATLAB functions from the user graphical interface window. The schematic diagram of the resultant external interface is shown in Figure 5.8.
Initially Prolog predicates were used to read and write data from and to the external ASCII data file. It was found that file handling in this way resulted in a very slow file storing and retrieval process speed. This is mainly due to the fact that Prolog predicates in turn use the embedded standard C-language for its file handling system. The data in external files are only accessed by the user via the graphical interface. To improve the processing speed, the Prolog predicates were replaced by a C-language routine which was written to enable the graphical interface to directly save and read data in external files.

5.9 Summary

In this chapter, implementation issues of an expert system for robust controller design have been presented. Criteria to be considered for selecting an expert system shell was discussed. The structure of the expert system implemented using the chosen expert system shell was presented. The motivation and the construction of components of the expert system were examined. Structural data organization using frame knowledge bases was emphasized. The control issues of the rule firing cycle were discussed. An external program interface for coordinating the data communication among the packages used by the expert system was described.
Chapter 5. Expert System Implementation

Figure 5.8: The Schematic Diagram Of The External Program Interface
NUMERICAL TECHNIQUES FOR
MULTIOBJECTIVE OPTIMIZATION

6.1 Introduction

Most of the real world problems require the simultaneous optimization of multiple and competing objectives. The solutions to such problems are usually computed by combining them into a single criterion to be optimized, according to some utility functions. However, in many cases the utility function is not well known prior to the optimization process. The whole function should then be treated as a multi-objective problem with equally important objectives. In this way, a number of solutions can be obtained to provide the decision maker with an insight into the characteristics of the problem before a final solution is chosen. It is well known that similar to other engineering design problems, control system design can be naturally formulated into constraint optimization problems, the solutions of which will characterize admissible designs. However, the optimization problem derived is usually very complicated with many unknowns, many non-linearities and in most cases, it has more than one conflicting design aims which need to be simultaneously achieved. It is also known that the direct parametrization approach of the controller will increase the complexity of the optimization problem.

In this chapter, some numerical optimization techniques are proposed which can be combined with the LSDP methods described in Chapter 3 to design robust controllers with multi-objective and conflicting constraints. Section 6.2 introduces the Method of
Chapter 6. Numerical Techniques For Multiobjective Optimisation

Inequalities framework developed by Zaldan [79] where the performance indices are defined as a set of inequalities. The solution which satisfies all the inequalities can be found by employing a simple numerical search algorithm. Section 6.3 presents another emerging opportunistic optimization technique known as the Genetic Algorithms. Numerical optimization schemes presented in these two sections will be applied to a benchmark control problem in Section 6.4 and a comparison of the results will be examined.

### 6.2 The Method of Inequalities

Performance specifications for control system design are frequently given in terms of algebraic or functional inequalities. For instance, a typical control design problem may require the closed-loop system to achieve the following constraints with response to a step input:

1. Rise Time < 5.0 seconds
2. Settling Time < 7.5 seconds
3. Maximum Overshoot < 1.2

In such cases, it is obviously more intuitive and convenient to express the design constraints explicitly using a set of inequalities.

The Method of Inequalities (MOI) is a computer-aided multi-objective design framework developed by Zakian [79], where all the design performance requirements are expressed as a set of algebraic inequalities and the aim of the design is to simultaneously satisfy these inequalities. The design problem is formulated as

$$\phi_i(p) \leq \varepsilon_i \text{ for } i = 1, 2, ..., n \quad (6.1)$$

where $\varepsilon_i$ are real numbers, $p \in P$ is a real vector $(p_1, p_2, ..., p_n)$ chosen from a given set $P$ and $\phi_i$ are real functions of $p$. The functions $\phi_i$ are the objective functions, the components of $p$ represent the design parameters and $\varepsilon_i$ are the design goals chosen by the designer to represent the largest tolerable values of $\phi_i$. The aim is to search for an acceptable design, $p$, which satisfies the set of inequalities.
Each inequality $\phi_i(p) \leq \epsilon_i$ of the set of inequalities (6.1) defines the set $S_i$ of points in the $q$-dimensional space of real valued vectors $\mathbb{R}^q$ and the co-ordinates of this space are $p_1, p_2, \ldots, p_q$, so

$$S_i = \{p : \phi_i(p) \leq \epsilon_i\} \quad (6.2)$$

The boundary of this set is defined by $\phi_i(p) = \epsilon_i$. A point $p \in \mathbb{R}^q$ is a solution to the set of inequalities (6.1) if and only if, it lies inside every set $S_i$, $i = 1, 2, \ldots, n$ and hence inside the set $S$ which denotes the intersection of all the sets $S_i$,

$$S = \bigcap_{i=1}^{n} S_i \quad (6.3)$$

$S$ is called the admissible set and any point in $S$ is called an admissible point denoted by $p_s$.

The objective of the optimisation is to find a point $p_s$ such that it belongs to the admissible set $S$. Such a point which satisfies inequalities (6.1) is regarded to be a solution. The point, $p_s$, is generally not unique unless the admissible set $S$ is a point in the space of $\mathbb{R}^q$. In addition, if the admissible set is empty, there is no solution to the problem. It is then necessary to relax the boundaries of some of the inequalities by increasing some of the values of $\epsilon_i$ until at least one admissible point $p_s$ exists.

For control system design, the function $\phi_i(p)$ may be functionals of the system step response, such as the rise time, settling time, and the overshoot, or functionals of frequency responses, such as the bandwidth or the crossover frequency. They can also be used to represent measures of stability and robustness of the system, such as the
gain margin or bounds on various representations of plant perturbations as discussed in Chapter 3. These inequalities are also very useful in representing critical bounds arising from the physical constraints of the system, such as the maximum control effort. Furthermore, the constraints on the design parameters $p$ which define the parameter set $P$ may also be included in the inequality set $S$. For instance, it might be necessary to limit the search from all the stable controllers or to search only from a specific range of the design parameters.

The design parameter, $p$, may parametrize a controller with a fixed structure, for example, $p = [p_1, p_2]$ may be used to parametrize a PI controller $p_1 + p_2/s$. Alternatively, $p$ may be used to parametrize the weighting functions required by the analytic optimization methods (such as the $H_\infty$ optimization approach) to provide a mixed optimization approach [76, 78, 77].

The solution to the set of inequalities (6.1) can be obtained by means of numerical search algorithms. Several algorithms for the solution of the MOI have been proposed by Whidborne et al. [73], including the original Moving Boundaries Process (MBP) algorithm by Zakian and Al-Naib [79] which will be presented in the next subsection. Generally, the design process employing the MOI is interactive and iterative. The results of the search algorithm employed by the computer provide essential information to the designer about conflicting design requirements and the extent of success in finding the solution. Additionally, the designer can adjust the bounds of inequalities to explore various feasible solutions to the problem. Ng [51] suggests the use of graphical displays to provide key information to the designer about the progress of the search process and the trade-offs involved in satisfying the conflicting design requirements. An important property of the MOI optimization method over the conventional single functional minimization approach is the shift of emphasis from the speed of convergence in the neighbourhood of the minimum to the likelihood of obtaining a feasible solution. Therefore, the MOI optimization problem is greatly influenced by the initial search point provided by the designer. A good starting point which is near to the admissible set $S$, can improve the likelihood of directing the search to a feasible solution.
Chapter 6. Numerical Techniques For Multiobjective Optimization

### 6.2.1 Moving Boundary Process

The original numerical search algorithm used to solve the MOI optimization was proposed by Zakian and Al-Naib [79], and is known as the Moving Boundaries Process (MBP). It starts from an arbitrary initial point and proceeds to an admissible point in an iterative manner.

Let $p^k$ denote the value of the parameter vector, $p$, at the $k$-th step and $S^k$ is the set formed by the inequalities $\phi_i(p) \leq \phi_i(p^k)$ with the boundary of $\phi_i(p) = \phi_i(p^k)$, i.e.,

$$S^k = \{ p^k : \phi_i(p) \leq \epsilon_i^k \}$$  \hspace{1cm} (6.4)

$$\epsilon_i^k = \begin{cases} 
\epsilon_i & \text{if } \phi_i(p^k) \leq \epsilon_i \\
\phi_i(p^k) & \text{if } \phi_i(p^k) > \epsilon_i 
\end{cases} \text{ for } i = 1, 2, ..., n \hspace{1cm} (6.5)

$$S^k = \bigcap_i S_i^k$$ \hspace{1cm} (6.6)

$S^k$ being the set of parameter vectors for which some but not all of the inequalities constraints are satisfied. A step is taken from the point $p^k$ to a trail point $p^{k*}$ which is generated by an algorithm developed by Rosenbrock [62]. The algorithm is simple but robust because it does not require the computation of gradients.

If for every $i = 1, 2, ..., n$ the boundary defined by $\phi_i(p) = \phi_i(p^k)$ is closer to the admissible set, $S$, or not further away from the previous boundary of $S^k$, i.e.,

$$\phi_i(p^{k*}) \leq \epsilon_i^k \quad i = 1, 2, ..., n \hspace{1cm} (6.7)$$

then the point $p^{k*}$ is accepted and becomes the new point $p^{k+1}$, i.e.

$$p^{k+1} = p^{k*} \hspace{1cm} (6.8)$$

we then have

$$\max_i \phi_i(p^{k+1}) \leq \max_i \phi_i(p^k) \hspace{1cm} (6.9)$$

therefore it is obvious that

$$S \subseteq S^{k+1} \subseteq S^k \hspace{1cm} (6.10)$$

implying that the boundary of the set in which the parameter vector is located has been moved towards the admissible set, as illustrated in Figure 6.2.
If any of the inequalities (6.7) do not hold, another trial point will be made from $p^k$ until a success occurs.

The process is terminated when after a sufficient number of successful steps, the boundaries of $S^k$ converged to coincide with the admissible set, i.e.

$$
\varepsilon_i^k = \varepsilon_i \quad \text{for} \quad i = 1, 2, \ldots, n
$$

(6.11)

The MBP is simple yet robust and has been used successfully in solving the MOI optimization problems for many years. Practical experiences have proven that it has finite convergence for a large class of practical problem [73]. However, a characterization of this class has not been theoretically established.

6.2.2 $H^\infty$ Loop Shaping Design with the Method of Inequalities

The nature of the $H^\infty$ LSDP design approach makes it amenable to combine with the MOI to form a mixed optimization technique for the design of robust controllers. In the LSDP method described in Section 3.5.3, the weighting functions are chosen by considering the open-loop response of the weighted plant, so effectively the weights $W_1$ and $W_2$ are the design parameters. Therefore, the design problem can be formulated as in the MOI, with the weighting parameters used as the design parameters to satisfy a set of closed-loop performance indices. The designer does not have to choose the order or structure of the controller, but instead chooses the structure and order of the weighting functions. Distinct from most of the $H^\infty$ optimization techniques, the optimal
Controller for the weighted plant can be synthesized directly from the solution of two algebraic Riccati equations and the \( \gamma \)-iteration can therefore be avoided.

The controller design problem of the mixed optimization approach can be stated as follows:

**Problem 6.1**

*For the system of Figure 3.11, find a pair of fixed order weighting functions \( W = (W_1, W_2) \) with real parameters \( w = (w_1, w_2, \ldots, w_p) \) such that*

\[
\gamma_o(W) \leq \varepsilon_{\gamma}
\]

where

\[
\gamma_o(W) = \inf_{K \text{ stabilizing}} \left\| \begin{bmatrix} W_1^{-1}K \\ W_2 \end{bmatrix} (I - G_s K)^{-1} \tilde{M}_s^{-1} \right\|_{\infty}
\]

and

\[
\phi_i(W) \leq \epsilon_i \quad \text{for} \quad i = 1, 2, \ldots, n
\]

where \( \phi_i(W) \) are functions of the closed-loop system, \( \epsilon_{\gamma} \) and \( \epsilon_i \) are real numbers representing the desired bounds on \( \gamma_{o} \) and \( \phi_i \) respectively.

A systematic mixed-optimization design procedure to solve Problem 6.1 is given by [76] as follows:

i) Define the plant model \( G \) and define the functionals \( \phi_i \).

ii) Define the bounds of \( \epsilon_{\gamma} \) and \( \epsilon_i \).

iii) Define the structure of the weighting functions \( W_1 \) and \( W_2 \). Bounds should be placed on the values of \( w_i \) to ensure that \( W_1 \) and \( W_2 \) are stable and minimum phase to prevent undesirable pole/zero cancellation. The order of the weighting functions, i.e. the value of \( q \), should initially be kept small.

iv) Define the initial value of \( w_i \) based on the open-loop frequency response of the plant.
v) Implement the MBP in conjunction with Equations (3.49) and (3.50) to find a $W$ which satisfies Inequalities (6.12) and (6.14). If a solution is found, the design is satisfactory. If no solution is found, either increase the order of the weighting functions, relax one or more of the bounds $e_2$ and $e_1$, or try again with different initial values of $w_i$.

vi) With satisfactory weighting functions $W_1$ and $W_2$, the final controller, $K_f$ can be constructed by absorbing $W_1$ and $W_2$ into the $H^\infty$ controller, i.e. $K_f = W_1 K_\infty W_2$.

A design example using the mixed-optimization design procedure will be presented in section 6.4 with the results being compared to other numerical optimization methods which will be presented in the next section.
6.3 Genetic Algorithms

When Darwin published *The Origin of Species* [10], he described a complex and dynamic biological system which embodied multiple interacting environments where creatures with attributes most suited to exploiting available resources fared best and were therefore fittest to survive. His theory has played a key role in the revolution of the search techniques and optimization approaches. Since the early 70s, a number of algorithmic analogies to the natural evolution have been developed as adaptive search mechanisms, capable of dynamically self-adjusting to the idiosyncrasies of the optimization problem encountered. This collection of intelligent computational algorithms is generically known as the Evolutionary Computations (ECs) which include the Evolutionary Programming, Genetic Programming and Genetic Algorithm. Among these three, the Genetic Algorithm is probably the best known variant and most widely applied to the control system design.

This section starts with a basic Genetic Algorithm and proceeds to the Multi-objective Genetic Algorithms for solving optimization problems in control system design. An extension of the Multi-objective Genetic Algorithms is proposed for automating the selection of the controller structure and their parameters simultaneously.

6.3.1 Motivation

Genetic algorithms (GAs) have recently found extensive applications in solving global optimization searching problems. GAs are parallel and global search techniques developed by J.H. Holland and his colleagues at the University of Michigan [32], which emulate the evolutionary process in nature. GAs apply operators inspired by the mechanics of natural selection to a population of binary strings encoding the parameters space and simultaneously evaluate a fix number of points in the parameters space. The idea is that GAs operate a population of individuals, with each individual representing a potential solution to the problem, and apply the principle of survival of the fittest to the population, so that individuals eventually evolve towards better solutions to the problem.

The most important differences of GAs from traditional optimization approaches are:
Chapter 6. Numerical Techniques For Multiobjective Optimization

1. GAs work with a coding of the parameter set rather than the parameters themselves.

2. GAs search from a population of points whereas traditional optimization techniques generally move from a single point towards an admissible point.

3. GAs optimize on objective functions rather than their derivatives or other auxiliary cost functions.

4. GAs use probabilistic transition rules rather than deterministic rules.

As a result of using coding, GAs are conveniently unconstrained by the limitations of other methods such as continuity, existence of derivatives, unimodality and smoothness. In many optimization methods, the user has to search from a single point in the decision space to the next using some rules to determine the next point. These point-to-point approaches are generally a perfect prescription for reaching false peaks in multi-modal (multi peaks) search space. In particular, algorithms which use hill-climbing search techniques may be trapped in local minima since the convexity of the objective functions with respect to the parameters space can not be guaranteed. By contrast, GAs search from a large population of points simultaneously, therefore climbing a number of peaks in parallel; thus the probability of finding a false peak is reduced over methods that go from point to point. Many search techniques require many auxiliary cost functions in order to perform the optimization. For example, the gradient-based techniques need derivatives in order to climb to the current peak and the $H^\infty$ optimization approaches need the calculation of a frequency weighted cost function for translating some desired time domain performance objectives which GAs do not need in performing the optimization. GAs do not require assumptions of continuity and differentiability on the search space. Therefore they overcome problems encountered by the gradient-based or hill climbing search approaches. To perform an effective search, they only require the values of the exact objective functions to be optimized. This characteristic renders the GAs as a canonical optimization method more suited to a broader class of optimization problem than many other search schemes. In addition, unlike many other methods, GAs use probabilistic transition rules to guide their search by employing random choice as a
Chapter 6. Numerical Techniques For Multiobjective Optimization

tool to guide a search towards regions of search space with likely improvement. The
four differences of the GAs over the traditional optimization approaches contribute to
a robust search method that can be used for solving a broad class of optimization
problems.

In Holland's original GAs [32], one parent is selected on a fitness basis (i.e. the higher
the fitness value, the better the chance of it being selected) while the other parent is
chosen randomly. They are then mated by choosing a random crossover point \( X_c \), and
the offspring inherits the pre-\( X_c \) section from one parent followed by the post-\( X_c \) section
of the other. One of the existing population is chosen at random to be replaced by one
of the offspring and this reproduction plan is repeated as many times as desired. Many
other variations on this theme are possible, for example, both parents can be selected
on a fitness basis, mutation can be applied, more than one crossover point could be used
and even more sophisticated operators can be employed. Nevertheless, in many cases,
simple GAs involving a single crossover point with mutation have proven to be powerful
enough to provide near optimum solutions. In the next subsection, we will examine a
multi-objective GA and various genetic operations in more detail.

6.3.2 Multi-Objective Genetic Algorithm (MOGA)

Most of the GAs have been used only for single objective problems. To extend the us­
ability of GA for multi-objective optimization problems, several multi-objective schemes
have been proposed [42, 64, 18]. In particular, Fonseca and Fleming [18] have suggested
a multi-objective approach called the Multi-Objective Genetic Algorithm (MOGA) which
is an extension of the Pareto-based ranking technique used by Goldberg [25]. Their for­
mulation maintains the genuine multi-objective nature of the problem, and is essentially
the scheme which will be studied further in the following text.

Multi-objective optimization (MO) seeks to optimize the components of a vector val­
ued cost function. Unlike the single objective optimization, the solution to the MO is
generally not a single point, but a set of points known as the Pareto-optimal set. Each
point in this set is optimal in the sense that no further improvements can be achieved
in one of the components of the cost valued vector without leading to any degradation
Chapter 6. Numerical Techniques For Multiobjective Optimization

in at least one of the remaining components. Each element in the Pareto-optimal set constitutes a non-inferior solution to the MO problem. By maintaining a population of solutions, MOGA can search for several non-inferior solutions in parallel, which makes it very attractive for solving MO problems.

To restrict the size of the near Pareto-optimal set and to formulate a practical setting to the MOGA, Fonseca and Fleming have formulated the problem in a very similar way to the MOI. The MOGA is set into a multi-objective context by means of the fitness function. The MOGA problem could be stated similarly to the MOI problem as below:

**Problem 6.2**

*Given a set of objective functions*

\[ \phi^k = [\phi^k_1, \phi^k_2, \ldots, \phi^k_n] \]

*Find a set of \( M \) admissible points \( p^k, k = 1, 2, \ldots, M \) such that*

\[ \phi^k_i \leq \varepsilon_i \quad \text{where} \quad k = 1, 2, \ldots, M \quad \text{and} \quad i = 1, 2, \ldots, n \]

*where \( \phi^k \) are non-dominated, \( \varepsilon_i \) are real numbers representing the desired bounds on the set of objective function.*

An individual \( k \) with a set of objective functions \( \phi^k = [\phi^k_1, \phi^k_2, \ldots, \phi^k_n] \) is non-dominated in a population of \( N \) individuals if there is no other individual \( j = 1, 2, \ldots, N \) and \( j \neq k \) such that

\[ \phi^j_i \leq \phi^k_i \quad \forall i = 1, 2, \ldots, n \quad \text{and} \quad \phi^j_i < \phi^k_i \quad \text{for at least one} \quad i \]

Closed-loop performance indices such as the rise time, the settling time and the overshoot, as well as the robust stability requirement given by Equation (6.12) can be included in the set of objective function.

The algorithm of the MOGA for solving Problem (6.2) can be stated as follows:

1. Create a chromosome population of \( N \) individuals.

2. Decode chromosomes to obtain phenotypes \( p^k \in \mathcal{P} \) where \( k = 1, 2, \ldots, N \) and \( \mathcal{P} \subset \mathbb{R}^n \).
3. Calculate index vectors $\phi^k$ where $k = 1, 2, ..., N$.

4. Rank individuals and calculate fitness function $f^k$ where $k = 1, 2, ..., N$.

5. Make selection of $N$ individuals based on fitness and pair them for the reproduction of offsprings.

6. Perform crossover on chosen individuals.

7. Perform mutation on some randomly selected individuals.

8. With the new chromosome population, return to (2) until the set of $M$ admissible points have been found.

Similar to the single objective GA, many variations of the MOGA have been proposed with different schemes for chromosomal representation, ranking and fitness determination, number of crossover point and selective pressure of mutation. Generally, the heuristic for deciding which scheme to use for each particular case is based on the rule of thumb, as well as the criteria for selecting the population size $N$ and the various probability constants for the various schemes.

To create a population of $N$ individual chromosomes, the real parameter space $\mathcal{P} \subset \mathbb{R}^n$ is discretized into a mesh of discrete points, each point is assigned a code based on a binary representation of its real value. For every $N$ individuals, a random point is selected from the discretized mesh and assigned with the binary code accordingly for each parameter in the real value parameter set. The binary codes are then concatenated together into a single binary string representing the chromosome.

Fonseca and Fleming [18] have proposed a fitness scheme for MOGA to solve the MOI problem which maintains the genuine multi-objective nature of the problem. To use the scheme, their definition of preference is required. This is the concept of one individual being preferable to another.

**Definition 6.1 Preferable**

Let two individuals $a$ and $b$ have objective function sets of $\phi^a$ and $\phi^b$ respectively, and let the set of design goals be $\epsilon = (\epsilon_1, ..., \epsilon_n)$.

Three possible cases can occur with respect to individual $a$:
Chapter 6. Numerical Techniques For Multiobjective Optimization

1. Individual a satisfies none of the inequalities
   i.e. \( \phi_i^a > \epsilon_i \) \( \forall i = 1, 2, ..., n \)
   In this case, a is preferable to b if and only if \( \phi_i^a \leq \phi_i^b \) \( \forall i = 1, 2, ..., n \) and there exists at least one \( \phi_i^a < \phi_i^b \) such that \( \phi_i^a < \phi_i^b \)

2. Individual a satisfies some of the inequalities
   i.e. there exists at least one \( \phi_i^a \leq \epsilon_i \) and there exists at least one \( \phi_i^a > \epsilon_i \) such that \( \phi_i^a \geq \epsilon_i \)
   In this case, individual a is preferable to individual b
   (a) if \( \phi_i^a \leq \phi_i^b \) \( \forall i \) such that \( \phi_i^a > \epsilon_i \)
      and there exists at least one \( \phi_i^a > \epsilon_i \) such that \( \phi_i^a < \phi_i^b \).
   (b) if \( \phi_i^a = \phi_i^b \) \( \forall i \) such that \( \phi_i^a > \epsilon_i \)
      and \( \phi_i^a \leq \phi_i^b \) \( \forall i \) such that \( \phi_i^a \leq \epsilon_i \)
      and there exists at least one \( \phi_i^a \leq \epsilon_i \) such that \( \phi_i^a < \phi_i^b \)
      or if there exists at least one \( \phi_i^a > \epsilon_i \) for some \( i \) such that \( \phi_i^a \leq \epsilon_i \)

3. Individual a satisfies all the inequalities
   i.e. if \( \phi_i^a \leq \epsilon_i \) \( \forall i = 1, 2, ..., n \).
   In this case, individual a is preferable to individual b if
   (a) \( \phi_i^a \leq \phi_i^b \) \( \forall i = 1, 2, ..., n \).
      and there exists at least one \( \phi_i^a < \phi_i^b \)
   (b) there exists at least one \( \phi_i^a > \epsilon_i \)

With Definition 6.1, all the individuals are assigned with a rank depending on how many individuals are preferable to an individual as below:

**Definition 6.2 Multi-objective Ranking**

Let \( r^i \) denotes the rank of an individual \( i \), and \( p_f^i \) be the number of individuals in that generation which are preferable to individual \( i \), then

\[ r^i = p_f^i + 1 \]
Chapter 6. Numerical Techniques For Multiobjective Optimization

For fitness calculations, the entire population is sorted according to their rank. A fitness value \( f^i \) \( \forall i = 1, 2, \ldots, N \) such that \( 0 \leq f^i \leq 2 \), is assigned to each individual by a linear interpolation from the best individual \( (f^{\text{best}} = 2) \) to the worst individual \( (f^{\text{worst}} = 0) \) according to the fitness definition which is defined as follows:

**Definition 6.3 Multi-objective Fitness Calculation**

Consider an individual \( i \) with the rank of \( r^i \). Let \( n_{r^i} \) be the number of individuals with the rank \( r > r^i \) and \( n_{eq} \) be the number of individuals with a rank \( r = r^i \). The fitness of individual \( i \), \( f^i \) can then be calculated as

\[
f^i = \frac{2n_{r^i} + n_{eq}}{N - 1}
\]

To illustrate the multi-objective ranking and fitness calculation, Figure 6.3 shows an example of 10 individuals with two objective functions \( \phi = (\phi_1, \phi_2) \) and their bounds.

![Figure 6.3: Example Of A 10 Individuals With 2 Objective Functions](image)
as $\epsilon = (\epsilon_1, \epsilon_2)$. The preference relationships between each pair of individuals with their subsequent ranking and fitness values are shown in Table 6.1.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>a is preferable to</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>b is preferable to</td>
<td>√</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>c is preferable to</td>
<td>√</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>d is preferable to</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>e is preferable to</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>f is preferable to</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>g is preferable to</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>-</td>
<td>x</td>
<td>-</td>
<td>x</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>h is preferable to</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td>x</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>i is preferable to</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td>x</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>j is preferable to</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>rank 6</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>fitness</td>
<td>0.222</td>
<td>1.111</td>
<td>1.111</td>
<td>0.222</td>
<td>0.666</td>
<td>1.555</td>
<td>1.888</td>
<td>1.888</td>
<td>1.111</td>
<td>0.222</td>
</tr>
</tbody>
</table>

Table 6.1: Preference Relationships With Ranking And Fitness Values

Many different selection schemes have been proposed. The most widely adopted scheme is probably the weighted probability approach, commonly known as the roulette wheel selection method. Each individual is assigned to a slot on the wheel in proportion to its fitness value. The wheel is then spun to decide a winning individual. The probability $\rho$, of an individual being selected is

$$\rho = \frac{f^i}{\sum_{j=1}^{N} f^j} \quad (6.15)$$

where $f^j$ represents the fitness of an individual $j$.

The simplest crossover scheme is the single point crossover originally proposed by Holland [32]. A single crossover point for each selected pair is randomly chosen, and the right-most bits of the two individuals are exchanged to produce two offsprings. For example, suppose two chromosomes $a_1|a_2|a_3|a_4|a_5$ and $b_1|b_2|b_3|b_4|b_5$ are the parents which have been chosen by some selection schemes. A crossover point is chosen at random among the numbers from 1 to 5. Suppose the crossover point is 2, then the two
Chapter 6. Numerical Techniques For Multiobjective Optimization

138

Figure 6.4; Example Of Chromosome Cross-over

Offsprings will be \( a_1a_2a_3a_4a_5 \) and \( b_1b_2b_3b_4b_5 \), as illustrated by Figure 6.4.

The mutation operation provides the opportunity to reach the area of the search space which might not possibly be reached by applying the crossover operation alone. The mutation operation consists of simply flipping bits of the chromosomes of each individual with a probability value of \( \rho_m \). For example, an individual with a chromosome structure \( \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \) will become \( \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \) if the second and the fifth bits are chosen for mutation.

6.3.3 MOGA with LSDP Controller Design Technique

MOGA has been combined with different controller design techniques for controller synthesis and good results have been reported [9, 73, 42, 43]. In particular Dakev et al. have proposed to combine the MOGA with LSDP design techniques for the fixed order controller design and successful results have been reported [9].

In this thesis, the work of Whidborne et al. [72] has been extended to a systematic optimization procedure using the LSDP and MOGA for treating the robust controller design problem. The resulting controller design problem of MOGA with the LSDP approach can be stated as below:

**Problem 6.3**

*Find a set of \( M \) admissible points, \( p^j \forall j = 1, ..., M \), with weighting functions \( W = (W_1, W_2) \) as the design parameters, such that*
Chapter 6. Numerical Techniques For Multiobjective Optimization

1. all the admissible points satisfy

$$\gamma_\theta(W) \leq \varepsilon_\gamma$$  \hspace{1cm} (6.16)

where

$$\gamma_\theta(W) = \inf_{K \text{ stabilizing}} \left\| \begin{bmatrix} W_1^{-1}K \\ W_2 \\ (I - G_sK)^{-1}\hat{M}_s^{-1} \end{bmatrix} \right\|_{\infty}$$  \hspace{1cm} (6.17)

2. $$\phi_j^i(W) \leq \varepsilon_i$$ for $$j = 1, ..., M$$ and $$i = 1, ..., n$$

where $$n$$ is the number of objective function and $$\phi_j^i$$ are non-dominated.

The key factor to the successful application of MOGA in conjunction with the LSDP lies in the coordination of the objective functions evaluation, the ranking of each individuals and the computation of their corresponding fitness values. A systematic procedure was developed to evaluate the objective functions of each individual and to compute their corresponding fitness values as follows:

Procedure 6.1 Systematic fitness evaluation procedure for MOGA with LSDP controller design approach

(i) For a chromosome in its binary coded form, generate the corresponding weighting functions $$W = (W_1, W_2)$$.

(ii) Build the shaped plant, $$G_s$$, where $$G_s = W_2GW_1$$, where G denotes the nominal plant model.

(iii) Calculate the $$\gamma_\theta(W)$$ of Equation (6.16).

(iv) Synthesize the $$H^\infty$$ controller, $$K^\infty$$, for the shaped plant by Equation 3.50 and build all the relevant closed-loop transfer functions.

(v) Compute all the closed-loop performance indices, i.e. $$\phi_j^i(W)$$ for the present chromosome.

(vi) Repeat Procedures (i)-(v) for the entire population of chromosomes.

(vii) Find the preference among all the individuals using Definition 6.1.
(viii) Based on the individuals' preference, rank all the individuals accordingly using Definition 6.2.

(ix) Use Definition 6.3 to calculate the fitness values of all the individuals.

To solve the controller design problem using the MOGA with the LSDP approach, as stated by Problem 6.3, a design algorithm was developed as follows:

Algorithm 6.1 MOGA for controller design using the LSDP approach

1. Define the nominal plant model, \( G \).
2. Define the set of objective functions, \( \phi_i \).
3. Define the required values of \( \epsilon_r \) and \( \epsilon_t \).
4. Define a fixed structure of the weighting functions \( W = (W_1, W_2) \) and the parameters to be searched through the real parameter space.
5. Define the parameters for conducting the MOGA.
6. Generate the first population randomly and use Procedure 6.1 to compute the objective functions and evaluate the fitness value of each individual.
7. Select the parents by using the roulette wheel selection scheme based on the probability of selection defined by Formula (6.15).
8. Generate a new population of chromosomes by cross-overs and mutations to replace the previous population.
9. Use Procedure 6.1 to compute the objective functions and evaluate the fitness value of each individual.
10. Repeat Steps (8)-(10) until Problem 6.3 is solved or the maximum allowable generation is reached.
Chapter 6. Numerical Techniques For Multiobjective Optimization

Although the combination of the MOGA and the LSDP technique forms a powerful design tool for the controller design, due to the fact that the order and structure of the weighting functions \( W_1 \) and \( W_2 \) are pre-fixed by the designer, the performance of the mixed-optimization approach is limited by the choice of weights, which may in turn affect the optimality of the design. A Multi-Layer Multi-Objective Genetic Algorithm (MLMOGA) will be considered in the next subsection to optimize over both the orders and coefficients of the weights, \( W_1 \) and \( W_2 \), used in the design.

6.3.4 Multi-Layer Multi-Objectives Genetic Algorithm (MLMOGA)

The Multi-Layer Multi-Objective Genetic Algorithm (MLMOGA) differs from the standard MOGA in its hierarchical structure of the chromosome. Each chromosome consists of two levels of genes. The lower level genes are known as the coefficient genes and it is essentially represented in the same way as the MOGA. The upper level genes are known as the structural control genes which are represented in the form of bits for the decision of the activation of the coefficient genes in the lower level. Figure 6.5 shows an example of such hierarchical structure of the MLMOGA.

![Figure 6.5: Example Of Chromosome Structure Of MLMOGA](image)

In this example the structure of the general transfer function, \( G(s) \), which will be...
Chapter 6. Numerical Techniques For Multiobjective Optimization

searched over the parameters space is

\[ G(s) = \frac{k(s + z_1)(s + z_2)}{(s + p_1)(s + p_2)(s + p_3)} \]

For instance, an individual which possesses the structural control gene and the coefficient gene of \([1|0|1|1]\) and \([2.5|0.1|0.2|0.3|0.4|0.5]\) respectively will represent a transfer function of

\[ G(s) = \frac{2.5(s + 0.1)}{(s + 0.3)(s + 0.5)} \]

The same selection, crossover and mutation schemes, can be applied to the binary string of both layers of the chromosomes as in the standard MOGA. The coefficient genes can be converted to real numbers using a standard binary to real conversion as in the MOGA. The structural control genes are represented in the binary form, therefore conversion between binary and real numbers are not needed. Both layers are used for the calculation of the objective value and subsequently for the fitness value assignment.

6.3.5 MLMOGA with LSDP Controller Design Technique

The MLMOGA can be applied in conjunction with the LSDP techniques to form a flexible and powerful multi-objective mixed-optimization approach for the robust controller design. The idea of using the MLMOGA with the LSDP is to develop a population of Pareto-optimal or near Pareto-optimal solutions with different weighting function structures and parameters. The controller design problem of the MLMOGA with the LSDP techniques can be stated as below:

**Problem 6.4**

Find a set of \(M\) admissible points, \(p^j \forall j = 1, ..., M\), with varying structures of the weighting functions \(W = (W_1, W_2)\), such that

1. all the admissible points satisfy

   \[ \gamma_0(W) \leq \epsilon_7 \]  \hspace{1cm} (6.18)

where

\[ \gamma_0(W) = \inf_{K \text{ stabilizing}} \left\| \begin{bmatrix} W_1^{-1}K \\ W_2 \end{bmatrix} (I - G_s K)^{-1} M_s^{-1} \right\|_{\infty} \]  \hspace{1cm} (6.19)
Chapter 6. Numerical Techniques For Multiobjective Optimization

2. \[ \phi^i(W) \leq s^i \text{ for } j = 1,...,M \text{ and } i = 1,...,n \]
   where \( n \) is the number of objective function and \( \phi^i \) are non-dominated.

A systematic procedure was developed for evaluating the set of objective function and fitness value of each individual in a similar way to the MOGA with the LSDP as follows [24]:

**Procedure 6.2 Systematic fitness evaluation procedure for MLMOGA with LSDP controller design approach**

(i) For a chromosome in its hierarchical binary coded form, generate the corresponding weighting functions \( W = (W_1, W_2) \) based on the information from the structural control and the coefficient genes.

(ii) Build the shaped plant, \( G_s \), where \( G_s = W_2GW_1 \), where \( G \) denotes the nominal plant model.

(iii) Calculate the \( \gamma(W) \) of Equation (6.18).

(iv) Synthesize the \( H^\infty \) controller, \( K_{\infty} \), for the shaped plant by Equation 3.50 and build all the relevant closed-loop transfer functions.

(v) Compute all the closed-loop performance indices, i.e. \( \phi(W) \) for the present chromosome.

(vi) Repeat Procedures (i)-(v) for the entire population of chromosomes.

(vii) Find the preference among all the individuals using Definition 6.1.

(viii) Based on the individuals' preference, rank all the individuals accordingly using Definition 6.2.

(ix) Use Definition 6.3 to calculate the fitness values of all the individuals.

To solve the problem of controller design using the MLMOGA with the LSDP approach, as stated by Problem 6.4, a design algorithm was developed as follows [24]:
Algorithm 6.2 MLMOGA for controller design using the LSDP approach

1. Define the nominal plant model, $G$.

2. Define the set of objective functions, $\phi_i$.

3. Define the required values of $e_r$ and $e_i$.

4. Define the fundamental structure of the weighting functions $W = (W_1, W_2)$ and the parameters to be searched through the real parameter space.

5. Define the parameters for conducting the MLMOGA.

6. Generate the first population by randomly selecting the corresponding structural control and coefficient genes, and use Procedure 6.2 to compute the objective functions and subsequently evaluate the fitness value of each individual.

7. Select the parents by using the roulette wheel selection scheme based on the probability of selection defined by Formula (6.15).

8. Generate a new population of chromosomes by performing cross-overs and mutations on both layers of genes to replace the previous population.

9. Use Procedure 6.2 to compute the objective functions and evaluate the fitness value of each individual.

10. Repeat steps (8)-(10) until Problem 6.4 is solved or the maximum allowable generation is reached.

6.3.6 MLMOGA with TDF-LSDP Controller Design Technique

Alternatively, the MLMOGA has been combined with the TDF-LSDP described in Section 3.5.4, to include the reference model to meet closed-loop time response specifications. The two degree of freedom controller design problem with the MLMOGA is then be stated as:
Chapter 6. Numerical Techniques For Multiobjective Optimization

Problem 6.5

Based on Figure 3.13, with a pre-defined reference model, $M$, find a set of $M$ admissible points, $p^j$ for $j = 1, ..., M$, with varying structures of the weighting functions $W = (W_1, W_2)$, such that

1. all the admissible points satisfy

$$\gamma(W) \leq \epsilon_\gamma$$

where $\gamma(W)$ can be calculated using Equation (3.49).

2. $\phi_i(W) \leq \epsilon_i$ for $j = 1, ..., M$ and $i = 1, ..., n$

where $n$ is the number of objective function and $\phi_i$ are non-dominated.

The procedures for evaluating the set of objective functions and the fitness values were listed as follows [24]:

Procedure 6.3 Systematic fitness evaluation procedure for MLMOGA with TDF-LSDP controller design approach

(i) For a chromosome in its hierarchical binary coded form, generate the corresponding weighting functions $W = (W_1, W_2)$ based on the information from the structural control and the coefficient genes.

(ii) Build the shaped plant, $G_s$, where $G_s = W_2GW_1$, where $G$ denotes the nominal plant model.

(iii) Calculate the $\gamma(W)$ of Equation (3.49).

(iv) By solving the standard $H^\infty$ optimization problem with the plant, $P(s)$, defined in Equation (3.67), find the optimal value of $\gamma$ which satisfies Inequality (3.64) via the $\gamma$-iteration.

(v) Based on the optimal $\gamma$, construct the feedforward and feedback controller, $K_1$ and $K_2$ respectively and build all the relevant closed-loop transfer functions.

(vi) Compute all the closed-loop performance indices, i.e. $\phi_i(W)$ for the present chromosome.
(vii) Repeat procedures (i)-(vi) for the entire population of chromosomes.

(viii) Find the preference among all the individuals using Definition 6.1.

(ix) Based on the individuals’ preference, rank all the individuals accordingly using Definition 6.2.

(x) Use Definition 6.3 to calculate the fitness values of all the individuals.

With the above listed systematic procedure to evaluate the fitness values of the population, an algorithm for solving Problem 6.5 can then be stated as follows [24]:

Algorithm 6.3 MLMOGA for controller design using the TDF-LSDP approach

1. Define the nominal plant model, G.

2. Choose a reference model, $M_o$, to be followed by the closed-loop system.

3. Select a value for $\rho$ in Figure 3.19 to weigh the importance of the property between the robust stabilisation and robust model matching.

4. Define the set of objective functions, $\phi_i$.

5. Define the required values of $\varepsilon_o$ and $\varepsilon_i$.

6. Define the fundamental structure of the weighting functions $W = (W_1, W_2)$ and the parameters to be searched through the real parameter space.

7. Define the parameters for conducting the MLMOGA.

8. Generate the first population by randomly selecting the corresponding structural control and coefficient genes, and use Procedures 6.3 to compute the objective functions and subsequently evaluate the fitness value of each individual.

9. Select the parents by using the roulette wheel selection scheme based on the probability of selection defined by Formula (6.15).
(10) Generate a new population of chromosomes by performing cross-overs and mutations on both layers of genes to replace the previous population.

(11) Use Procedure 6.3 to compute the objective functions and evaluate the fitness value of each individual.

(12) Repeat steps (8)-(10) until Problem 6.4 is solved or the maximum allowable generation is reached.

Although the γ-iteration performed for the synthesis of the two degree of freedom controller is time consuming, the mixed-optimization approach of the MLMOGA with the TDF-LSDP remains attractive. Closed-loop time domain specifications can be explicitly included in the formulation of the controller design problem to transform the controller design process to be more straightforward.

In the next section, various GA schemes are applied to a benchmark control problem and a comparative study will be performed.

6.4 Design Example

This design example is taken from the benchmark control problem of the 1993 IFAC World Congress [26]. Most control problems such as the IFAC 1993 benchmark have stringent closed-loop performance requirements to be met in the presence of significant plant uncertainty, hence closed-loop stability and performance robustness are required. It will be shown in this subsection that both closed-loop performance and a certain level of robustness can be achieved by combining the MOI with the LSDP techniques. This can be done by using the MOI to design the weighting functions required for the analytical LSDP approach.

6.4.1 Plant Models

The IFAC benchmark problem involves a time-varying SISO process where the loop operates at three different stress levels, with higher stress levels inducing larger time
variations. A linearized nominal plant model is given in the form of a transfer function

\[ G(s) = \frac{K(-T_2 s + 1)w_0^2}{(s^2 + 2\zeta w_0 s + w_0^2)(T_1 s + 1)} \]  

(6.21)

where \( T_1 = 5, T_2 = 0.4, w_0 = 5, \zeta = 0.3, K = 1. \)

The transfer function of the plant for each stress level is also given by Equation (6.21) with the parameter variation intervals tabulated in Table 6.2.

<table>
<thead>
<tr>
<th>Stress Level</th>
<th>( \delta T_1 )</th>
<th>( \delta T_2 )</th>
<th>( \delta w_0 )</th>
<th>( \delta \zeta )</th>
<th>( \delta K )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \pm 0.20)</td>
<td>( \pm 0.05)</td>
<td>( \pm 1.50)</td>
<td>( \pm 0.10)</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>( \pm 0.30)</td>
<td>( \pm 0.10)</td>
<td>( \pm 2.50)</td>
<td>( \pm 0.15)</td>
<td>( \pm 0.15)</td>
</tr>
<tr>
<td>3</td>
<td>( \pm 0.30)</td>
<td>( \pm 0.15)</td>
<td>( \pm 3.00)</td>
<td>( \pm 0.15)</td>
<td>( \pm 0.50)</td>
</tr>
</tbody>
</table>

Table 6.2: The parameters variation intervals of the plant at each stress level

The complete transfer function \( G_t(s) \) of the plant for each stress level is given by

\[ G_t(s) = \frac{K(-T_2 s + 1)w_0^2}{(s^2 + 2\zeta w_0 s + w_0^2)(T_1 s + 1)(s^2 + 2\zeta T_3 s + w_0^2)(T_4 s + 1)} \]  

(6.22)

where \( T_1 = \frac{1}{\delta}, T_2 = \frac{1}{\delta}, T_3 = 3, T_4 = 6, \zeta = 0.6. \)

The design can be simulated via a C-language 'black box' simulation code. The code for the supplied unity feedback controller can be replaced by the final design controller.

The following points must be observed:

- Due to noise and parameter variations, the response will be different every time the program is run, especially at the higher stress levels.
- A representative picture of the variations can be achieved by simulating the plant over at least 300 seconds.
- The variations can be achieved by either simulating over a 300 second period, or by simulating several times over equivalently shorter periods. The latter is recommended for convenience.
The final design should be simulated by running the program at least 15 times over a 20 seconds period and plotting the 15 output curves on top of each other in a window with 0 to 20 seconds on the horizontal axis, and with amplitudes from -1.5 to +1.5 on the vertical axis. This should be done separately for each of the three stress levels, yielding a total of three windows with 15 curves each.

6.4.2 Specifications

The set-point to the process is a square wave varying between +1 and -1 within a period of 20 seconds. For each stress level, a controller has to be designed to meet the following specifications:

1. The plant output must be within -1.5 and +1.5 at all times.
2. Zero steady state tracking error.
3. The undershoot/overshoot is preferred to be within 0.2 for most of the time, although occasional larger undershoots/overshoots are acceptable as long as their outputs are within ±1.5.
4. Fast settling time and rise time.
5. The plant input saturation magnitude of ±5.0 must be avoided.

6.4.3 Closed-loop Performance Functionals

A set of closed-loop performance functionals \( \phi_i(p), i = 1, 2, \ldots, 5 \) was defined based on the design specifications given in Subsection 6.4.2. The performance functionals are calculated from the closed-loop time responses of the linear system to a unit reference step input. The performance functionals \( \phi_1(p) - \phi_5(p) \) are measures of the maximum overshoot, maximum undershoot, rise time, settling time and maximum control effort respectively. Let \( y(p, t) \) be the plant output response, and \( u(p, t) \) be the plant input response, the functionals are defined as

\[
\phi_1(p) = \max_i y(p, t) \tag{6.23}
\]
From the specifications given in Subsection 6.4.2, the overshoot, undershoot and control
signal constraints are rigid, therefore $\varepsilon_1$, $\varepsilon_2$, $\varepsilon_3$ are fixed at 1.2, 1.2 and 5.0 respectively.
The rise time and settling time specifications are less rigid, therefore increased or reduced
values of $\varepsilon_5$ and $\varepsilon_4$ are tried on different stress levels to obtain the best result. Generally,
$\varepsilon_5$ is kept between 2 to 3 seconds for both stress levels 1 and 2.

6.4.4 Controller Designs and Simulations

In this subsection, controllers designed using various optimization approaches described
in Sections 6.2.2, 6.3.2 and 6.3.4 for the IFAC 1993 benchmark problem will be pre­
sented. One controller is designed using each optimization approach for every stress
level. The controllers are simulated with the given black box simulation code to generate
time domain closed-loop responses. For each stress level, the approximate rise
time and settling time with maximum overshoot and undershoot values are tabulated
for comparisons.

The MOI and the MOGA are used in conjunction with the 1DOF LSDP techniques to
choose the parameters of the weighting function $W_1$ with a fixed structure of

$$W_1 = \frac{K(s + z_1)(s + z_2)}{s(s + p_1)(s + p_2)}$$

(6.28)

where $K$, $z_1$, $z_2$, $p_1$ and $p_2$ are the parameters to be searched in the real parameter space
and $W_2$ is fixed at 1.

The MLMOGA is used with the 1DOF LSDP and 2DOF LSDP separately to optimize
the structure and the parameters of the weighting function $W_1$. The same generic
structure of $W_1$ as in Equation (6.28) is chosen to be optimized. For both stress levels
1 and 2, a first order reference model of $\frac{2}{s+2}$ is selected for the 2DOF LSDP approach.
In stress level 3, a reference model of $\frac{0.8}{s+0.8}$ is selected to avoid unstable responses. In
all stress levels, the values of $\rho$ for the 2DOF LSDP is fixed at 1 to give equal weight.
Chapter 6. Numerical Techniques For Multiobjective Optimization

on the properties robust stability and model-matching. Generally, a weighting function with the lowest order which satisfies all the specifications (or most of them) is chosen for the black box simulation.

Stress Level 1

Weighting functions designed by various optimization approaches for stress level 1 are tabulated in Table 6.3.

<table>
<thead>
<tr>
<th>Design Methods</th>
<th>Weighting Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOI 1DOF LSDP</td>
<td>(18(s+0.0504)(s+117.102)) (s(s+3.8031)(s+114))</td>
</tr>
<tr>
<td>MOGA 1DOF LSDP</td>
<td>(518(s+30.5289)(s+0.0398)) (s(s+69.6022)(s+13.4884))</td>
</tr>
<tr>
<td>MLMOGA 1DOF LSDP</td>
<td>(3.8516(s+0.2976)(s+0.1674)) (s(s+0.7583))</td>
</tr>
<tr>
<td>MLMOGA 2DOF LSDP</td>
<td>(2.5014(s+0.1969)) (s(s+44.3742))</td>
</tr>
</tbody>
</table>

Table 6.3: Weighting functions chosen for stress level 1

The approximate values of the worst rise time, maximum overshoot and undershoot of the simulations are tabulated in Table 6.4.

<table>
<thead>
<tr>
<th></th>
<th>max. overshoot</th>
<th>max. undershoot</th>
<th>max. rise time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOI 1DOF LSDP</td>
<td>1.33</td>
<td>1.13</td>
<td>2.8</td>
</tr>
<tr>
<td>MOGA 1DOF LSDP</td>
<td>1.22</td>
<td>1.18</td>
<td>2.7</td>
</tr>
<tr>
<td>MLMOGA 1DOF LSDP</td>
<td>1.24</td>
<td>1.18</td>
<td>2.5</td>
</tr>
<tr>
<td>MLMOGA 2DOF LSDP</td>
<td>1.36</td>
<td>1.21</td>
<td>2.1</td>
</tr>
</tbody>
</table>

Table 6.4: The approximate values of the design indices for stress level 1 simulations

All the designs met the specifications. The results of the time varying simulations for various optimization methods are shown in Figures 6.6 - 6.9.
Comparing the results of all the 1DOF LSDP designs, it is obvious that both of the genetic algorithms approaches obtain better performance than the MOI. In addition, under the MLMOGA approach, a better result is observed by using a lower order weighting function than the fixed order weighting function tuned by the MOGA. Due to the model matching properties of the 2DOF LSDP approach, the controller designed by the MLMOGA produces better simulation results; the rise time was faster with the expense of higher overshoots compared to the 1DOF design.

Stress Level 2

The weighting functions designed by various optimization approaches for stress level 2 are tabulated in Table 6.5.

<table>
<thead>
<tr>
<th>Design Methods</th>
<th>Weighting Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOI 1DOF LSDP</td>
<td>( \frac{3.3441(s+1.4751)(s+1.3296)}{(s+3.0009)(s+2.5321)} )</td>
</tr>
<tr>
<td>MOGA 1DOF LSDP</td>
<td>( \frac{17.8188(s+0.1017)(s+0.5292)}{(s+1.2280)(s+0.0252)} )</td>
</tr>
<tr>
<td>MLMOGA 1DOF LSDP</td>
<td>( \frac{3.6776(s+0.1022)}{s} )</td>
</tr>
<tr>
<td>MLMOGA 2DOF LSDP</td>
<td>( \frac{1.8376(s+0.9539)}{(s+8.4240)}(s+0.2653) )</td>
</tr>
</tbody>
</table>

Table 6.5: Weighting functions designed for stress level 2

<table>
<thead>
<tr>
<th>Design Methods</th>
<th>max. overshoot</th>
<th>max. undershoot</th>
<th>max. rise time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOI 1DOF LSDP</td>
<td>1.21</td>
<td>1.20</td>
<td>3.9</td>
</tr>
<tr>
<td>MOGA 1DOF LSDP</td>
<td>1.37</td>
<td>1.17</td>
<td>2.7</td>
</tr>
<tr>
<td>MLMOGA 1DOF LSDP</td>
<td>1.43</td>
<td>1.28</td>
<td>3.9</td>
</tr>
<tr>
<td>MLMOGA 2DOF LSDP</td>
<td>1.23</td>
<td>1.22</td>
<td>2.5</td>
</tr>
</tbody>
</table>

Table 6.6: The approximated values of the design indices for stress level 2 simulations
The results of the time varying simulations for various optimization methods are shown in Figures 6.10 - 6.13. The approximate values of the worst rise time, maximum overshoot and undershoot of the simulations are tabulated in Table 6.6.

All specifications are met by the controllers designed by both of the MOGA 1DOF LSDP and the MLMOGA 2DOF LSDP approaches. Controller designed by the MOI and the MLMOGA with the 1DOF LSDP approach result in unsatisfied rise time specification.

### Stress Level 3

Weighting functions designed by various optimization approaches for stress level 2 are tabulated in Table 6.7.

<table>
<thead>
<tr>
<th>Design Methods</th>
<th>Weighting Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOI 1DOF LSDP</td>
<td>$1.8200(s+0.3851)(s+200.07)$</td>
</tr>
<tr>
<td>MOGA 1DOF LSDP</td>
<td>$0.8305(s+0.4206)(s+643.8408)$</td>
</tr>
<tr>
<td>MLMOGA 1DOF LSDP</td>
<td>$0.8305(s+0.4206)(s+192.24)$</td>
</tr>
<tr>
<td>MLMOGA 2DOF LSDP</td>
<td>$1.0956(s+0.4339)$</td>
</tr>
</tbody>
</table>

Table 6.7: Weighting functions designed for stress level 3

The approximate values of the worst rise time, maximum overshoot and undershoot of the simulations are tabulated in Table 6.8.

The results of the time varying simulations for the various optimization methods are shown in Figures 6.14 - 6.17.

All the designs do not meet the specifications. Nevertheless, all the controllers designed by the GA approaches result in a faster rise time than those designed by the MOI. Comparing the two genetic-based design approaches, the MLMOGA managed to design controllers with a lower order weighting function but faster rise time. All the specifications were improved by the MLMOGA 2DOF LSDP approach.
Chapter 6. Numerical Techniques For Multiobjective Optimization

<table>
<thead>
<tr>
<th></th>
<th>max. overshoot</th>
<th>max. undershoot</th>
<th>max. rise time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOI 1DOF LSDP</td>
<td>1.15</td>
<td>1.20</td>
<td>8.9</td>
</tr>
<tr>
<td>MOGA 1DOF LSDP</td>
<td>1.42</td>
<td>1.35</td>
<td>6.7</td>
</tr>
<tr>
<td>MLMOGA 1DOF LSDP</td>
<td>1.42</td>
<td>1.32</td>
<td>6.2</td>
</tr>
<tr>
<td>MLMOGA 2DOF LSDP</td>
<td>1.37</td>
<td>1.25</td>
<td>5.5</td>
</tr>
</tbody>
</table>

Table 6.8: The approximate values of the design indices for stress level 3 simulations

6.4.5 Design Analysis and Conclusions

As a whole, the controllers designed by using the GA methods result in better closed-loop performance. The two LSDP methods combined with the GA approaches provide controllers that achieve good robust performance for stress levels 1 and 2. The rise time specification at stress level 3 was not met by any mixed optimization techniques. It was shown that at stress level 3, the performance of the controller can be improved by a variable gain adaptive scheme at the expense of some robustness [61].

Although the formulation of the problem using the MOI has simplified the trade-offs between different performance requirements, the success of the controller design still heavily depends on the initial search point. The finding of such a point can be difficult especially in the case of a multi-modal controller design problem. On the other hand, the GA-based mixed optimization techniques have been proven to be useful and efficient in solving controller design problems with multi-objective requirements. In particular, the combination of the MLMOGA with the LSDP methods is found to be effective in designing controllers with varying structures and parameters where the user has the freedom to choose the controller order.

The $H^\infty$ loop shaping approaches provide reasonable performance robustness in designing controllers for plants with large parameter variations. In general, the TDF-LSDP technique is capable of designing controllers which achieve better time domain performance specification than those of the 1DOF LSDP techniques.
6.5 Summary and Discussion

In this chapter, we considered several numerical optimization techniques i.e. the MOI, MOGA and MLMOGA, which were used in conjunction with the LSDP approaches described in Chapter 3. The IFAC 1993 benchmark problem was used to illustrate the applications of these mixed optimization methods to perform the robust controller design. These mixed optimization approaches may be used as a complementary package to the expert system described in this thesis.

The overall design techniques employed by various numerical optimization methods were essentially identical to the systematic design approaches utilized by the expert system. These numerical optimization approaches may provide alternative ways of designing a controller to overcome design problems which can not be resolved by heuristics embedded within the expert system.

The expert system can automatically generate the relevant MATLAB script files for performing the various numerical optimization procedures described in this chapter. The graphical user interface of expert system provides a MATLAB command calling facility for the user to evoke the optimization script files written in MATLAB. The user can therefore execute the optimization process and monitor the search process through the MATLAB command display screen in the graphical user interface.
Figure 6.6: Stress level 1 Simulations: 1DOF LSDP controller designed by MOI

Figure 6.7: Stress level 1 Simulations: 1DOF LSDP controller designed by MOGA
Chapter 6. Numerical Techniques For Multiobjective Optimization

Figure 6.8: Stress level 1 Simulations: 1DOF LSDP controller designed by MLMOGA

Figure 6.9: Stress level 1 Simulations: 2DOF LSDP controller designed by MLMOGA
Figure 6.10: Stress level 2 Simulations: 1DOF LSDP controller designed by MOI

Figure 6.11: Stress level 2 Simulations: 1DOF LSDP controller designed by MOGA
Figure 6.12: Stress level 2 Simulations: 1DOF LSDP controller designed by MLMOGA

Figure 6.13: Stress level 2 Simulations: 2DOF LSDP controller designed by MLMOGA
Figure 6.14: Stress level 3 Simulations: 1DOF LSDP controller designed by MOI

Figure 6.15: Stress level 3 Simulations: 1DOF LSDP controller designed by MOGA
Figure 6.16: Stress level 3 Simulations: 1DOF LSDP controller designed by MLMOGA

Figure 6.17: Stress level 3 Simulations: 2DOF LSDP controller designed by MLMOGA
Chapter 7

DESIGN EXAMPLES

7.1 Introduction

In this chapter, three design case studies are presented by employing the expert system developed in Chapter 4 and 5 to perform control system design. The first case study involves a square plant which is linearized at a particular operating point. The system is open-loop unstable but minimum phase. The aim is to design a controller for the closed-loop system to satisfy a set of performance objectives with a limited closed-loop bandwidth. We use the expert system to design the controller for this plant in an automatic mode which is driven by the prescribed specifications without any intervention from the user. The second example is a four-input, six-output model with the two extra outputs being the rates of two of the four regulated signals. This example is used to illustrate the ability of the expert system to handle non-square models using the automatic mode as in the first example. The third case study is regarding the control of an ill-conditioned distillation system under a range of plant parameter variations. The expert system is used to design a controller to obtain a specified level of robust performance with respect to the allowable model uncertainty. In this example, an interactive mode of controller design is taken where the frequency domain specifications are modified to accommodate the allowable model uncertainty.
7.2 The CH-47 System

The first example is an abstract of the longitudinal model of the CH-47 tandem rotor helicopter. The design objective is to control two measured outputs, i.e. the vertical velocity and the pitch attitude, by manipulating both the collective and the differential collective rotor thrust commands. Therefore, it is a two-input and two-output system with four states. A nominal model for the dynamics relating to these four variables at 40 knots airspeed [29] is used. The state space of the derived model is listed as follow:

\[
A = \begin{bmatrix}
-0.020 & 0.005 & 2.400 & -32.00 \\
-0.140 & 0.440 & -1.300 & -30.00 \\
0.000 & 0.018 & -1.600 & 1.200 \\
0.000 & 0.000 & 1.000 & 0.000
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
0.140 & -0.120 \\
0.360 & -8.600 \\
0.350 & 0.009 \\
0.000 & 0.000
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
0.000 & 1.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 57.30
\end{bmatrix}
\]

\[
D = \begin{bmatrix}
0.000 & 0.000 \\
0.000 & 0.000
\end{bmatrix}
\]

The open-loop poles are located at:

-3.3597
-0.09376 - 0.28987 \text{i}
-0.09376 + 0.28987 \text{i}
+0.38726
Chapter 7. Design Examples

There is a finite zero at -0.01792 and three infinite zeroes. The system is open-loop unstable but minimum phase. Figure 7.1 shows the singular values of the CH-47 model. The LQG design using the full-state Loop Transfer Recovery technique had been performed by Doyle and Stein [13] with satisfactory results. A bandwidth of about 10 rad/s was suggested by them to cope with the unstructured uncertainties. A closed-loop design and a Characteristics Locus Design were reported by Bloch and Postlethwaite [5]. The same system was used by Pang et al. [56] to design a controller for the closed-loop tracking performance using the Reverse Frame Alignment technique.

The same system is used to design a controller using our expert system. For the open-loop singular value loop-shaping design, the open loop singular values are aligned at 10 rad/s. Based on the design done by Pang et al., a list of time response specifications from the command inputs to the measured outputs is estimated as below:

- Rise time specifications
  1. Input 1 to Output 1: 0.60 seconds
  2. Input 2 to Output 2: 0.45 seconds

- Settling time specifications
Chapter 7: Design Examples

1. Input 1 to Output 1: 1.00 seconds
2. Input 2 to Output 2: 0.70 seconds

- Maximum overshoot specifications
  1. Input 1 to Output 1: 5%
  2. Input 2 to Output 2: 25%

Furthermore, strict limits on the actuator outputs are imposed as listed below:

1. Command Input 1 to Plant Input 1: 0.10
2. Command Input 1 to Plant Input 2: 0.65
3. Command Input 2 to Plant Input 1: 2.00
4. Command Input 2 to Plant Input 2: 2.00

7.2.1 LSDP One Degree of Freedom Design

With the time response specifications, the expert system is able to produce a satisfactory result with a one degree of freedom controller. The open-loop rule knowledge base adds integrators to all \( W_1 \) to ensure good steady-state tracking and aligns the crossover frequency at 10 rad/s. Figure 7.2 shows the singular values of the shaped plant.

The closed-loop frequency domain rule knowledge base changes the open-loop of the shaped plant to reduce the peak of the closed-loop sensitivity function from 6 dB to less than 3 dB to ensure good output disturbance rejections. This is done by reducing the slope of the singular values at the aligned crossover frequency to less than -30 db/decade. Figure 7.3 shows the singular values of the final shaped plant and figure 7.4 shows the singular values of the closed-loop sensitivity function. The final closed-loop bandwidth is found to be approximately at 8 rad/s and the closed-loop singular values roll off fast enough after 10 rad/s to cope with unstructured uncertainties. Figure 7.5 shows the singular values of the closed-loop complementary sensitivity function.

The closed-loop time domain rule knowledge base finds that all the closed-loop time domain specifications have been satisfied. Therefore no rules in the rule knowledge
Figure 7.2: Open-Loop Singular Values For Shaped Plant Of CH-47

Figure 7.3: Open-Loop Singular Values of the Final Shaped Plant (CH-47)
base are fired. Figures 7.6 and 7.7 show the step responses of the resultant closed-loop state-space from the command inputs to the measured outputs. The control input of the plant is checked by the time domain rule knowledge base to be satisfactory.

Figure 7.4: Closed-Loop Sensitivity Function of CH-47

Figure 7.5: Closed-Loop Complementary Sensitivity Function of CH-47
Figure 7.6: Time Response of the Closed-Loop System With Step Input At Input 1

Figure 7.7: Time Response of the Closed-Loop System With Step Input At Input 2
7.2.2 LSDP Two Degree of Freedom Design

The two degree of freedom module uses the weighting function $W_1$ which is constructed by the open-loop rule knowledge base of the one degree of freedom design module. The open-loop singular values are again aligned at 10 rad/s. An initial two degree of freedom stabilizing controller is synthesized by the expert system to build all the necessary closed-loop transfer functions. Based on these closed-loop transfer functions, all the specifications of the time and frequency domain contained in the design specification frames are updated. All the frequency domain specifications are found to be satisfactory, therefore the frequency domain rule knowledge base is not fired.

Figure 7.8 shows the singular values of the sensitivity function. Figure 7.9 shows the singular values of the complementary sensitivity function.

![Sensitivity Function TDF (CH-47)](image)

Figure 7.8: Singular Values Of Sensitivity Function (CH-47)

The closed-loop time domain rule knowledge base is triggered to tune the time domain reference signal tracking responses from the command inputs to the measured outputs. The reference model is modified so that specifications for the tracking from the command inputs to the measured outputs are satisfied. The step responses of the final two degree of freedom closed loop system are shown in figures 7.10 and 7.11.
Figure 7.9: Singular Values Of Complementary Sensitivity Function (CH-47)

Figure 7.10: Time Responses of the Closed-loop System (2 DOF) With Step Input At Input 1 (CH-47)
Figure 7.11: Time Responses of the Closed-loop System (2 DOF) With Step Input At Input 2 (CH-47)
7.3. The Bell-205 Helicopter

The model used for this design example is a basic six degree of freedom bell-205 helicopter model which is linearized at 10 knot [31]. The model is a non-square plant of 4 inputs, 6 outputs with 8 states.

The control inputs are listed below with their units:

1. Main rotor collective (cm)
2. Longitudinal cyclic (cm)
3. Lateral cyclic (cm)
4. Tail rotor collective (cm)

The states and their units are described as follows:

- $u$ - forward velocity (m/s)
- $w$ - vertical velocity (m/s)
- $q$ - pitch rate (rad/s)
- $v$ - lateral velocity (m/s)
- $p$ - roll rate (rad/s)
- $r$ - yaw rate (rad/s)
- $\theta$ - pitch attitude (rad)
- $\phi$ - roll attitude (rad)

The outputs which are chosen to be controlled are $[w, \theta, \phi, r]$, with the additional pitch and roll rates feedback to the controller.
The state-space of the linearized model is listed as follows:

\[
A = \begin{bmatrix}
-0.0036 & 0.0300 & 0.2490 & -0.0056 & -0.4154 & -0.0795 & -0.8100 & 0 \\
-0.1841 & -0.4456 & 0.3393 & -0.0512 & -0.1812 & 0.6229 & 0 & 0 \\
0.0062 & -0.0091 & -0.2695 & 0.0066 & 0.2333 & 0.0250 & 0 & 0 \\
0.0149 & -0.0016 & -0.4157 & -0.0544 & -0.3341 & 0.2726 & 0 & 9.81 \\
0.0195 & -0.0116 & -0.8566 & -0.0396 & -0.6855 & 0.1429 & 0 & 0 \\
-0.0184 & -0.0204 & 0.0274 & 0.0692 & -0.3037 & -0.7329 & 0 & 0 \\
0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
0.0741 & 0.1236 & -0.0007 & -0.0025 \\
-1.1351 & 0.0594 & 0.0010 & -0.0084 \\
0.0027 & -0.0673 & 0.0003 & 0.0001 \\
-0.0270 & 0.0027 & 0.1062 & 0.1927 \\
-0.0309 & 0.0054 & 0.2216 & 0.1625 \\
0.1570 & 0.0011 & 0.0318 & -0.4636 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000 \\
0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
\end{bmatrix}
\]
Therefore the plant is open-loop unstable. Figure 7.12 plots the open-loop singular values of the open-loop unstable model.

The closed-loop bandwidth is limited to 10 rad/s due to a lightly damped structural transmission mode of the fuselage at about 14 rad/s. The time domain closed-loop simulation is required to have fast step responses with low overshoot and small cross-channel interactions.

The linearized plant is used by the expert system to demonstrate its capability of designing a controller for the non-square plant. The following time domain specifications which are based on the unit step inputs are used by the expert system as the design constraints:
Figure 7.12: Plots of Open-loop Singular Values of the Bell-205 Model

- Rise time specifications:
  1. Input 1 to Output 1: 2 seconds
  2. Input 2 to Output 2: 3 seconds
  3. Input 3 to Output 3: 3 seconds
  4. Input 4 to Output 4: 2 seconds

- Maximum overshoot specifications:
  1. Input 1 to Output 1: 1%
  2. Input 2 to Output 2: 1%
  3. Input 3 to Output 3: 1%
  4. Input 4 to Output 4: 1%

In addition, the control effort is limited to ±16.5 cm for all plant inputs.
Based on the time domain specifications, the expert system estimated the crossover frequency of the open-loop gain to be 10 rad/s and imposed the requirement of an integrator to be included in the open-loop gain.
7.3.1 LSDP One Degree of Freedom Design

The open-loop rule knowledge base adds integrators to all the diagonal elements of $W_i$ to ensure good steady-state characteristic of unit step signal tracking and elevates the open-loop gain to have maximum cross-over frequency at 10 rad/s. Figure 7.13 shows the plots of singular values of the shaped plant.

\[ \text{Figure 7.13: Open-loop Singular Values Plots of the Shaped Plant} \]

The closed-loop frequency domain rule knowledge base is used to ensure that the closed-loop bandwidth is less than 10 rad/s. The final controller designed by this rule knowledge gives the plots of singular values of the complementary sensitivity function as in Figure 7.14.

The closed-loop time domain rule knowledge base is triggered to satisfy the specifications of the closed-loop reference tracking of unit step inputs. After several iterations, the closed-loop time domain rule knowledge base is terminated with no candidates of the controller satisfying all the reference tracking specifications and the control input specifications simultaneously. The best design which satisfies all the control input specifications has closed-loop step responses depicted by Figure 7.15 - 7.18. From the plots, it can be seen that the output responses with respect to step reference signals injected to channels 1 and 3 exhibit unwanted overshoots, and moreover, the output responses
Figure 7.14: Singular values of the complementary sensitivity function (1 DOF) with respect to step reference signals injected to channel 2 have significant steady state error with large cross-coupling interactions.

Figure 7.15: Closed-loop step responses with unit step input on reference 1
Chapter 7. Design Examples

Figure 7.16: Closed-loop step responses with unit step input on reference 2

Figure 7.17: Closed-loop step responses with unit step input on reference 3

Figure 7.18: Closed-loop step responses with unit step input on reference 4
7.3.2 LSDP Two Degree of Freedom Design

The weighting functions designed by the one degree of freedom design module are used for the two degree of freedom design module. The control input tuning rule knowledge base is first fired to ensure that control efforts remain within the prescribed bounds of ±16 cm. The two degree of freedom closed-loop time domain rule knowledge base is fired to tune the controller to meet the specifications of the unit step reference tracking. After several iterations, a controller which meets all the closed-loop time domain specifications is found.

The singular value plots of the complementary sensitivity function are shown in Figure 7.19 where it can be seen that all the singular values roll off well before 10 rad/s to avoid the lightly damped structural transmission mode of the plant at 14 rad/s.

![Closed Loop COMPLEMENTARY SENSITIVITY (2 dof)](image)

Figure 7.19: Singular values of the complementary sensitivity function (BELL-205)

The closed-loop time domain responses of the controlled outputs based on unit step demands at each reference channel are illustrated in Figure 7.20 - 7.23. The closed-loop responses are reasonably fast with low cross channel and no overshoot. The plant input signals based on the same unit step demands are shown in Figure 7.24 - 7.27. All the control effort responses are within the bound of ±16 cm.
Figure 7.20: Closed-loop step responses with unit step input on reference 1

Figure 7.21: Closed-loop step responses with unit step input on reference 2

Figure 7.22: Closed-loop step responses with unit step input on reference 3
Figure 7.23: Closed-loop step responses with unit step input on reference 4

Figure 7.24: Plant input responses with unit step input on reference 1

Figure 7.25: Plant input responses with unit step input on reference 2
Figure 7.26: Plant input responses with unit step input on reference 3

Figure 7.27: Plant input responses with unit step input on reference 4
Chapter 7. Design Examples

7.4 A High Purity Distillation Column Plant

The high purity distillation column is described in [40, 66], which has a demanding mixture of time domain performance specifications and robust stability requirement. The purpose of the column is to split the input feed into its light and heavy components. The column may be operated in two possible configurations; in the first configuration, the controlled variables are the reflux and the boil-up (the \textit{LV} configuration), while in the second configuration, the controlled variables are the distillate flow rate and the boil-up (the \textit{DV} configuration). In this design example, only the \textit{LV} configuration will be considered.

The model of the \textit{LV} configuration is given by [40, 66] as:

\[
G_{LV}(s) = \frac{1}{7.5s + 1} \begin{bmatrix} 0.878 & -0.864 \\ 1.082 & -1.096 \end{bmatrix} \begin{bmatrix} k_1 e^{-5s} \\ 0 \end{bmatrix}
\]

The time delays are used to model the uncertain flow dynamics, they lie in the range of \(0 < \tau_1, \tau_2 < 1\) minute. The actuator gains are also uncertain due to the errors of the flow rate measurement, and the variables lie in the range of \(0.8 < k_1, k_2 < 1.2\).

7.4.1 Design Specification

The aim of the case study is to produce a controllers to meet the robust stability requirement and a set of time domain performance specifications for the \textit{LV} configuration. In the context of the robust stability requirement, the closed-loop systems are required to be stable for all \(0 < \tau_1, \tau_2 < 1\) minute and all \(0.8 < k_1, k_2 < 1.2\).

The time domain performance specifications are given in terms of the output responses to step demands applied to the plant models described by Equation (7.1) as follows:

1. If \(H(t)\) is the unit step applied at \(t = 0\), the output response of the plant to the demand of \(H(t)[1 \ 0]^T\) must satisfies the following constraints:

   o \(y_1(t) \leq 1.1\) for all \(t\).

   o \(y_2(t)\) should go from 0 to \(\geq 0.9\) in less than 30 minutes.

   o \(y_2(t)\) should not exceed 0.5.
Chapter 7. Design Examples

1. The step input of \( H(t) [0.4 \ 0.6]' \) is in the high gain direction of the plant, as described in [66]. In this case, the following time response requirements must be satisfied:

- \( y_1(t) \leq 0.5 \) for all \( t \)
- \( y_1(t) \) should go from 0 to \( \geq 0.35 \) in less than 30 minutes
- \( y_2(t) \leq 0.7 \) for all \( t \)
- \( y_2(t) \) should go from 0 to \( \geq 0.55 \) in less than 30 minutes
- \( 0.41 \geq y_1(\infty) \geq 0.39 \)
- \( 0.61 \geq y_2(\infty) \geq 0.59 \)

2. With the step input of \( H(t) [0 \ 1]' \), the following requirements must be satisfied:

- \( y_1(t) \leq 1.1 \) for all \( t \)
- \( y_2(t) \) should go from 0 to \( \geq 0.9 \) in less than 30 minutes
- \( y_2(t) \) should not exceed 0.5
- \( 1.01 \geq y_2(\infty) \geq 0.99 \)
- \( 0.01 \geq y_1(\infty) \geq -0.01 \)

To avoid controllers with unrealistic gains and bandwidths, Limebeer [40] suggested the following closed-loop frequency domain requirements:

1. The closed-loop transfer function between the demand input and the plant input must be less than 50 dB.
2. The unity gain cross-over frequency of the largest singular value must be less than 150 rad/min.
Chapter 7. Design Examples

<table>
<thead>
<tr>
<th>Plant</th>
<th>$k_1$</th>
<th>$k_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{nom}$</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>$G_1$</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>$G_2$</td>
<td>0.8</td>
<td>1.2</td>
</tr>
<tr>
<td>$G_3$</td>
<td>1.2</td>
<td>0.8</td>
</tr>
<tr>
<td>$G_4$</td>
<td>1.2</td>
<td>1.2</td>
</tr>
</tbody>
</table>

Table 7.1: Plant Models For Unit Step Response Simulations

To assess the robustness and the performance of the designed controller, a set unit step response simulation with the nominal and extreme plants listed in Table 7.4.1 will be used.

In addition, the extreme plant simulations will be carried out with time delays of 0.5 minute and 1 minute separately to show the satisfaction of the level of constraints as proposed in [40] by Limebeer. The same extreme plants can be used for high directional gain simulation of the step input of $H(t)[0.4 0.6]$ with time delays of 0.5 minute and 1 minute separately.

7.4.2 LSDP One Degree of Freedom Design

The open-loop rule knowledge base adds integrators to the diagonal elements of the pre-compensator $W_1$ to ensure good steady-state unit step tracking of the closed-loop system. The loop gain is shifted vertically by increasing the gains of the diagonal elements of $W_1$ so that the open-loop singular values of the shaped plant have its maximum cross-over frequency at about 2 rad/s. Figure 7.28 shows the singular values of the augmented plant shaped by the open-loop rule knowledge base.

The closed-loop frequency domain rule knowledge base is fired to check the magnitude constraint imposed on the closed-loop transfer function between the reference inputs and the plant inputs. The bound of 50 dB is found to be satisfied. Figure 7.29 shows the closed-loop transfer function between the demand input and the plant input.

The time domain rule knowledge base is triggered to compensate for the closed-loop specifications between the reference inputs and the regulated outputs. After several
Chapter 7. Design Examples

Figure 7.28: The open-loop singular values plot of the shaped plant (1 DOF)

Figure 7.29: The closed-loop transfer function between the demand input and the plant input (1 DOF)
iterations, a controller which satisfies all the reference tracking specifications is found. The controller is then used for the extreme plant simulations with time delays of 0.5 minute and 1 minute separately.

Figure 7.30 and 7.31 show the extreme plant simulations with time delays of 0.5 minute for the step inputs at channel 1 and 2 respectively. In channel 1 simulations, the rise time specification has been satisfied by all the extreme plant responses, however, the cross-channel interaction has been violated by one of the extreme plant simulations. In channel 2 simulations, two of the extreme plant responses have longer rise time than the specified value, and the cross coupling specification has been violated by two plots.

Figure 7.30: LV Time Domain Simulations (channel 1) : time delay 0.5 minute (1 DOF)

Figure 7.32 shows the high directional gain step input of $H(t)[0.4 \ 0.6]'$ with the time delay of 0.5 minute. All the specifications for these simulations have been satisfied.

Figure 7.33 and 7.34 show the extreme plant simulations with time delays of 1 minute for the step inputs at channel 1 and 2 respectively. For channel 1 simulation, the specification of the rise time and the cross-channel interaction have been violated by several extreme plants. In the channel 2 simulation, the rise time specification is marginally satisfied by all the extreme plants. Nevertheless, the cross-coupling specification is exceeded by most of the extreme plants.

Figure 7.32 shows the high directional gain step responses of the closed-loop system with the time delay of 1 minute. All the simulations satisfy the rise time specification
Figure 7.31: LV Time Domain Simulations (channel 1): time delay 0.5 minute (1 DOF)

Figure 7.32: 1 DOF Time Domain Simulations: time delay of 0.5 minute with step input of [0.4 0.6]
Figure 7.33: LV Time Domain Simulations (channel 1): time delay 1 minute (1 DOF)

Figure 7.34: LV Time Domain Simulations (channel 2): time delay 1 minute (1 DOF)
but two of them have violated the cross-coupling requirement.

Figure 7.35: 1 DOF Time Domain Simulations: time delay of 1 minute with step input
of [0.4 0.6]

7.4.3 LSDP Two Degree of Freedom Design

The augmented plant shaped by the one degree of freedom LSDP open-loop rule knowledge base is applied to the two degree of freedom LSDP design module. Based on the time domain reference tracking requirement, a first order reference model is synthesized by the two degree of freedom design module to construct the two degree of freedom design configuration for calculating the initial two degree of freedom $H^\infty$ controller. Based on the initial controller, all the closed-loop transfer functions are formed to check the satisfaction of all the related specifications.

The frequency domain specifications are found to be satisfied as in the one degree of freedom design module. The priority of the magnitude constraint imposed on the singular values of the closed-loop transfer function between the reference inputs and the plant inputs has been increased so that the subsequent firing of any rule knowledge bases will not cause the violation of this constraint. The singular values of the closed-loop transfer function from the reference inputs to the plant inputs are depicted in Figure 7.36.

The time domain rule knowledge base is triggered to tune the initial controller to satisfy the time domain reference tracking specifications. Initially, only the specification of the
Figure 7.36: The closed-loop transfer function between the demand input and the plant input (TDF)

rise time of channel 1 closed reference tracking step response is found to be unsatisfied. After several iterations, a controller which satisfies all the specifications is found. The controller is used for the extreme plant simulations with time delays of 0.5 minute and 1.0 minute separately as before.

Figures 7.37 and 7.38 show the extreme plant simulations with time delays of 0.5 minute for the step inputs at channel 1 and 2 respectively. In both simulations, all the plots were within the specified bounds, the cross channel interactions are particularly small compared to the controller designed by the one degree of freedom design module.

Figure 7.37: LV Time Domain Simulations (channel 1) : time delay 0.5 minute (2 DOF)
Figure 7.38: LV Time Domain Simulations (channel 2): time delay 0.5 minute (2 DOF)

Figure 7.39 shows the closed-loop simulations using the extreme plants with the same time delay in the high directional step inputs of $H(t)[0.4 \ 0.6]'$. All the specifications have been satisfied without showing any significant cross coupling between the two channels.

Figure 7.39: 2 DOF Time Domain Simulations: time delay of 0.5 minute with step input of $[0.4 \ 0.6]'$

Figure 7.40 and 7.41 show the extreme plant simulations with time delays of 1 minute for both step inputs at channel 1 and 2 respectively. The effects of the cross coupling interactions become more significant as expected. Nevertheless, all the specifications are satisfied in both sets of simulation.

Figure 7.39 shows the closed-loop simulations using the extreme plants with the same
Figure 7.40: LV Time Domain Simulations (channel 1): time delay 1 minute (2 DOF)

Figure 7.41: LV Time Domain Simulations (channel 2): time delay 1 minute (2 DOF)
time delay in the high directional step inputs of $H(t)[0.4\ 0.6]'$. All the specifications have been satisfied with two of the plots showing minor oscillatory behaviour.

![Graph](image)

Figure 7.42: 2 DOF Time Domain Simulations: time delay of 1 minute with step input of $[0.4\ 0.6]'$

### 7.5 Summary

In this chapter, three design case studies were performed using the implemented expert system. Based on the closed-loop time domain and frequency domain specifications, the expert system had proved to be useful in designing robust controllers for the multivariable plants. The systematic design procedures embedded within the rule knowledge bases of the expert system were found to be sufficient in treating the problems of control system design. The two degree of freedom loop shaping design rule knowledge base had shown significant improvements on the time-domain reference tracking specifications over the one degree of freedom rule knowledge base.
CONCLUSIONS AND FUTURE WORK

8.1 Concluding Remarks

This thesis demonstrates the applicability of expert system techniques to control engineering problems. An integrated expert system environment for robust control design was developed. $H^\infty$ loop-shaping techniques were examined and employed by rule knowledge bases in the implemented expert system to design robust controllers systematically. In the process of developing these rule knowledge bases, a number of multivariable loop shaping design issues have been brought to light. Answers to these issues have resulted in an overall design methodology which can be applied to the generic problems of control system design. Various implementation issues in the development of an expert system for robust controller design have been raised, and efforts have been made to overcome the difficulties encountered in constructing the expert system. The main conclusions and contributions of this thesis can be summarized as follows:

1. The $H^\infty$ optimization approach for robust controller design has been investigated. The concept of normalized coprime factor perturbations for modelling plant uncertainty and its related problem of robust stabilizing controller design were studied. A one degree of freedom Loop Shaping Design Procedure (LSDP) and its two degree of freedom extension were examined to extract multivariable controller design heuristics. These $H^\infty$ loop-shaping approaches were chosen as the controller design methods to be used by the expert system due to their systematic and intuitive appeal. They are based on the shaping of open-loop singular values and for mul-
multivariable systems are a natural generalization of classical SISO frequency domain
open-loop gain-shaping. As such, they are easier to use by engineers possessing
a classical background in control engineering. Furthermore, multivariable open-
loop singular value loop-shaping heuristics can be formulated systematically and
captured efficiently by the rule knowledge base of the expert system.

2. Although the LSDP design technique discussed in Chapter 3 offered a systematic
and powerful robust controller design approach, effective application of the tech­
nique relies on the experience of the practitioner. Chapter 4 raises a number of
issues relating to the development of an expert system which employs the LSDP
design method. It is suggested that a thorough knowledge of the heuristics of
the central design methodology is insufficient. The more fundamental issues of
formulating and choosing the appropriate design objectives generic to any con­
trol system design need to be addressed. After careful consideration, a list of
essential constraints and design objectives has been identified as constituting the
generic design specification. This list attempts to encompass all the time and fre­
quency domain characteristics. The organization of these design constraints into a
framework is essential if the data is to be processed efficiently. Before implement­
ing these constraints within the expert system, the heuristics of robust controller
design were examined. In general, since most of the design requirements are spec­
ified in the form of closed-loop time/frequency domain specifications, the main
difficulty in successfully applying the $H^\infty$ loop-shaping techniques is to interpret
the closed-loop time/frequency domain specifications into the required shape of
the open-loop singular values. To circumvent this obstacle, the relationships be­
tween the closed-loop design objectives and the required shape of the open-loop
singular values were investigated in Section 4.3. Systematic procedures for per­
forming robust controller design using one degree of freedom and two degree of
freedom LSDP methods were presented in Section 4.4 and 4.5 respectively. Using
these systematic guidelines, the one degree of freedom and two degree of freedom
LSDP rule knowledge bases were developed in Section 4.6 and 4.7 respectively.
In addition, a systematic loop-shaping design methodology which integrates the
one degree of freedom Loop Shaping Design Procedure and its two degree of freedom extension was proposed for treating the generic robust control problem. A flow-chart illustrating the design process for this overall systematic LSDP design methodology was depicted in Figure 4.12. The main contribution of this systematic loop-shaping design methodology is to provide an explicit insight to link all the rule knowledge bases of the expert system accordingly, to draw a clear picture of the overall design flow in using the loop shaping design rule knowledge bases which in turn alleviate the task of designing the rule firing control mechanisms.

3. The implementation process of the expert system was described in Chapter 5. In particular, a formal method was presented for organizing the data related to control system design using a structural and object-oriented framework, i.e. the Frame System. Within this framework, control system data are collected in an inverted tree with the top node being a generic object attached to a series of specialized objects accordingly. At the bottom of the structure, general design specifications in time and frequency domain are found. They are the basis of all the specification classes from which instances of various design specifications can be created and linked to the entire structure (refer to Figure 5.3). Frame system's data processing functions and procedures which managed the dynamic behaviour of the Frame System were presented in Section 5.6.2. The overall organization of the rule knowledge bases and the rule inference mechanisms were discussed in Section 5.7 with emphasis placed on the modularity of the individual rule knowledge base and the dependency of the design modules on each other. The generic rule-firing mechanisms of the rule knowledge bases were studied in depth. In particular, methods of testing for improvement of a design step, and actions to be taken after a non-improving design step, were presented. Consequently, a generic rule firing control mechanism was devised in Section 5.7.3 to be used by each rule knowledge base. The internal structure of the implemented expert system was presented to show the novelty of the software package. Attention was paid to the motivation and the philosophy behind constructing the components of the expert system, so that the entire implementation phase was well justified.
and properly understood. An external program interface was implemented using the standard C-language to provide data communications between the various software packages, and for data loading/storing purposes (refer to Figure 5.8).

4. Several numerical optimization techniques were considered to form mixed-optimization approaches for a robust controller design. The Method of Inequalities coupled with the Moving Boundary Process was found to be very helpful in formulating and solving the control system design problems by using a set of explicit inequalities which can be the functionals of the closed-loop time domain and/or frequency domain specifications. A Multi-Objective Genetic Algorithm (MOGA) was examined, and various genetic-based chromosomal operations used by the MOGA were discussed. The Multi-Layer Multi-Objective Genetic Algorithm (MLMOGA) was developed as an extension of the Multi-Objective Genetic Algorithm. The MLMOGA was then combined with the $H^\infty$ loop-shaping controller design approaches to devise the mixed optimization techniques for solving the robust control problems. An IFAC benchmark problem was used to illustrate the applicability of these parallel and numerical controller design techniques to tackle the robust performance problem, and reasonably good results were reported. It was observed that the MLMOGA was useful in situations where both the order of the weighting functions and their parameters were required to be searched.

5. Three design case studies have been described to illustrate the applications of the implemented expert system for various robust controller designs. For each case study, the design details of the controller were worked through from the one degree of freedom design module to the two degree of freedom design module. Based on the time domain closed-loop performance specification, the expert system was proven to be capable of designing the robust controller with the knowledge embedded within the rule knowledge bases. By employing the expert system, the robust controller design process was transformed to a more straightforward task. In particular, the user is relieved from the laborious efforts of remembering the formal syntax of the functions involved in the control system design software package, and
8.2 Suggestions for Future Research

The work presented in this thesis points to a number of areas where future research can be conducted. These are discussed as follows:

1. The developed expert system provides an open structure for the inclusion of additional rule knowledge bases to capture other controller design techniques which are related to the $H^\infty$ optimization approach. In particular, the $H^\infty$ mixed sensitivity optimization methods can be integrated into the expert system as an alternative approach for the robust controller design. Furthermore, appropriate non-linear controller design techniques could be incorporated into the expert system to handle non-linear control problems.

2. The design of a controller is only one step in the entire control system design process. Other issues such as the identification of the plant, simulation of the designed closed-loop system, plant and controller size reduction and the implementation issues of controller are yet to be addressed in the design cycle. Experience gained in using the expert system should be very helpful in further developing the system to assist users in handling other tasks in the control system design process, drawing on a more extensive knowledge of control engineering.

3. The concerns of the loop-shaping design approach and the strategies developed point to the need for a more comprehensive graphical interface. The user interface of the expert system leaves plenty of room for improvement. The current graphical user interface can be further developed with an intelligent graphical front end which can adapt itself to different kinds of users (from novice to expert), providing appropriate assistance and guidance. Pang et al. [57] has demonstrated a strong case for using an Intelligent Front End (IFE) to support the $H^\infty$ optimization control design method.
4. An extensive on-line help system can be incorporated into the expert system to provide a more comprehensive help system. In particular, the mouse-based and word sensitive (hypertext) on-line guidance system appears to be attractive in providing useful information to the user, for example, the user should be able to click on some pre-defined keywords to reveal more information about certain topics which are of interest or are even crucial to understand before proceeding further in the design process.

5. The frame knowledge base can be integrated with a design database to store conventional control data and design history which are invaluable to the designer [44]. The co-ordination of data manipulation and the explanatory system will be greatly enhanced by the expressive power available in database management systems. Much research has been carried out to apply the concept of an intelligent database system to create hybrid environments for CACSD [30, 1, 3], which in turn can be learned and used for incorporating an intelligent database into the expert system described by this thesis. Furthermore, another potential contribution of the database management facility is the introduction of machine learning capabilities into the expert system. The expert system can be programmed in such a way that, certain good design characteristics can be picked up by the expert system based on past design knowledge and stored in the database to be used for future designs.

6. Parallel optimization approaches such as Genetic Algorithms have proven to be useful in conducting the numerical search processes as suggested in Chapter 6. Currently, Genetic Algorithms are loosely incorporated into the expert system in such a way that they can be invoked from the MATLAB command line within the user interface. Further research can be carried out to structure the optimization and search processes. Rule knowledge bases can be developed to monitor the search process and perform the necessary trade-offs among the design objectives. In addition, other probabilistic-based optimization techniques such as the Simulated Annealing algorithm can be explored for multi-objective control system
design. The Simulated Annealing algorithm has its origins in the statistical mechanical annealing of solids. A simple algorithm was proposed by Metropolis et al. [48] for simulating the annealing process which can be used to solve optimization problems [69, 74]. Initial research has been conducted in the Control Systems Research Group at Leicester University to explore the use of Simulated Annealing approach in the controller design [75]. The major difficulty in applying the Simulated Annealing approach is in choosing a suitable initial temperature and estimating the effects of the other parameters, such as the cooling rate. Additional work is required to investigate these problems in order that simulated annealing might be successfully applied to optimization problems in controller design.
References


Chapter 8. Conclusion and Future Work


Chapter 8. Conclusion and Future Work


Chapter 8. Conclusion and Future Work


Chapter 8. Conclusion and Future Work


Chapter 8. Conclusion and Future Work


Chapter 8. Conclusion and Future Work


Chapter 8. Conclusion and Future Work


