4

CONTROL SYSTEM DESIGN USING ARTIFICIAL INTELLIGENCE

Colin Dean Tebbutt

Submitted to the department of Electrical and Electronic Engineering, University of Cape Town, in fulfilment of the requirements for the degree of Doctor of Philosophy.

November 1990

CONTROL SYSTEM DESIGN USING ARTIFICIAL INTELLIGENCE

Colin Tebbutt

ABSTRACT.

Successful multivariable control system design demands knowledge, skill and creativity of the designer. The goal of the research described in this dissertation was to investigate, implement, and evaluate methods by which artificial intelligence techniques, in a broad sense, may be used in a design system to assist the user. An intelligent, interactive, control system design tool has been developed to fulfil this aim.

The design tool comprises two main components; an expert system on the upper level, and a powerful CACSD package on the lower level. The expert system has been constructed to assist and guide the designer in using the facilities provided by the underlying CACSD package. Unlike other expert systems, the user is also aided in formulating and refining a comprehensive and achievable design specification, and in dealing with conflicts which may arise within this specification. The assistance is aimed at both novice and experienced designers.

The CACSD package includes a synthesis program which attempts to find a controller that satisfies the design specification. The synthesis program is based upon a recent factorization theory approach, where the linear multivariable control system design problem is translated

into, and solved as, a quadratic programming problem. Techniques which significantly improve the time and space efficiency of this method have been developed, making it practical to solve substantial multivariable design problems using only a microcomputer.

The design system has been used by students at the University of Cape Town. Designs produced using the expert system tool are compared against those produced using classical design methods.

KEYWORDS.

Expert Systems, Computer Aided Control System Design,
Linear Systems, Multivariable Control Systems,
Matrix Fractions, Quadratic Programming.

ACKNOWLEDGEMENTS.

I would like to express my thanks to Professor Martin Braae for supervising this research.

I would also like to thank AECI (Pty) Ltd., and to the Foundation for Research Development, for generous financial support.

Finally, thanks to Anne Sargent of the University of Detroit library for tracking down numerous conference papers.

"Whatever you do,
work at it with all your heart,
as working for the Lord."

Colossians 3:23

TABLE OF CONTENTS.

SYMBOLS AND ABBREVIATIONS.

AI	Artificial Intelligence.
C	The set of complex numbers.
C^n	The set of complex vectors of dimension n.
C_-	A subset of the complex plane defined as the region of stability; the set $\{z: \|z\| < 1\}$ is often chosen.
C_{+e}	the complement of C_-, including the point at infinity.
CAD	Computer Aided Design.
CACSD	Computer Aided Control System Design.
DNA	Direct Nyquist Array.
F	The field of fractions associated with S; this is the set of real rational transfer functions.
FIR	Finite Impulse Response.
I	The unit or identity matrix.
INA	Inverse Nyquist Array.
LQG	Linear Quadratic Gaussian.
LTR	Loop Transfer Recovery.

M(S) The set of matrices with elements in S, i.e. the set of matrices where the elements are proper stable transfer functions.

M(F) The set of matrices with elements in F, i.e. the set of matrices where the elements are transfer functions.

MIMO Multi-Input Multi-Output.

R The set of real numbers.

$\mathbf{R}^n$ The set of real vectors of dimension n.

R[z] The set of polynomials in the indeterminate z, with real coefficients.

R(z) The field of fractions associated with R[z]. This is the set of real rational transfer functions.

RAM Random Access Memory.

S The subset of R(z) comprising all rational functions analytic on C_{+e}. This is the set of proper stable transfer functions.

SISO Single-Input Single-Output.

SVD Singular Value Decomposition.

U The set of units in S, i.e. the set of proper stable transfer functions whose inverses are also proper stable transfer functions.

U(S) The set of unimodular matrices in **M(S)**, i.e. those matrices in **M(S)** whose inverses are also members of **M(S)**.

TRADEMARKS.

dmX is a trademark of Decision Management Software.

IBM-PC is a trademark of International Business Machines Corporation.

Personal Consultant is a trademark of Texas Instruments, Inc.

MS-DOS is a trademark of Microsoft Corporation.

Synapse is a trademark of Hitep.

TURBO-C is a trademark of Borland International.

VP-Expert is a trademark of Paperback Software International.

CHAPTER ONE. INTRODUCTION.

1.1. INTRODUCTION.

The study of artificial intelligence has enjoyed much attention over the past few decades, and the technology has been applied in a wide variety of fields. In control system engineering, artificial intelligence techniques have been employed in two main categories: the online and offline applications. Real-time expert systems [A4] and neural networks [A2], for example, have been used to implement physical controllers or supervisory control systems. Applications in the second category include system identification, for example [H1] and [L1], and control system design.

Despite all the attention received, artificial intelligence as a subject still lacks an adequate definition. One popular definition, as expressed by Graham [G4], is :

> *"Artificial Intelligence is the branch of computer science devoted to programming computers to carry out tasks that if carried out by human beings would require intelligence."*

This definition is hardly satisfactory in the sense that the computation of the first five significant digits of $\sqrt{2}$ requires a great deal of human intelligence, while a simple hand-held calculator, performing this task with breathtaking speed, is rarely considered intelligent. Conversely, it is a simple matter for a human to identify a friend in a crowd, but the same task is formidable even for present day intelligent machines. Clearly humans and computers have

differing 'natural' abilities and skills.

Trying to account for this, Rich [R1] proposes an alternative definition :

> *"Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better."*

The design of a control system certainly requires intelligence in a human, and is generally considered to fall within the scope of artificial intelligence, fitting both definitions. Engineering design is a combination of (human) creativity and intelligent decision making [D1]. However the aim of any intelligent design system is to produce a good design, and not necessarily to replace the designer. Dreyfus and Dreyfus [D3,D4] argue persuasively against relying on the skills of the computer alone. The designer and intelligent design system should rather be viewed as a unit or team, where each member contributes specific skills while working towards a common goal.

Pang and MacFarlane [P1] list some of the relative strengths and weaknesses of the human and the computer. Humans, for example, have powerful abstraction and pattern-recognition faculties, while computers can perform complex calculations with speed, accuracy and reliability. A combination of human and machine could go a long way towards eliminating the weaknesses of either. Finding this combination should be one of the goals of artificial intelligence research.

The intelligent design system described here, named MV-CXS, is based on this teamwork principle. Its purpose is to assist the user in formulating a comprehensive design specification, to deal with possible conflicts in the design constraints, and to find a controller which meets these specifications. The designer is ultimately responsible for the engineering decisions.

Despite the power of present-day computers, excessive demands on system resources, such as processing time or memory, may render a certain task more suitable for human solution. For example, a specific algorithm may enable a computer to decipher a hand-written letter, but with less speed and/or lower accuracy than a human. In this instance, improving the efficiency of the algorithm dramatically would qualify as artificial intelligence work using Rich's [R1] definition. In many ways intelligence may be related to speed of response; a person who correctly answers a question quickly is often considered more intelligent than another who takes longer to answer. Similarly, the 'intelligence' of two equivalent algorithms could be related by comparing their efficiencies.

A sizeable portion of the work reported here relates to improving the time and space efficiency of a quadratic programming algorithm. This algorithm attempts to find the vector $\mathbf{x}$ which minimizes the quadratic cost function

$$J(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c,$$

while satisfying many (typically hundreds of) linear constraints in the form

$$\mathbf{d}^T\mathbf{x} \geq e.$$

Although this task would certainly require intelligence if performed by a human, the algorithm does not fit Rich's [R1] definition, and is not commonly classed as an artificial intelligence technique. Nevertheless its efficiency is vital to usefulness or intelligence of the overall design system, in that for a given computer system it affects the rate at which problems are solved, and determines the upper limit on the size of problems which may be addressed. Similar considerations apply to the 'intelligent' (efficient) search strategies, such as the alpha-beta method [R1] frequently used in games programs, which have been pervasive in AI

research. In both cases knowledge of the specific problem is used to reduce the search space and storage requirements, and accelerate its solution.

1.2. WHY USE AN EXPERT SYSTEM?

There have been two distinct trends in the evolution of control engineering software. Firstly, computer aided design systems have grown from mainly single-purpose programs for analysis and design into comprehensive packages covering a wide range of control engineering activities [T1]. Secondly, the design systems are being aimed at an increasingly wide range of users, and not only at experienced designers [A3]. Progress, particularly in the latter direction, has often been based upon expert system technology. While existing computer aided control system design (CACSD) tools are almost exclusively analysis packages [P3], the use of expert system techniques have offered the hope of producing fully-fledged design packages, which are able to provide meaningful guidance for the user during the design process.

Expert system techniques deal effectively with the problem of complexity management. In this application, as with many artificial intelligence problems, there is a complex decision making structure and a large amount of knowledge. Expert systems offer a powerful facility for representing decision making knowledge in terms of rules and facts. According to Taylor and Frederick [T1],

> *"...a rule-based expert system can be endowed with greater flexibility than conventional software, because of its knowledge base, inference capability, and more natural interaction with the user".*

There is no such thing as an instant human expert; in practice every expert goes through a learning phase.

Similarly it can be expected that an intelligent design system will go through a similar development cycle. One of the attractive features of the expert system methodology is that, as experience with the system grows, the knowledge base can be extended with relative ease, and this may usually be done without disturbing the structure of the knowledge already present. Again, Taylor and Frederick [T1] comment

> *"...an expert system is easier to expand than a conventional program, in the sense that the mechanics of adding rules that embody 'new expertise' is straightforward".*

Expert system shells usually offer good online debugging aids; for example one may trace the reasoning process to determine how the value was inferred for a particular variable. Effective debugging facilities are a vital component of any complex software project.

1.3. PREVIOUS APPLICATIONS OF EXPERT SYSTEMS IN CACSD.

Taylor and Frederick [T1] (1984), and James *et al.* [J2] (1986), present an overview of the application of expert systems to control engineering, in particular outlining the wide range of design activities which should be addressed. They propose an architecture where the expert system coordinates and integrates many analysis and design procedures.

James *et al.* [J3] (1987), describe an expert system implementation of an algorithm for single variable lead-lag compensator design. This in turn forms a small part of Taylor and Frederick's [T1] expert system mentioned above, and interfaces to subroutines in the Cambridge Linear Analysis and Design Program CLADP [E1]. After specifying the performance required, the designer has little further

involvement in the design process.

Trankle, Sheu and Rabin [T9] (1986) describe a two-level expert planning system, based upon the CTRL-C package [L3]. A high level planner produces a skeleton plan, and a list of performance specifications for the low level planner to satisfy. The low level planner in turn creates a list of commands for the CTRL-C package. Side effects, resulting from the tradeoffs inherent in controller design, complicate the planning. Again, the user has minimal involvement in the design process.

Nolan [N2] (1986) developed an expert system which deduces the feasibility of various single variable feedback configurations, as well as an appropriate synthesis method to be used, based upon the plant type number and order. The system uses algebraic manipulation of the plant transfer function and controller structure, and once a synthesis method is selected, guides the designer in using an appropriate conventional CAD package.

Birdwell *et al.* [B3] (1985) discuss an expert system interface to a multivariable LQG/LTR design package. The expert system, named CASCADE, was found useful in automating some of the design details, thus allowing the user to concentrate on the design process and significantly reducing the time required to complete a trial design [B4].

The MAID expert system described by Pang *et al.* [P1,P2] (1987), and Boyle *et al.* [B8] (1989), uses three design techniques to address the design problem. These are their 'simple design technique', a 'reverse frame alignment technique', and an observer-based controller design technique, in order of increasing complexity. Data for design analysis is obtained from the characteristic gains and phases and principal gains of the system. The design is attempted using the simpler methods first, and if not satisfactory, the more complex methods are tried. The

authors emphasize interactive design, and recommend that engineering judgement decisions be left to the designer. This design system has recently been extended to include a stable factorization design method not unlike that used here (Pang *et al.* [P3], 1990).

1.4. DESCRIPTION OF THE DESIGN SYSTEM.

Each of the expert system based design systems mentioned above attempts to find a controller which satisfies some design specification. The design system described here takes this approach one step further by also assisting the designer in developing the specification. As stressed by MacFarlane *et al.* [M2], design is an exploratory and experimental process during which the specification is systematically refined. Therefore it is appropriate that an intelligent design tool should address the evolution of the specification during this process.

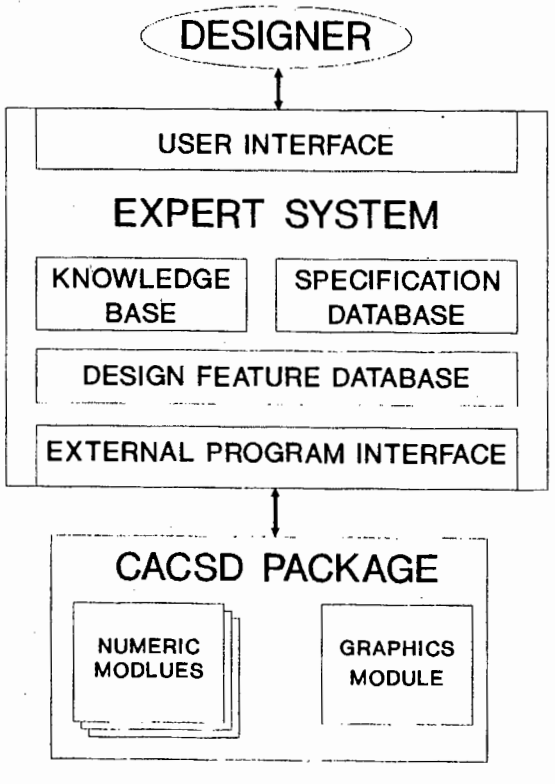


Figure 1.1. Structure of the design system.

The overall structure of the MV-CXS design system illustrated in figure 1.1 shows an expert system interposed between the designer and a CACSD package; an essentially similar structure is found in each of the applications mentioned above. In broad functional terms, the expert system helps the designer to formulate the design specification, and the CACSD package attempts to synthesize a controller which meets those specifications. Some commands from the designer, for example those for plotting the control system responses, pass directly through the expert system to the CACSD package (figure 1.2(a)), in line with the "command spy" concept of Larsson and Persson [L1], and functionally equivalent to the structure used by Pang and MacFarlane [P1] (figure 1.2(b)). Other commands, for example the SOLVE command, are translated into a sequence of calls to the CACSD package. Still others, for example the HELP commands, do not (at least directly) result in any calls to the CACSD package.

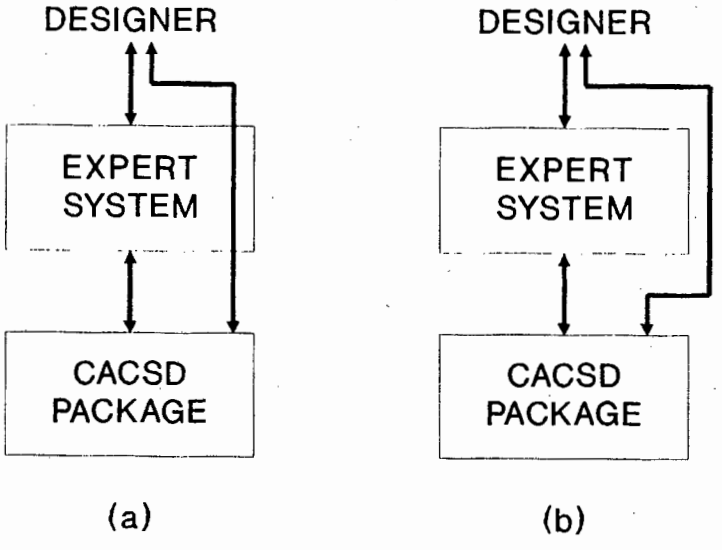


Figure 1.2. Possible relationships between the designer, expert system and CACSD package.

The CACSD package is based upon the design method of Boyd [B6], which is in turn similar to that of Fegley [F1], and in theory to that of Gustafson and Desoer [G5,G6]. It is applicable to linear time-invariant multivariable sampled

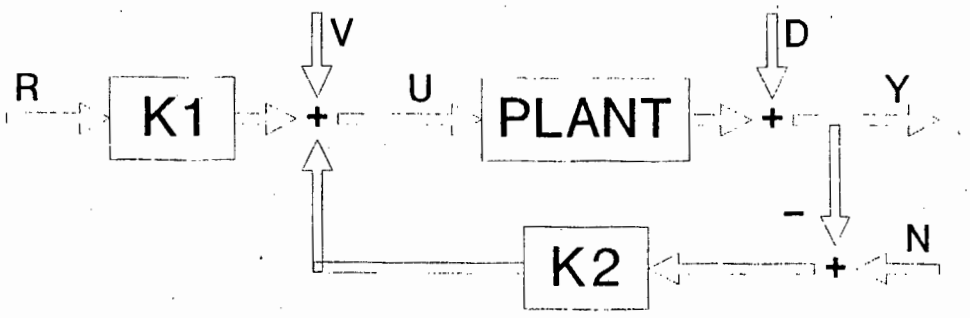


Figure 1.3. Two-parameter controller structure.

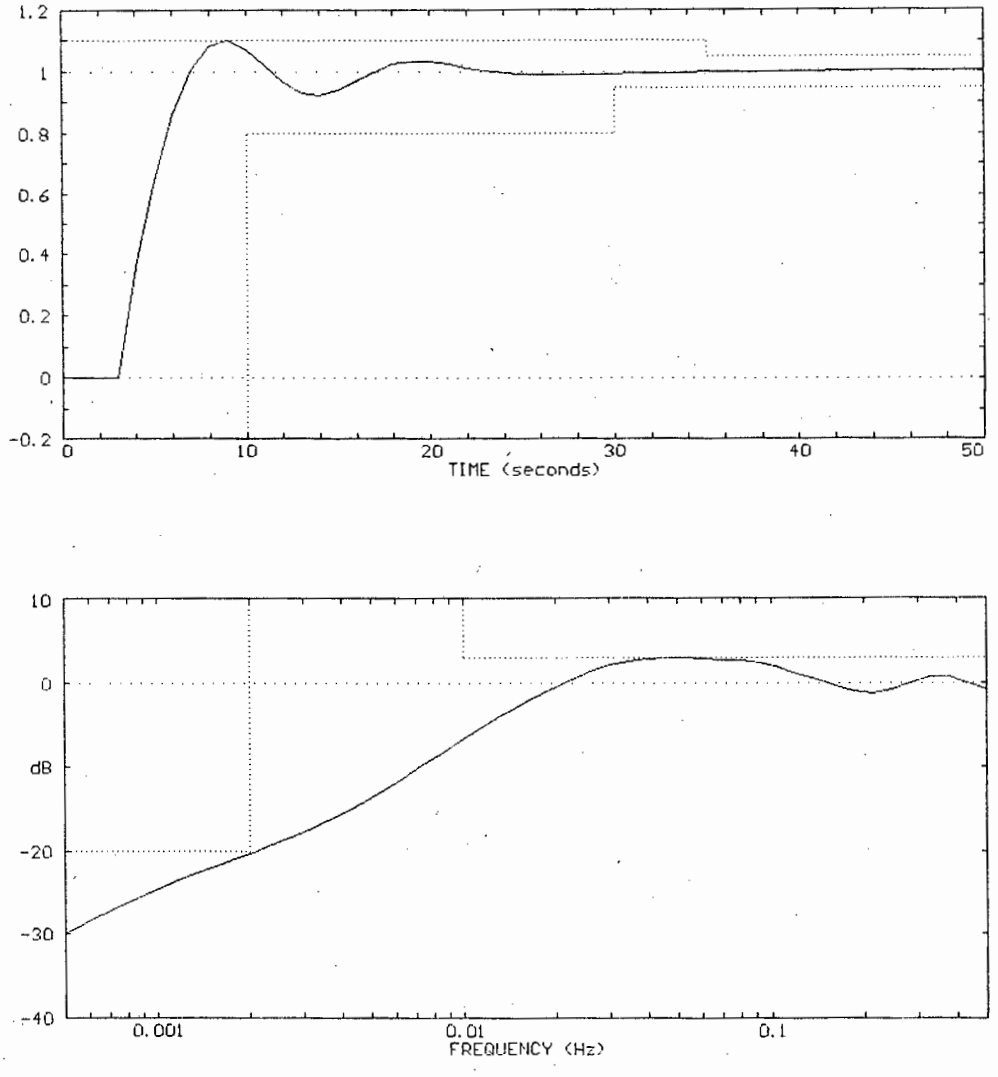


Figure 1.4. Typical performance constraints.

data systems using the two-parameter controller structure shown in figure 1.3. The plant must be described by a strictly proper z-domain transfer function, and should have a diagonal, left-coprime factorization as described in chapter 2. Design specifications may include performance constraints on the closed loop step or frequency responses to inputs at nodes R, N, or D, as well as a quadratic cost function based on these responses. Further specifications, such as constraints on the singular values of various responses, are treated implicitly by the expert system. Typical performance constraints on the closed loop step and frequency responses are illustrated in figure 1.4.

The CACSD design method transforms the control system design problem into a linearly constrained quadratic programming problem, which is then solved using a standard algorithm. The solution, if any exists, is then transformed into a controller transfer function using an explicit formula. Figure 1.5 illustrates the important steps in this design process. Unlike most other design methods for multivariable systems, the complexity of the design procedure, as seen by the designer, does not increase dramatically as the dimension of the plant increases. The designer does have more trade-off options to consider, but a great deal of the additional complexity is absorbed by the numerical software. Many conceptually similar constrained optimization methods for control system design have been proposed elsewhere, for example [B1,P4,Z1]. The advantage of Boyd's method [B6], however, is that the quadratic programming algorithm always finds the global optimum solution where one exists; the absence of any feasible solution is also determined in a finite number of iterations. This advantage is a result of convexity of the problem; the optimization function, being quadratic, has no local minima which are not also the global minimum, and only performance constraints which can be translated into a convex set of linear constraints on the search vector are allowed. Linear programming, closely

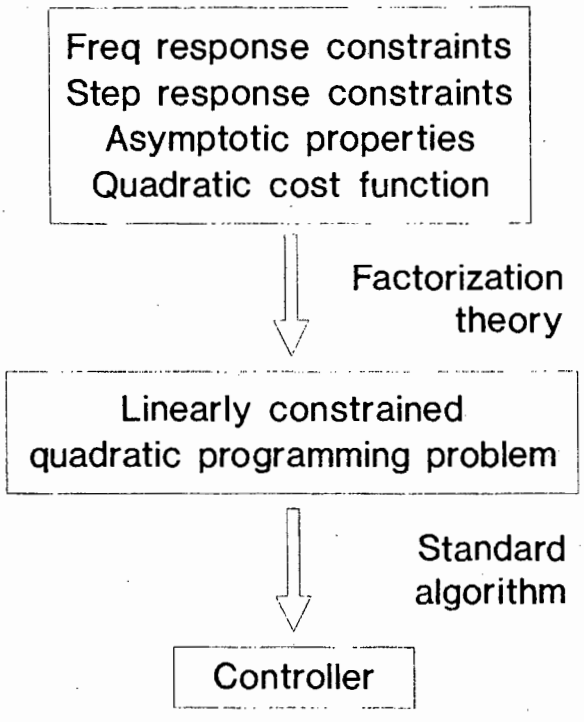


Figure 1.5. Steps in the CACSD design method.

related to quadratic programming, has also found applications in control system design, and forms the basis of Fegley's work [F1]. More recently it has also been used by Bhattacharyya *et al.* [B2,M5].

1.5. STATEMENT OF CONTRIBUTION.

The aim of the research described here has been to investigate the contribution which artificial intelligence may make to control system design. More specifically, the investigation strove to develop strategies which could both improve the efficiency of experienced designers, and assist and guide novice designers, and to implement, demonstrate and evaluate these techniques.

According to Pang *et al.* [P3], while existing CACSD tools allow the designer to analyse a control system with a particular controller, in general they do not provide the user with any guidance on how the design could be improved.

Providing this type of guidance has been an explicit goal for this research.

Much of the work has already been reported elsewhere [T2,T3,T7]. In [T4], [T5], and [T6], the implementation and testing of an early version of the design system for SISO controller design was discussed; this version constituted a feasibility study for the full multivariable design system discussed here.

The efficiency of the CACSD design method has been improved substantially relative to that of the standard algorithms, to the extent that implementation on a low cost personal computer is feasible and practical for medium sized multivariable problems. To achieve this, a compact and computationally efficient representation for storing the linear constraints generated by the design method has been developed [T2]. In addition, a method for decomposing a large class of multivariable systems into smaller independent sub-problems, and a novel parameterization useful for approximating the set of stable transfer functions, have been presented [T3]. Chapters 2 and 3 deal with the CACSD package in detail.

The expert system aims to provide a flexible design environment, catering for both novice and experienced users. In addition to assisting the designer in using the CACSD package, and unlike previous expert systems for control system design, it also aids the designer in formulating a comprehensive and achievable specification, and in dealing with conflicting design constraints. The expert system has also been useful in effectively expanding the scope of the CACSD package. Details of the expert system are provided in chapters 4 and 5.

Examples illustrating the capabilities of the design system are given in chapter 6. The design system has also been used by students at the University of Cape Town [T5], and

chapter 7 contains details of these experiences. Chapter 8 describes methods for implementing the controllers synthesized by the system.

Almost all of the software described here has been designed and programmed by the author. The exceptions are the Householder transformation [G3], a subroutine to find the roots of a polynomial [P5], and a complex singular value decomposition subroutine [B10], where well-known algorithms were used. The quadratic programming subroutine was based on algorithms given by Scales [S1]. The software has been implemented and tested on a 4.77 MHz 8088/87 IBM-PC compatible computer with 640k bytes of RAM, using the MS-DOS operating system. The numeric software was programmed in the C programming language, using the TURBO-C compiler and associated libraries. The CXS expert system shell was written using the same compiler. Thus the software should be easily ported to other (more powerful) computers.

1.6. NOTATION

In mathematical descriptions, matrices are shown in bold type with upper case names, for example **A**; vectors are also shown in bold type, but with lower case names, for example **a**. **R** is used to denote the set of real numbers, and **C** the set of complex numbers. Some additional mathematical notation is required in chapter 2, and is introduced there.

The plant is assumed to have n outputs and n inputs, and is represented by the transfer function matrix **G**(z). Thus all other transfer function matrices will have the same dimensions. The theory underlying this design system extends readily to plants which are not square. In many cases, the dependence of transfer functions on the variable z is not shown explicitly; for example the plant is often represented simply as **G**.

During its interaction with the designer, the expert system and CACSD package refer to the various closed loop responses of figure 1b using a two letter notation; for example, DY indicates the closed loop response to stimuli at input D, as observed at output Y. Individual elements in a response matrix are identified using the notation [i,j]; thus RU[2,3] denotes the response at output U2 to stimuli at input R3. In addition, plant transfer function matrix is denoted by **G**, those of the controller by **K1** and **K2**, and the open loop response (i.e. **G.K2**) by **GK**. A similar notation is used here. The closed loop transfer functions are referred to as $\mathbf{H}_c$, where c indicates the input and output nodes. For example, $\mathbf{H}_{RY}$ is the closed loop transfer function from R to Y. The corresponding time domain response at the output node to a unit step at the input node is indicated by $\mathbf{h}_c(kT)$, where T is the sampling time.

Frequency domain responses are evaluated at a number of points logarithmically spaced on the unit circle in the complex plane. Graphical plots of these responses are done using linear interpolation between successive frequency points. Similarly, plots of time domain responses also use linear interpolation between successive sampling instants. In some instances the plots of the multivariable responses are superimposed.

CHAPTER TWO. THE CACSD METHOD.

2.1. INTRODUCTION.

A powerful new CAD method for the design of linear control systems has been introduced recently [B6]. It is based on translating the control system design problem into an approximately equivalent linearly constrained quadratic programming problem, finding the solution to this using a standard algorithm, and then translating the solution back to give the corresponding controller. The design may be specified directly in terms of constraints on the closed loop time and frequency domain responses, and a quadratic cost function to be minimized.

This design algorithm turns out to be computationally demanding in terms of both memory size and processing speed, especially when used for multivariable systems. However there are a number of techniques which, while retaining most of the strengths of the original algorithm, ease the computational burden substantially, and make implementation on a low cost personal computer practical.

Much of the effort of this chapter is directed at reducing the dimension of the parameter vector used in the quadratic programming algorithm. This dimension impacts on the memory needed to store the linear constraints, and on the internal storage requirements of the quadratic programming algorithm, effectively limiting the size of problems which may be addressed. It will be shown that under certain conditions the multivariable design problem may be divided into a number of smaller independent sub-problems, which greatly reduces the parameter vector dimension. Further

computational advantage may be obtained using the efficient representation for the linear constraints developed by the author [T2]. The task of obtaining stable coprime factorizations of the plant transfer function matrix is also addressed; for the case where the plant is stabilized using a stable controller, explicit formulae for these factorizations have been developed. While some of these techniques are not applicable in every design situation, the range of designs which may be tackled remains large.

2.2. NOTATION.

The notation used below follows that of Vidyasagar [V1] closely. Let R[z] denote the set of polynomials in the indeterminate z, with real coefficients, and R(z) the field of fractions associated with R[z]. Define a subset of the complex plane C_- as a region of stability; the set $\{z: |z| < 1\}$ is often chosen, and will be assumed for the examples given later. Let C_{+e} denote the complement of this region, including the point at infinity. Let S denote the subset of R(z) comprising all rational functions analytic on C_{+e}, i.e. the set of proper stable transfer functions. S is then a commutative ring with identity, and is a domain. Let F be the field of fractions associated with S, which is also R(z).

Let M(S) denote the set of matrices with elements in S, and M(F) the set of matrices with elements in F. Let U denote the set of units in S, and U(S) the set of unimodular matrices in M(S).

In transfer function terminology, M(F) is the set of matrices with transfer functions as elements, and M(S) is the set of matrices where the elements are proper stable transfer functions. U is the set of proper stable transfer functions whose inverses are also proper stable transfer functions, and U(S) comprises those matrices in M(S) whose

inverses are also members of **M(S)**.

This application assumes that the plant to be controlled is a linear time-invariant sampled-data process which is described by a strictly proper z domain matrix transfer function $G(z) \in \mathbf{M(F)}$, with n inputs and n outputs. The results presented in this chapter apply equally to non-square plants, and those which are proper (but not necessarily strictly proper).

The closed loop transfer functions are refered to as H_c, where c indicates the input and output nodes shown in figure 2.1. For example, H_{RY} is the closed loop transfer function from R to Y. The corresponding time domain response at the output node to a unit step at the input node is indicated by $h_c(kT)$, where T is the sample time.

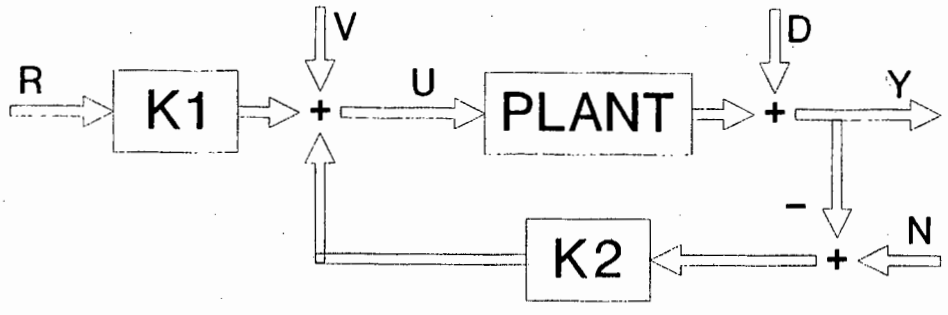


Figure 2.1. Two-parameter control system configuration.

2.3. SUMMARY OF FACTORIZATION THEORY.

A brief summary of the factorization approach to the design of linear control systems is given below; for a thorough treatment of the subject see Vidyasagar [V1]. Matrix fraction descriptions were proposed by Rosenbrock [R2], and used by Youla *et al.* [Y1] to parameterize the family of stabilizing controllers. This application is based upon the two-degree-of-freedom multivariable control system structure, shown in block diagram form in figure 2.1.

Given stable right- and left-coprime factorizations $\mathbf{N}$, $\mathbf{D} \in \mathbf{M(S)}$ and $\tilde{\mathbf{N}}$, $\tilde{\mathbf{D}} \in \mathbf{M(S)}$ of the plant

$$\mathbf{G} = \mathbf{N}.\mathbf{D}^{-1} = \tilde{\mathbf{D}}^{-1}.\tilde{\mathbf{N}} \qquad (2.3.1)$$

there exist matrices $\mathbf{X}$, $\mathbf{Y} \in \mathbf{M(S)}$ which satisfy the Bezout Identity

$$\mathbf{X}.\mathbf{N} + \mathbf{Y}.\mathbf{D} = \mathbf{I}. \qquad (2.3.2)$$

Then all internally stable closed loop transfer functions $\mathbf{H}_C \in \mathbf{M(S)}$ may be parameterized in terms of some $\mathbf{Q} \in \mathbf{M(S)}$ as

$$\mathbf{H}_C = \mathbf{H0}_C + \mathbf{H1}_C.\mathbf{Q}.\mathbf{H2}_C. \qquad (2.3.3)$$

The transfer function matrices $\mathbf{H0}_C$, $\mathbf{H1}_C$ and $\mathbf{H2}_C \in \mathbf{M(S)}$ are defined in terms of $\mathbf{N}$, $\mathbf{D}$, $\tilde{\mathbf{N}}$, $\tilde{\mathbf{D}}$, $\mathbf{X}$, $\mathbf{Y} \in \mathbf{M(S)}$; table 2.1 lists the formulae for each of the closed loop transfer functions. $\mathbf{Q}$ is one of two independent parameter matrices $\mathbf{Q1}$ and $\mathbf{Q2}$; these are chosen by the designer to give the required closed loop performance.

$\mathbf{H}_C$	$= \mathbf{H0}_C + \mathbf{H1}_C.\mathbf{Q}.\mathbf{H2}_C$
$\mathbf{H}_{RY}$	$= \mathbf{N}.\mathbf{Q1}$
$\mathbf{H}_{RU}$	$= \mathbf{D}.\mathbf{Q1}$
$\mathbf{H}_{NY}$	$= \mathbf{N}.\mathbf{X} + \mathbf{N}.\mathbf{Q2}.\tilde{\mathbf{D}}$
$\mathbf{H}_{NU}$	$= \mathbf{D}.\mathbf{X} + \mathbf{D}.\mathbf{Q2}.\tilde{\mathbf{D}}$
$\mathbf{H}_{DY}$	$= \mathbf{I}-\mathbf{N}.\mathbf{X} - \mathbf{N}.\mathbf{Q2}.\tilde{\mathbf{D}}$
$\mathbf{H}_{DU}$	$= -\mathbf{D}.\mathbf{X} - \mathbf{D}.\mathbf{Q2}.\tilde{\mathbf{D}}$
$\mathbf{H}_{VY}$	$= \mathbf{N}.\mathbf{Y} - \mathbf{N}.\mathbf{Q2}.\tilde{\mathbf{N}}$
$\mathbf{H}_{VU}$	$= \mathbf{D}.\mathbf{Y} - \mathbf{D}.\mathbf{Q2}.\tilde{\mathbf{N}}$

Table 2.1. Closed loop transfer functions.

The final step of the design process is the computation of the controller from the formulae

$$K1 = (Y - Q2.\tilde{N})^{-1}(Q1)$$

and

$$K2 = (Y - Q2.\tilde{N})^{-1}(X + Q2.\tilde{D}). \qquad (2.3.4)$$

The matrix

$$K0 = Y^{-1}.X \qquad (2.3.5)$$

may be thought of as an initial stabilizing controller for the one-degree-of-freedom system shown in figure 2.2.

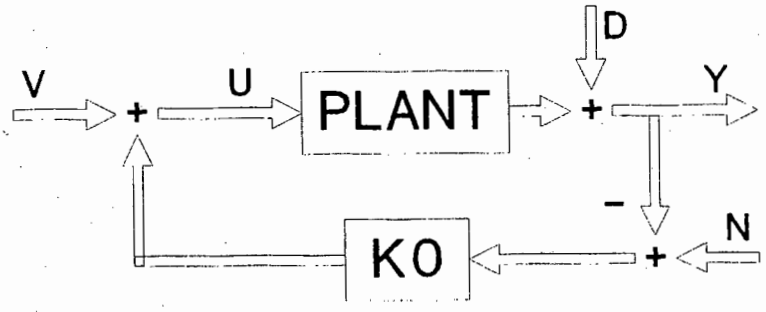


Figure 2.2. One-parameter control system configuration.

2.4. COMPUTING THE COPRIME MATRIX FRACTIONS.

For the general case, when a state-space representation of the plant is available together with stabilizing state feedback matrices, formulae for stable coprime fractions are available [N1]. Zhao and Kimura [Z2] give formulae based on the Smith-McMillan form of the plant transfer function matrix. However neither of these sets of equations are necessarily the most convenient to use; in particular there are two special cases which lend themselves to simplified computation of coprime factorizations.

1. *Stable plant.* [F2], [G5]
   In this case the factorization is trivial, and no stabilizing controller is required.

$$\mathbf{N} = \tilde{\mathbf{N}} = \mathbf{G}$$

$$\mathbf{D} = \tilde{\mathbf{D}} = \mathbf{Y} = \mathbf{I}$$

$$\mathbf{X} = \mathbf{0} \qquad (2.4.1)$$

2. *Stable initial controller.*
   Many unstable plants can be stabilized using a stable controller $\mathbf{K0} \in \mathbf{M(S)}$; Vidyasagar [V1] gives the conditions under which this is possible (corollary 5.3.2). In this case a right-coprime factorization $\mathbf{N,D} \in \mathbf{M(S)}$ can be chosen as

$$\mathbf{N} = \mathbf{G}(\mathbf{I} + \mathbf{K0.G})^{-1}$$

$$\mathbf{D} = (\mathbf{I} + \mathbf{K0.G})^{-1}$$

$$\mathbf{X} = \mathbf{K0}$$

$$\mathbf{Y} = \mathbf{I} \qquad (2.4.2)$$

   Proof

$$\mathbf{N.D}^{-1} = \mathbf{G}(\mathbf{I} + \mathbf{K0.G})^{-1}.(\mathbf{I} + \mathbf{K0.G}) = \mathbf{G}$$

$$\mathbf{X.N} + \mathbf{Y.D} = \mathbf{K0.G}(\mathbf{I} + \mathbf{K0.G})^{-1} + (\mathbf{I} + \mathbf{K0.G})^{-1}$$

$$= (\mathbf{K0.G} + \mathbf{I})(\mathbf{I} + \mathbf{K0.G})^{-1}$$

$$= \mathbf{I}$$

   As **K0** is a stabilizing controller, **N** and **D** are stable, representing the closed loop transfer functions from V to Y and V to U in figure 2.2 respectively. Since the

Bezout Identity is also satisfied, it follows that **N** and **D** are right-coprime ([V1] corollary 4.1.17).

∎

Similar expressions exist for stable left-coprime $\tilde{\mathbf{N}}$ and $\tilde{\mathbf{D}}$, and are derived in an analogous fashion; they are

$$\tilde{\mathbf{N}} = (\mathbf{I} + \mathbf{G}.\mathbf{K0})^{-1}.\mathbf{G}$$

$$\tilde{\mathbf{D}} = (\mathbf{I} + \mathbf{G}.\mathbf{K0})^{-1}. \qquad (2.4.3)$$

While the expressions for **N** and **D** may be quite complex, their algebraic form need not be computed explicitly. In the frequency domain these expressions are easily evaluated at discrete frequencies in terms of the transfer functions **G** and **K0**. In the time domain their impulse responses are required; these may be obtained by simulating the corresponding closed loop systems.

Gustafson and Desoer [G6] in fact propose that the frequency and impulse responses of the matrix fractions be obtained directly from plant measurements, without explicitly identifying a model for the plant. For example, the Fourier transform of the impulse response could be used to give the frequency response. Unfortunately, for true multivariable systems, this approach is compatible with the diagonal factorization discussed below only when the plant is stable.

2.5. DIAGONAL FACTORIZATION.

It is interesting to note that if the matrix $\mathbf{H2}_c$ is diagonal, then the individual elements of a specific closed loop transfer function can be written as

$$\mathbf{H}_c[i,j] = \mathbf{H0}_c[i,j] + \sum_{k=1}^{n} \left[\mathbf{H1}_c[i,k].\mathbf{Q}[k,j].\mathbf{H2}_c[j,j] \right]. \qquad (2.5.1)$$

Thus column j of $\mathbf{H}_c$ depends only on column j of **Q**, and the design may be reduced from a single design problem of size αn^2 to n independent sub-problems each of size αn. Size here refers to the dimension of the search vector used by the quadratic programming algorithm. This reduction, when possible, greatly extends the scale of design problems which may be tackled given limited computer memory.

Consider a plant with 4 inputs and 4 outputs, and where each element of **Q** has 5 decision variables. Here the reduction results in 4 sub-problems each of size 20, instead of a single problem of size 80. Each of the linear constraint vectors will require n times as much storage for the single problem, and there will usually be about n times as many of them, increasing the storage requirements by a factor n^2. For example, assume 1000 linear constraints are generated per sub-problem, and 8 bytes are needed per floating point number; then 1000.20.8 = 160 kilobytes are required to store the constraint vectors for each sub-problem, as opposed to 4.1000.80.8 = 2560 kilobytes for the single problem. Note that since the sub-problems are solved independently, the constraints for each need not be stored simultaneously. Furthermore the storage requirement for the six matrices in the quadratic programming algorithm is only 20^2.8.6 = 19200 bytes for each sub-problem, instead of 80^2.8.6 = 307200 bytes.

The time required to produce the final solution is usually also reduced for the partitioned problem as, although there are n problems to solve instead of just one, each one is very much simpler. An additional advantage of subdividing the problem is that conflicts within the engineering specifications are generally easier to identify and resolve, as there are fewer specifications in each sub-problem. Unfortunately the partitioning scheme hinges on a diagonal $\mathbf{H2}_c$. The circumstances under which this may be arranged are investigated next. From table 2.1,

$$H2_C = \begin{cases} \mathbf{I}, & c \in \{ RY, RU \} \\ \tilde{\mathbf{D}}, & c \in \{ DY, DU, NY, NU \} \\ \tilde{\mathbf{N}}, & c \in \{ VY, VU \}. \end{cases}$$

The matrix I is diagonal by definition. Clearly both $\tilde{\mathbf{N}}$ and $\tilde{\mathbf{D}}$ cannot be diagonal simultaneously, except when the plant is diagonal; this case will not be considered further as it may be solved using standard single variable methods. Assuming that $\tilde{\mathbf{D}}$ may be chosen to be diagonal, it is then necessary to forgo the opportunity of explicitly designing the closed loop transfer functions H_{VY} and H_{VU}. This is considered a small sacrifice compared to the advantages of the resulting independence. By comparison many other multivariable design methods, such as the INA and characteristic loci methods, allow disturbances to be considered explicitly at either the input or the output of the plant, but not at both simultaneously.

The statements above should not be taken to imply that the designer has no control over the closed loop responses H_{VY} and H_{VU}, only that it will not be possible to design them *explicitly*. Since

$$H_{VY} = H_{DY}.G,$$

and $$H_{VU} = H_{DU}.G,$$

these responses may be designed indirectly, particularly in the frequency domain, by careful shaping of the H_{DY} and H_{DU} responses, bearing in mind the characteristics of the plant which can be thought of as a pre-filter.

Definition 2.1
A matrix factorization will be termed diagonal when the denominator matrix is diagonal.

∎

A technique for computing a diagonal factorization of the plant is presented below in the form of a simple algorithm. While the factorization is not always coprime, a simple test is available to determine if the factorization is coprime.

According to Gustafson and Desoer [G6] (1985),

> *"There is no reliable software available that will perform coprime factorizations, multiplications or additions of matrices over the ring of polynomials or rational functions. Much of the available software suffers from numerical sensitivity and ill-conditioning."*

It is expected that the simplicity of the proposed diagonal factorization method will make it less susceptible to the numerical problems described above. The method has performed well in the applications tested.

The two theorems that follow are based on theorems found in Vidyasagar [V1]. While he generally treats only the right-coprime case explicitly, the corresponding left-coprime forms used below follow readily. Theorem 2.1 is derived from problem 4.1.11 of [V1].

Theorem 2.1
Let $\mathbf{G} \in \mathbf{M(F)}$ have a left-coprime factorization $\mathbf{A,B} \in \mathbf{M(S)}$. Then $\mathbf{G} + \mathbf{H}$ has a left-coprime factorization $\mathbf{A} + \mathbf{B.H,B}$ for all $\mathbf{H} \in \mathbf{M(S)}$.

Proof
As **A** and **B** are left-coprime, there exist $\mathbf{X,Y} \in \mathbf{M(S)}$ such that

$$\mathbf{A.X} + \mathbf{B.Y} = \mathbf{I} \qquad (2.5.2)$$

([V1] corollary 4.1.17). Define $\mathbf{Y'} \in \mathbf{M(S)}$ as

$$\mathbf{Y'} = \mathbf{Y} - \mathbf{H}.\mathbf{X}. \qquad (2.5.3)$$

Then

$$(\mathbf{A} + \mathbf{B}.\mathbf{H})\mathbf{X} + \mathbf{B}.\mathbf{Y'} = \mathbf{A}.\mathbf{X} + \mathbf{B}.\mathbf{Y} = \mathbf{I}. \qquad (2.5.4)$$

Thus $(\mathbf{A} + \mathbf{B}.\mathbf{H}), \mathbf{B} \in \mathbf{M(S)}$ are left-coprime ([V1] corollary 4.1.17). Furthermore

$$\mathbf{B}^{-1}(\mathbf{A} + \mathbf{B}.\mathbf{H}) = \mathbf{G} + \mathbf{H}. \qquad (2.5.5)$$

∎

Diagonal factorization algorithm.
Split the plant into stable and unstable components

$$\mathbf{G} = \mathbf{G}_s + \mathbf{G}_u$$

such that $\mathbf{G}_s \in \mathbf{M(S)}$, and $\mathbf{G}_u \in \mathbf{M(F)}$ contains only the unstable poles of $\mathbf{G}$. Then, according to theorem 2.1, if $\mathbf{G}_u$ has a diagonal left-coprime factorization, so does $\mathbf{G}$.

A left factorization of $\mathbf{G}_u$

$$\mathbf{G}_u = \tilde{\mathbf{D}}^{-1}.\tilde{\mathbf{N}}_u,$$

with $\tilde{\mathbf{N}}_u, \tilde{\mathbf{D}} \in \mathbf{M(S)}$ and $\tilde{\mathbf{D}}$ diagonal, may be produced as follows :

$$\text{let} \quad \mathbf{G}_u[i,j] = \frac{a_{ij}}{b_{ij}}, \qquad i,j \in \{1, 2, \ldots n\} \qquad (2.5.6)$$

$$\text{and} \quad d_i = \text{LCM } b_{ik}, \qquad k \in \{1, 2, \ldots n\} \qquad (2.5.7)$$

where a_{ij}, b_{ij}, $d_i \in \mathbf{R}[z]$. Choose some polynomial $c_i \in \mathbf{R}[z]$ with the same order as d_i, and all of its zeros in C_-; then define the elements of the denominator matrix as

$$\tilde{D}[i,i] = \frac{d_i}{c_i} \qquad (2.5.8)$$

Finally the numerator matrix is given by,

$$\tilde{N}_u = \tilde{D}.G_u$$

and $$\tilde{N} = \tilde{D}.G \qquad (2.5.9)$$

Next it is necessary to determine if this diagonal factorization is left-coprime. Theorem 2.2 provides a simple test to establish if a matrix pair is coprime or not.

Theorem 2.2
Let $\mathbf{A},\mathbf{B} \in \mathbf{M(S)}$ each have n rows, and let the sum of the number of columns of each be at least n. Then **A** and **B** are left-coprime if rank([**A B**]) = n at all points in C_{+e}.

Proof
There exists $\mathbf{U} \in \mathbf{U(S)}$ such that

$$[\mathbf{A}\ \mathbf{B}]\mathbf{U} = [\mathbf{R}\ \mathbf{0}], \qquad (2.5.10)$$

where $\mathbf{R} \in \mathbf{M(S)}$ is a greatest common left divisor of **A** and **B** ([V1] corollary B.2.15). Further there exist $\mathbf{X},\mathbf{Y} \in \mathbf{M(S)}$ such that

$$\mathbf{A.X} + \mathbf{B.Y} = \mathbf{R}. \qquad (2.5.11)$$

([V1] theorem 4.1.7).

Since **U** is unimodular, det(**U**) $\in$ **U** ([V1] fact B.1.26), and thus det(**U**) is nonzero at all points in C_{+e}. Thus **U** is nonsingular in C_{+e}, and

$$\text{rank}([\mathbf{A}\ \mathbf{B}]) = \text{rank}(\mathbf{R}). \qquad (2.5.12)$$

If rank([**A B**]) = n in C_{+e}, then det(**R**) is nonzero in C_{+e}.

Note that $\det(\mathbf{R}) \in \mathbf{S}$ by definition of the determinant, and therefore $\det(\mathbf{R})$ has neither poles nor zeros in C_{+e}. Consequently $\det(\mathbf{R}) \in \mathbf{U}$, $\mathbf{R} \in \mathbf{U(S)}$ ([V1] fact B.1.26), and $\mathbf{R}^{-1} \in \mathbf{M(S)}$.

Multiplying (2.5.11) on the right by $\mathbf{R}^{-1}$ gives

$$\mathbf{A.X.R}^{-1} + \mathbf{B.Y.R}^{-1} = \mathbf{R.R}^{-1} = \mathbf{I}. \qquad (2.5.13)$$

As $\mathbf{X.R}^{-1}, \mathbf{Y.R}^{-1} \in \mathbf{M(S)}$, it follows that **A** and **B** are left-coprime ([V1] corollary 4.1.17).

∎

Remarks

Although not required here, an "only if" clause for this theorem can also be proved. A similar theorem, using **R**[z] in place of **S**, is found in Kailath [K1].

The application of this theorem to the diagonal left factorization described above is eased by the diagonal structure of $\tilde{\mathbf{D}}$. $[\tilde{\mathbf{N}}_u \ \tilde{\mathbf{D}}]$ clearly has full rank in C_{+e}, except (possibly) at the unstable poles of **G**. At each of these poles z_u, full rank is possible if the rows of $\tilde{\mathbf{N}}_u(z_u)$ corresponding to those of $\tilde{\mathbf{D}}(z_u)$ which are now zero, are linearly independent. Stable matrices, and matrices where the unstable poles of different rows are distinct, are amongst those which have a diagonal left-coprime factorization.

As an example, consider the unstable plant

$$\mathbf{G}(z) = \begin{bmatrix} \dfrac{-10.517}{z - 1.1052} & \dfrac{-10.517}{z - 1.1052} \\ \dfrac{-10.517}{z - 1.1052} & \dfrac{-11.070}{z - 1.2214} \end{bmatrix} z^{-2}$$

examined in [Z2]. Applying the technique above gives the

diagonal left factorization

$$\tilde{\mathbf{D}}(z) = \begin{bmatrix} \dfrac{z - 1.1052}{z - 0.9} & 0 \\ 0 & \dfrac{(z - 1.1052)(z - 1.2214)}{(z - 0.9)^2} \end{bmatrix}$$

$$\tilde{\mathbf{N}}(z) = \begin{bmatrix} \dfrac{-10.517}{z - 0.9} & \dfrac{-10.517}{z - 0.9} \\ \dfrac{-10.517(z-1.2214)}{(z - 0.9)^2} & \dfrac{-11.070(z-1.1052)}{(z - 0.9)^2} \end{bmatrix} z^{-2}$$

Evaluating $[\tilde{\mathbf{N}}\ \tilde{\mathbf{D}}]$ at the unstable poles $z = 1.1052$ and $z = 1.2214$ gives

$$\begin{bmatrix} -41.96 & -41.96 & 0 & 0 \\ 23.76 & 0 & 0 & 0 \end{bmatrix}$$

and

$$\begin{bmatrix} -21.93 & -21.93 & 0.36 & 0 \\ 0 & -8.34 & 0 & 0 \end{bmatrix}$$

respectively. Both of these matrices have full row rank, and thus $\tilde{\mathbf{N}}$ and $\tilde{\mathbf{D}}$ are left-coprime.

2.6. THE QSTEP PARAMETER.

A result from factorization theory is that all stable closed loop transfer functions may generated by equation (2.3.3) for some stable transfer function matrix $\mathbf{Q} \in \mathbf{M(S)}$. When the domain of Q is restricted, this fact no longer holds. Unfortunately some restriction is inevitable when Q is

represented on a finite computer.

To produce an acceptable engineering solution it is generally not essential that all possible transfer functions be generated; a representative range suffices. Consider a typical computer representation of real numbers, where both the range of numbers, as well as the precision of the representation, is limited; yet for most problems this set of values available is adequate. Similarly it is required that the set of possible values for **Q** spans an adequate subset of all stable transfer function matrices, and with adequate precision.

The design method of Boyd [B6] is based upon a finite impulse response (FIR) representation of **Q**; in essence the method requires a matrix of proper stable polynomial ratios, where the denominator polynomials are fixed, and the coefficients of the numerator polynomials are determined by the search algorithm. To what extent does this representation of **Q** approximate the set of all stable transfer functions? For low order filters, it would seem rather poorly.

A clue to the physical meaning of **Q** is obtained from the case where the plant is stable. Choosing the factorization of (2.4.1) gives

$$\mathbf{H}_C = \mathbf{Q}, \quad C \in \{RU, NU\}.$$

Here **Q** is required to represent the closed loop transfer functions to the plant input. In the time domain it is clearly seen that the order of the FIR filter marks a time window over which control actions, following a disturbance impulse, may be taken. A high order filter is therefore necessary if a 'slow' control is desired. Unfortunately the dimension of the search vector in the quadratic programming algorithm is directly proportional to the order of the FIR filter, which makes the use of high order filters

prohibitive on a small computer. In order to gain the computational advantages of using a low dimension search vector, and yet retain some of the benefits of a high order FIR filter, the following (first order) parameterization was proposed [T3] :

$$\mathbf{Q}[i,j](z) = q_{0,ij} + \sum_{k=1}^{p-1} \left[q_{k,ij} \sum_{r=1}^{QSTEP} z^{-(r + (k-1)*QSTEP)} \right] \qquad (2.6.1)$$

where $q_{k,ij}$, $k \in \{0, 1, \ldots p-1\}$, are the coefficients (decision variables) of the new filter $\mathbf{Q}[i,j]$. The number of decision variables per element of $\mathbf{Q}$, p, is also referred to as NVARS by the expert system. QSTEP is a positive integer which effectively stretches the FIR filter; the standard FIR form results when QSTEP is unity. This parameter gives the designer a further degree of freedom; generally QSTEP is chosen in the light of the required speed of response (relative to the sampling rate).

This first order parameterization performs well in many cases; however the control signal from designs of this form often exhibit undesirable high frequency properties, seen as sharp changes in the control action. To resolve this problem, a second order parameterization has been developed, which gives a much smoother control action.

As before, let $q_{k,ij} \in \mathbf{R}$, $k \in \{0, 1, \ldots p-1\}$, be the coefficients of the filter $\mathbf{Q}[i,j]$. Then the filter can be parameterized as

$$\mathbf{Q}[i,j](z) = q_{0,ij} + \sum_{k=1}^{p} \left[\sum_{r=1}^{QSTEP} \alpha_{k,r} z^{-(r + (k-1)*QSTEP)} \right] \qquad (2.6.2)$$

where

$$\alpha_{k,r} = \begin{cases} QSTEP*q_{k,ij} & \text{when } k = 1 \\ (QSTEP - r)*q_{k-1,ij} & \text{when } k = p \\ r*q_{k,ij} + (QSTEP - r)*q_{k-1,ij} & \text{elsewhere} \end{cases}$$

Figure 2.3 shows the effective weighting functions for the first and second order parameterizations. These are given in the form of FIR filters qp[k], k ϵ (0, ... p), for each of the parameters $q_{k,ij}$, for the case where p = 4, and QSTEP = 3. Thus the composite FIR filter is given by

$$Q[i,j](z) = \sum_{k=0}^{p-1} \left[q_{k,ij} \sum_{t=0}^{\infty} \left[qp[k](t).z^{-t} \right] \right] \qquad (2.6.3)$$

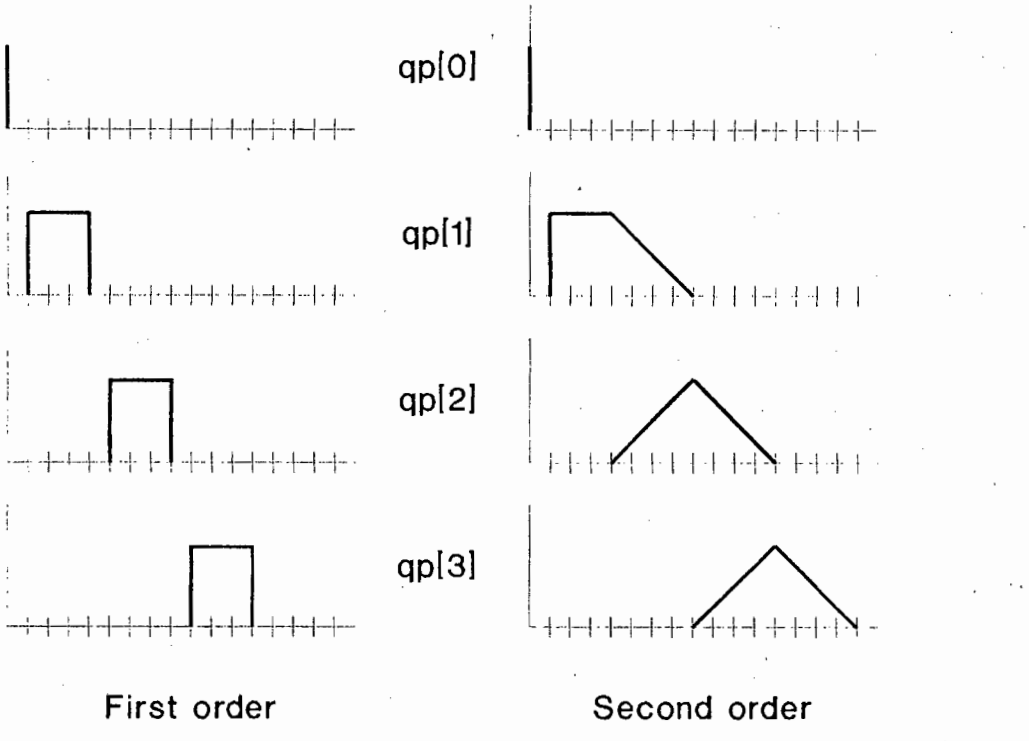


Figure 2.3. Weighting functions for the first and second order Q parameterizations.

The merit of the QSTEP parameter is most clearly illustrated by a single-input single-output example. Consider the plant, sampled at a rate of 1 Hz, with transfer function

$$G(z) = \frac{0.01}{(z - 0.8)(z - 0.95)}$$

Let the design problem be the minimization of the cost function J based on the response to a unit step disturbance at the plant output,

$$J = \sum_{k=0}^{50} \left[h_{DY}(kT)^2 \right]$$

subject to the constraint on the control signal

$$| h_{DU}(kT) | \leq 5.0, \qquad k = 0, 1, \ldots 50$$

and the asymptotic disturbance rejection requirement

$$H_{DY}(1) = 0.$$

Table 2.2 lists the minimum value found for the cost function J for different combinations of the number of decision variables p and the value of QSTEP. Using 5 decision variables, the minimum cost is obtained with QSTEP set at 4, and this cost is only slightly higher than that of the best long FIR filter. Figure 2.4 shows the step responses for the cases (a: p=5, QSTEP=1), (b: p=5, QSTEP=4) and (c: p=25, QSTEP=1). Note that linear interpolation is used between the output values at the sampling instants in this and the other graphs.

p	QSTEP	Cost
5	1	5.442
5	2	5.008
5	3	4.960
5	4	4.936
5	5	5.140
10	1	4.915
15	1	4.910
25	1	4.910

Table 2.2. Design cost for various values of p and QSTEP.

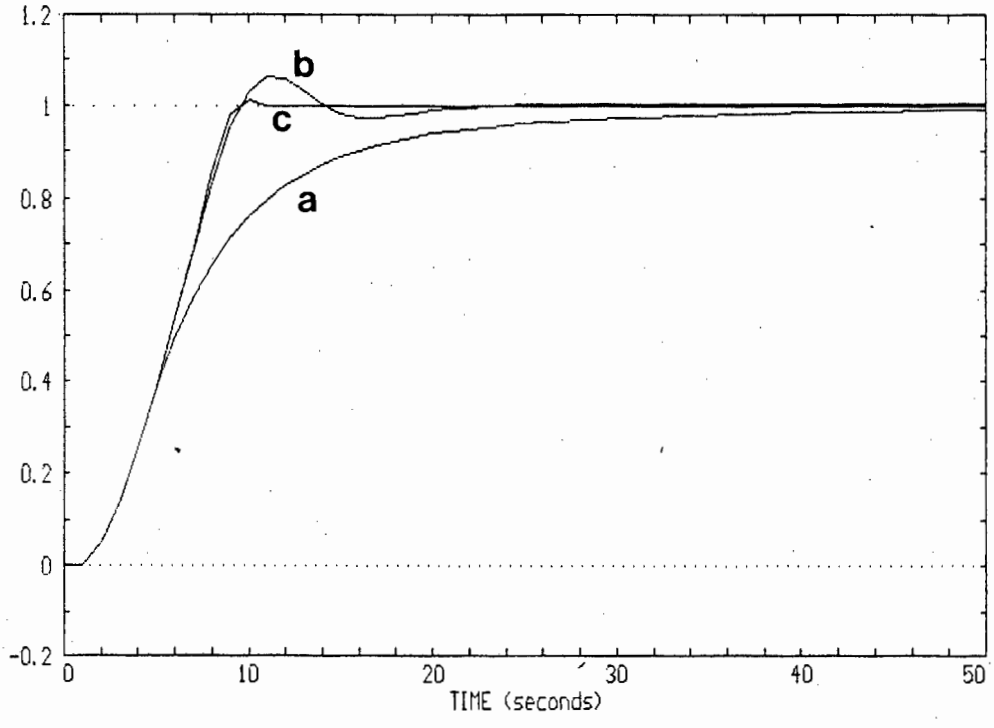


Figure 2.4. Step responses for various values of p and QSTEP.

This SISO example is also used to illustrate the difference between the first and second order parameterizations. Using the first order approximation with p set at 5, the lowest cost (4.948) is also achieved with QSTEP = 4. Figure 2.5 shows the control signals generated using the first (a) and second (b) order parameterizations, with p = 5 and QSTEP = 4. It can clearly be seen that the second order

parameterization gives a much smoother control action.

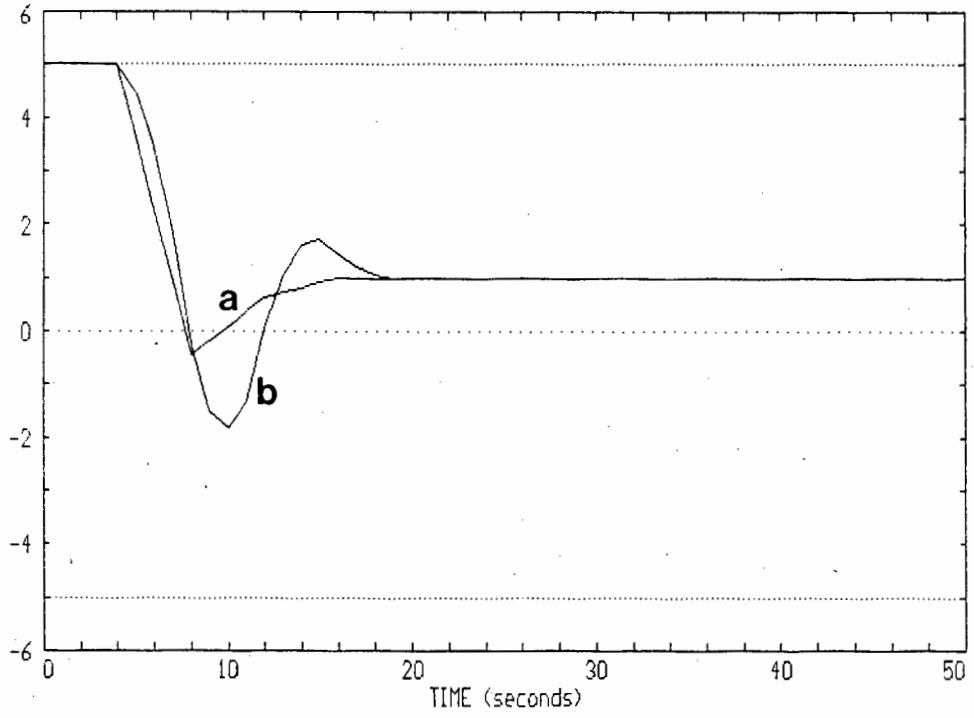


Figure 2.5. Control signal for the first (a) and second (b) order Q parameterizations.

2.7. THE CLOSED LOOP POLES.

From equation 2.3.3 and table 2.1 it is clear that the closed loop poles are given by those of the stable transfer function matrices **N**, $\tilde{\mathbf{D}}$, $\tilde{\mathbf{N}}$, **D**, **X**, and **Y**. Nevertheless there is a considerable amount of freedom available for choosing the closed loop pole positions, which in turn results from the choice of the nominal stabilizing controller **K0**, the transfer function matrix **Q**, and the left factorization $\tilde{\mathbf{N}}$ and $\tilde{\mathbf{D}}$.

The choice **K0** (equation 2.3.5) determines the poles for $\mathbf{H0}_c$ (the nominal closed loop system) in equation 2.3.3. When equation 2.4.2 is used to compute the matrix fractions **N** and **D**, the choice of K0 also determines the poles for $\mathbf{H1}_c$. With

stable plants it is common to use equation 2.4.1 to compute **N** and **D** (**K0** = **0**); in this case the poles of the plant will be amongst the closed loop poles.

The diagonal factorization described in section 2.5 also offers some freedom for choosing the poles. The zeros of the polynomial c_i become poles of $\tilde{D}$ (equation 2.5.8), and thus of $\mathbf{H2}_c$ and the closed loop system, may be chosen freely. The CACSD package computes the c_i polynomial based on the zeros of d_i (equation 2.5.7) and a maximum pole modulus parameter ß specified by the designer, with $0 < ß < 1$. Let d_i have the form

$$d_i = \alpha_i \prod_k (z - p_{i,k}) \qquad (2.7.1)$$

with $$p_{i,k} = r_{i,k} e^{j\theta_{i,k}} \qquad (2.7.2)$$

Since d_i contains only the unstable poles of the plant, $r_{i,k} \geq 1$. Then c_i is chosen by the CACSD package as

$$c_i = \prod_k (z - s_{i,k}) \qquad (2.7.3)$$

where $$s_{i,k} = \begin{cases} (r_{i,k})^{-1} e^{j\theta_{i,k}} & (r_{i,k})^{-1} \leq ß \\ ß.e^{j\theta_{i,k}} & (r_{i,k})^{-1} > ß \end{cases} \qquad (2.7.4)$$

The choices mentioned above are virtually irrelevant when **Q** can span the entire set of stable transfer function matrices, since the stable poles of $\mathbf{H1}_c$ and $\mathbf{H2}_c$ can be cancelled by zeros of **Q**. Unfortunately this is not generally true when using any finite computer representation for **Q**. Here the poles of **Q**, which may be chosen by the designer, appear as poles of the closed loop system, and the poles of $\mathbf{H1}_c$ and $\mathbf{H2}_c$, and those of **Q** itself, cannot always be cancelled by zeros of **Q** (which are selected by the design algorithm). This application, as well as that described by Boyd [B6], use a FIR filter parameterization for **Q**, which

has poles at z = 0 only.

This entire section should be tempered by the knowledge that the importance of the pole positions diminishes as the number of poles increases, and systems designed using this method have a large number of poles. A 50 tap FIR filter provides an effective illustration of this principle; an extremely wide range of responses may be generated by varying the filter coefficients, while the pole positions remain fixed at z = 0. Boyd [B6] gives a similar example showing that even when the closed loop pole positions differ greatly, the overall response of the system does not necessarily change much.

2.8. THE s DOMAIN.

While the CACSD package has been designed for z domain transfer functions, modifications for it to operate in the s domain should not be difficult. Frequency domain responses will be evaluated at points on the imaginary axis instead of the unit circle; time domain responses will require integration of differential equations instead of summation of difference equations. Possibly the most important change is that the elements of the Q matrices will require a different parameterization; in general they will still take the form of a transfer function where the denominator coefficients are fixed, and the numerator coefficients are computed by the quadratic programming algorithm. It may also be feasible to transfer the FIR filter structure of these elements in the z domain to the s domain using a sum e^{-skT} terms with variable coefficients. However the z domain transfer functions are better suited for digital computer implementation, and algorithms for this domain are usually more efficient than the corresponding s domain versions.

2.9. SUMMARY.

To improve the efficiency of the design method of Boyd *et al.* [B6], a diagonal factorization technique has been developed. This allows the multivariable design problem to be reduced to a number of smaller sub-problems, which may then be solved independently. Although the diagonal factorization is not always coprime, it is suitable for a wide range of plant transfer function matrices. A theorem to check that the factorization is coprime has been developed, and is easy to apply. Formulae for (non-diagonal) coprime factorizations, where the nominal stabilizing controller is stable, have also been presented. A novel parameterization for the design transfer function Q has been introduced; by appropriate choice of the QSTEP parameter the designer benefits from the efficiency of a low order approximation and while enjoying almost the same precision as for a high order approximation.

CHAPTER THREE. IMPLEMENTATION OF THE CACSD PACKAGE.

3.1. INTRODUCTION.

In chapter 2 techniques for improving the efficiency of the design method of Boyd [B6] were described. This chapter discusses the structure and implementation of a CACSD package using these techniques, and describes some additional strategies for improving the efficiency of the algorithms employed.

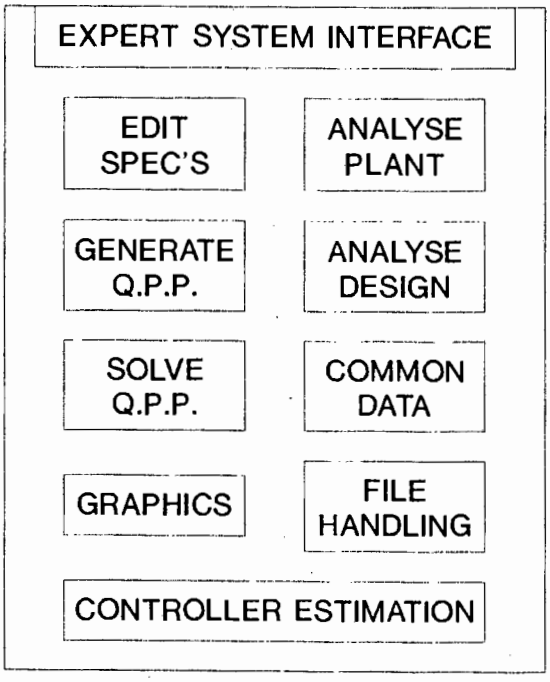


Figure 3.1. Overall structure of CACSD package.

3.2. STRUCTURE OF THE CACSD PACKAGE.

The CACSD package is structured as a collection of subroutines which the expert system can call upon. The main functions are shown in figure 3.1.

The plant transfer function matrix G(z) is read from file, or entered via a spreadsheet style table. Left and right stable diagonal matrix fractions of the plant are computed; the package also checks that they are coprime, and returns this information to the expert system. Under certain circumstances, for example when the plant is unstable, a nominal stabilizing controller transfer function matrix K(z) must also be specified.

Performance constraints on individual closed loop step and frequency responses, as well as those on the singular values, are entered using the form shown in figure 3.2, which is similar to a spreadsheet. The expert system discussed in chapters 4 and 5 also allows constraints to be entered on multiple elements of a response simultaneously, by copying the constraints entered on one element of a response to other elements as required.

Time domain constraints : RY[1,1] max = 50 Seconds

Maximum

Value	From (s)	To (s)
1.1	0	30
1.05	30	50

Minimum

Value	From (s)	To (s)
0.8	10	30
0.95	30	50

Select using [Ctrl] cursor keys : ESC to exit

Figure 3.2. Entry of performance constraints.

The package includes graphics facilities for plotting the system step and frequency responses, as well as the minimum and maximum singular values of the various frequency responses; performance constraints on the particular response are also shown on the plot. The direct and inverse Nyquist arrays, with Gershgorin circles, may also be plotted.

Some of the decisions taken by the expert system to guide the user are based on analyses of the design performed by the CACSD package. The CACSD package can, for example, compute the actual closed loop response over a time or frequency interval, or determine whether or not a specification on the singular values of a frequency response has been satisfied.

The CACSD package also includes a simple controller estimation facility which allows a reduced order controller to be estimated and then evaluated. Details of this are given in chapter 8.

3.3. GENERATING THE QUADRATIC PROGRAMMING PROBLEM.

The closed loop transfer functions $H_c(z)$ are evaluated using the equation

$$H_c = H0_c + H1_c.Q.H2_c \qquad (3.3.1)$$

However it is not always necessary to determine the transfer function matrices $H0_c$, $H1_c$ and $H2_c$ in algebraic form. Considering responses from inputs R, N and D, it can be seen from table 3.1 that it is necessary to evaluate only **N.X** or **N** for responses with output Y ($H0_Y$ and $H1_Y$ respectively), **D.X** or **D** for those with output U ($H0_U$ and $H1_U$ respectively), and $\tilde{D}$ when the input is N or D (**H2**).

H_{RY}	=	0	+ $H1_Y.Q1$
H_{NY}	=	$H0_Y$	+ $H1_Y.Q2.H2$
H_{RU}	=	0	+ $H1_U.Q1$
H_{NU}	=	$H0_U$	+ $H1_U.Q2.H2$
H_{DY}	=	$I - H_{NY}$	
H_{DU}	=	$- H_{NU}$	

Table 3.1. Closed loop transfer functions.

In table 3.1 above,

$H0_Y = \mathbf{N.X}$,
$H1_Y = \mathbf{N}$,
$H0_U = \mathbf{D.X}$,
$H1_U = \mathbf{D}$,
and $H2 = \tilde{\mathbf{D}}$.

Table 3.2 below gives the forms used for H0 and H1 depending on the stability of the plant and nominal controller, and whether or not a right-coprime factorization of the plant is available. Note that in the case where the right factorization of the plant is not coprime, and the controller is not stable, the design will be sub-optimal. This is also the case when the left factorization of the plant is not coprime. In this table,

G_{stab} indicates if the plant is stable,
K indicates if a nominal controller has been specified,
G_{co} indicates if the diagonal right factorization $\mathbf{G = N.D^{-1}}$ is coprime,
K_{stab} indicates if the nominal controller is stable,
F is the common factor $(I + K.G)^{-1}$.
and x means don't care.

Gstab	K	Gco	Kstab	$H0_U$	$H0_Y$	$H1_U$	$H1_Y$
yes	no	x	x	0	0	I	G
yes	yes	x	x	F.K	G.F.K	I	G
no	yes	no	yes	F.K	G.F.K	F	G.F
no	yes	yes	x	F.K	G.F.K	D	N
no	yes	no	no	F.K	G.F.K	D	N
no	no	x	x	not allowed			

Table 3.2. Formulae for computing H0 and H1.

Frequency responses are evaluated at a set of discrete points, logarithmically spaced on the unit circle in the complex plane. The matrices **H0**, **H1** and **H2** are evaluated at these frequencies, and stored for future use. For step responses, only the impulse responses of **H0**, **H1** and **H2** must be stored. Impulse responses for the expressions in table 3.2 involving the complicated common factor **F** are evaluated through a simulation of the corresponding closed loop system made up of **G** and **K**. Using the notation of figure 3.3, **F.K** represents the closed loop response from input D to output B, **G.F.K** that from D to C, **F** that from A to B, and **G.F** that from A to C. Thus the algebraic forms of these matrices need not be determined explicitly. The program computes and stores these impulse and frequency responses; thus they need not be recomputed each time a response is evaluated, which saves much time.

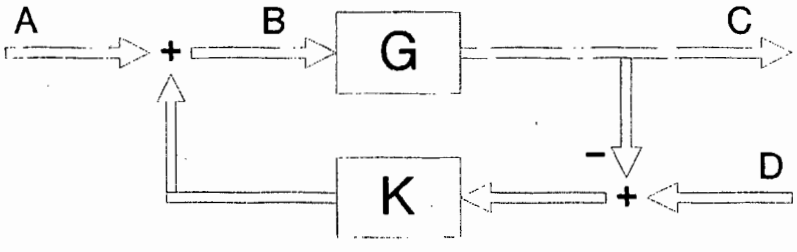


Figure 3.3. Closed loop system used to evaluate impulse responses.

Performance constraints on the closed loop responses are translated into linear constraints on the search vector, as shown by Boyd [B6]. Given the diagonal factorization of chapter 2, the equation for the closed loop response of element [i,j] can be written as

$$\mathbf{H}_c[i,j] = \mathbf{H0}_c[i,j] + \sum_{k=1}^{n} \left[\mathbf{H3}_c[i,k]_j . \mathbf{Q}[k,j] \right] \quad (3.3.2)$$

where

$$\mathbf{H3}_c[i,k]_j = \mathbf{H1}_c[i,k] . \mathbf{H2}_c[j,j]. \quad (3.3.3)$$

For clarity, the analysis below assumes a value of one for the QSTEP parameter used to model the elements of **Q**. This makes **Q**[i,j] a standard p-tap FIR filter, which is parameterized as

$$\mathbf{Q}[i,j] = \sum_{r=0}^{p-1} \left[q_{i,j,r} z^{-r} \right], \quad (3.3.4)$$

$q_{i,j,r} \in \mathbf{R}$. Define a parameter vector $\mathbf{x}_j \in \mathbf{R}^{np}$ made up of the coefficients of the FIR filters in column j of **Q**, according to the formula

$$q_{i,j,r} = \mathbf{x}_j[(i-1)n + r]. \quad (3.3.5)$$

Then the closed loop step response of a particular element at time uT may be written as

$$\begin{aligned} h_c[i,j](uT) &= h0_c[i,j](uT) \\ &\quad + \sum_{k=1}^{n} \sum_{r=0}^{p-1} \left[\mathbf{h3}_c[i,k]_j((u-r)T) q_{k,j,r} \right] \\ &= d0 + \sum_{k=1}^{n} \sum_{r=0}^{p-1} \left[d_{i,j,k,r} q_{k,j,r} \right] \\ &= d0 + \mathbf{d}^T \mathbf{x}_j, \end{aligned} \quad (3.3.6)$$

where d0, $d_{i,j,k,r} \in \mathbf{R}$, and $\mathbf{d} \in \mathbf{R}^{np}$. Thus constraints on the minimum or maximum value of a closed loop step response element [i,j] at time uT

$$v1 \leq h_c[i,j](uT) \leq v2 \tag{3.3.7}$$

can be transformed into linear constraints on the parameter vector $\mathbf{x}$

$$v1 - d0 \leq \mathbf{d}^T\mathbf{x}_j \leq v2 - d0. \tag{3.3.8}$$

The closed loop frequency response of a particular element, evaluated at a point w on the unit circle, may be written as

$$\begin{aligned} H_c[i,j](w) &= H0_c[i,j](w) \\ &\quad + \sum_{k=1}^{n} \sum_{r=0}^{p-1} \left[H3_c[i,k]_j(w) w^{-r} q_{k,j,r} \right] \\ &= c0 + \sum_{k=1}^{n} \sum_{r=0}^{p-1} \left[c_{i,j,k,r} q_{k,j,r} \right] \\ &= c0 + \mathbf{c}^T\mathbf{x}_j, \end{aligned} \tag{3.3.9}$$

where $c0$, $c_{i,j,k,r} \in \mathbf{C}$, and $\mathbf{c} \in \mathbf{C}^{np}$.

A constraint on the maximum modulus of this closed loop frequency response

$$| H_c[i,j](w) | \leq m \tag{3.3.10}$$

then becomes

$$| c0 + \mathbf{c}^T\mathbf{x}_j | \leq m. \tag{3.3.11}$$

Splitting these into real and imaginary components $c0r$, $c0i \in \mathbf{R}$, and $\mathbf{cr}$, $\mathbf{ci} \in \mathbf{R}^{np}$, gives

$$(c0r + \mathbf{cr}^T\mathbf{x}_j)^2 + (c0i + \mathbf{ci}^T\mathbf{x}_j)^2 \leq m^2. \tag{3.3.12}$$

Boyd *et al.* [B6] have shown how this complex modulus constraint may be approximated by linear constraints; an efficient representation [T2] developed for these is discussed later. Using these techniques, the constraint on the modulus of the frequency response element [i,j] may also be translated into linear constraints on the parameter vector **x**.

The linear constraints generated are grouped according to the performance constraints they represent; these groups are presented to the quadratic programming algorithm until all have been satisfied, or a conflict is found. When there is no feasible solution, corresponding to a conflict in the design specifications, the expert system uses the status of each group (satisfied, active, not satisfied, or not yet attempted) to explain the cause of the conflict (see section 5.5.6).

The design method also allows for optimization of the closed loop responses. The quadratic cost function to be minimized is again based on these responses. In the time domain, the response of are particular element, at time uT, has a cost

$$\begin{aligned} J_{i,j} &= (h_c[i,j](uT) - e)^2 \\ &= (\mathbf{d}^T\mathbf{x}_j + d0 - e)^2. \\ &= \mathbf{x}_j{}^T(\mathbf{d}\mathbf{d}^T)\mathbf{x}_j + 2(d0 - e)\mathbf{d}^T\mathbf{x}_j + (d0 - e)^2 \end{aligned} \qquad (3.3.13)$$

associated with it, where $e \in \mathbf{R}$ is an constant offset (usually 0 or 1). In the frequency domain, at frequency w, the corresponding cost is

$$J_{i,j} = | H_c[i,j](w) - f |^2$$

$$= (c0r + \mathbf{cr}^T\mathbf{x}_j - fr)^2 + (c0i + \mathbf{ci}^T\mathbf{x}_j - fi)^2$$

$$= \mathbf{x}_j^T(\mathbf{cr}.\mathbf{cr}^T + \mathbf{ci}.\mathbf{ci}^T)\mathbf{x}_j$$

$$+ 2(c0r - fr)\mathbf{cr}^T\mathbf{x}_j + 2(c0i - fi)\mathbf{ci}^T\mathbf{x}_j$$

$$+ (c0r - fr)^2 + (c0i - fi)^2. \qquad (3.3.14)$$

where fr, fi ϵ **R** are the real and imaginary components of the offset constant f ϵ **C** (also usually 0 or 1). Note that in both cases the cost is usually scaled by some real positive constant, refered to as the optimization weight; the cost is also defined as the square of the difference between the response and some constant offset. Furthermore, the symmetric matrices $\mathbf{dd}^T$, $\mathbf{cr}.\mathbf{cr}^T$, and $\mathbf{ci}.\mathbf{ci}^T$ are positive semi-definite; in practice the sum of many of these terms is generally positive definite. Thus there is always some finite value of $\mathbf{x}_j$ giving the minimum cost, and this value is almost always unique.

3.4. REPRESENTING THE LINEAR CONSTRAINTS.

An efficient representation for the linear constraints generated by the design method has been developed by the author [T2] (appendix D). The inherent structure in these constraints is exploited to reduce both the storage requirements and execution time of the quadratic programming algorithm substantially, without changing the characteristics of this algorithm in any way. The gains in efficiency are particularly significant for the frequency domain constraints. In the examples given in [T2], the constraints were represented using only about 15% of storage space for the equivalent simple linear constraints; execution speed was increased by a factor between 3.5 and 5.1 at the same time.

Formulae developed for the compound representation, which treats groups of linear constraints together, turn out to be much more efficient than the corresponding formulae for the equivalent group of linear constraints in standard form, which need to be treated independently. Most of the speed advantage is achieved at lines 45-56 in the QPSOL algorithm listed in section 3.6; during phase I some advantage is also obtained at lines 9-14.

3.5. SOLVING THE QUADRATIC PROGRAMMING PROBLEM.

The function of the quadratic programming algorithm QPSOL is to find the vector **x** which minimizes the quadratic cost function

$$\tfrac{1}{2}\mathbf{x}^T.\mathbf{EA}.\mathbf{x} + \mathbf{eb}^T.\mathbf{x}, \qquad (3.5.1)$$

and satisfies the set of linear constraints

$$C_i(\mathbf{x}) \geq 0. \qquad (3.5.2)$$

Each constraint C_j is represented by the linear equation

$$C_j(\mathbf{x}) = \mathbf{a}_j.\mathbf{x} + b_j. \qquad (3.5.3)$$

For clarity of the algorithm shown below, it is assumed that there is at least one constraint.

A two-phase algorithm is used to solve the quadratic programming problem; phase I locates a feasible point and, if one exists, phase II then finds the constrained optimum solution. This algorithm is based on two of the Gill-Murray active set methods given in Scales [S1]; note that algorithm 6.4 [S1] used for locating an initial feasible point is incorrect.

The columns of the active set **A** are made up of the vectors a_i of those constraints C_i which are active. An orthogonal QR factorization of **A**,

$$\mathbf{A} = \mathbf{Q}.\mathbf{R}, \tag{3.5.4}$$

with **Q** column orthonormal ($\mathbf{Q}^T.\mathbf{Q} = \mathbf{I}$), and **R** upper triangular, is used extensively in the algorithm. This **Q** matrix should not be confused with the matrices **Q1** and **Q2** used to parameterize the closed loop transfer functions. Scales [S1] gives efficient techniques for updating these QR factors when a column is added to or deleted from **A**, based upon the Householder transformation [G3]. The computational efficiency of the factorization has been significantly improved by taking advantage of the structure of matrix **R** and the Householder transformation matrix **P** during matrix multiplications.

When q constraints are active, the QR factors can be partitioned as

$$\mathbf{A} = \left[\mathbf{Qa} \mid \mathbf{Qb}\right]\left[\frac{\mathbf{R}}{\mathbf{0}}\right], \tag{3.5.5}$$

where **Qa** has q columns and **R** q rows. The matrix **Qb**, with np-q columns, is used frequently in the QPSOL algorithm.

3.6. THE QPSOL ALGORITHM.

```
1   Set phase2 = feasible = terminate = try_del = false
2   Set α = 1, q = 0
3   Set x = 0, A = 0, Q = I
4   REPEAT
5      IF (α > 0)
6         IF (phase2)
7            Set g = EA.x + eb
8         ELSE
```

```
 9            IF (C_i(x) ≥ 0 for all i)
10               Set feasible = terminate = true
11            ELSE
12               Set g = - Σa_i for all i
13               such that C_i(x) < 0
14            END
15         END
16      END
17      Set deletion = false
18      IF (NOT terminate)
19         IF (NOT try_del)
20            Set v = Qb^T.g
21            IF (phase2)
22               Solve Qb^T.EA.Qb.y = v for y
23               Set v = y
24            END
25            Set p = -Qb.v
26         END
27         IF (try_del OR (|Qb^T.g| < eps))
28            Solve R.μ = Qa^T.g for μ,
29            the Lagrange multipliers.
30            IF (no μ_i < 0)
31               Set terminate = true
32            ELSE
33               Find smallest μ_i, and delete the
34               corresponding constraint from A
35               Set q = q - 1
36               Update the factorization A = Q.R
37               Set deletion = true
38               Set α = 0
39            END
40         END
41      END
42      Set try_del = false
43      IF (NOT(terminate OR deletion))
44         Set addition = true
45         Find the smallest distance ß in direction p
46         to any constraint (C_j) which is currently
```

```
47          satisfied but not active.
48          IF (phase2)
49             ß = 1.001
50          ELSE
51             IF (none found in this direction)
52                Find the distance ß in direction p to
53                the furthest constraint (C_j) which is
54                currently violated.
55                Set addition = false
56             END
57          END
58          IF (phase2 AND (ß > 0.999))
59             Compute the distance δ to the minimum of the
60             cost function in the direction p, using the
61             formula
62                      δ = -(p^T.EA.p)/(g^T.p)
63             IF (δ < ß)
64                Set ß = δ
65                Set addition = false
66                Set try_del = true
67             END
68          END
69          Set α = ß
70          IF (α > 0)
71             Set x = x + αp
72          END
73          IF (addition)
74             Add constraint C_j to active set A
75             Update factorization A = Q.R
76             Set q = q + 1
77          ELSE
78             IF ((NOT phase2) OR (α < eps))
79                Set terminate = feasible = true
80             END
81          END
82       END
83       IF (feasible AND (NOT phase2))
84          Set terminate = false, phase2 = true
```

```
85          Set α = 1
86       END
87    UNTIL (terminate)
88    END
```

During phase II, the section of the algorithm for computing the smallest distance to a constraint boundary (line 45), takes advantage of the implicit scaling of the algorithm. This scaling is such that the distance ß to the minimum of the cost function, subject to the active constraints, is unity. Rounding errors (even when using double precision arithmetic) make this distance only approximately one, and the formula at line 62 is found to give a more accurate assessment of the distance.

Having solved the quadratic programming problem for **Q1** and **Q2**, it is possible to compute the controller transfer functions **K1** and **K2** from the formulae (2.3.4). This is not done explicitly; rather **Q1** and **Q2** are used directly to implement the controller, or to estimate a low order approximation, as discussed in chapter 8.

3.7. SUMMARY.

This chapter has dealt with issues relating to the implementation of the CACSD package. Efficient forms for evaluating the coprime matrix fractions have been given, and the translation of the control system design problem into a quadratic programming problem has been described. A very efficient representation for the linear constraints generated by this method, both in terms of memory requirements and execution speed, has been developed. Finally an algorithm for solving the resulting quadratic programming problem, derived from two active-set methods, is given.

CHAPTER FOUR. THE EXPERT SYSTEM.

4.1. INTRODUCTION.

Although the numerical design method discussed in chapters two and three is extremely powerful, it forms only one step in the overall design process. There are many issues which need to be addressed in order to apply it successfully, and it is here where the expert system proves very useful. Broadly speaking, the expert system coordinates the functions of the CACSD package, and provides an intelligent user interface. It assists the designer in formulating a comprehensive and achievable design specification, and in dealing with conflicting design constraints. The expert system also effectively extends the scope of the underlying CACSD package.

The CACSD package requires that the design be specified in terms of performance constraints on the closed loop response of the system. The design cycle comprises setting performance constraints and a cost function for the control system, and then using the CACSD package to find a controller which satisfies these. In general this is an iterative process whereby the specification is engineered in stages, with specifications being added, tightened, or relaxed at each step, until optimal performance is achieved. The main function of the expert system is to assist the user in formulating and refining this specification; the assistance takes place at many levels, ranging from simply explaining the commands available and how they should be used, to providing advice on how the specification could be improved.

Commands available to the user may be divided into two main groups. The first group comprises those used for editing the specification, solving for a controller, and examining the performance of the controller, for example the EDIT, SOLVE and PLOT commands. The second comprises those used to assist the designer, and includes the HELP, SUGGEST and NEXT STEP commands. This group, and the most important tasks performed by the expert system, are examined in detail below.

4.2. THE USER INTERFACE AND PHILOSOPHY.

The expert system attends to almost all of the interaction with designer. Unfortunately novice and experienced designers have different requirements of a user interface. Experienced designers need ready access to comprehensive and detailed information on the state of the design, and complete freedom to choose from all possible design alternatives; the expert system should also contribute towards improving the productivity of the expert. Novice designers are often overwhelmed by too much information, or too many degrees of freedom in the design method; they need assistance in interpreting the information, and guidance in choosing amongst the design options. Infrequent users of a design system, novice or otherwise, may require help with the design language. Taylor and Frederick [T1] give further examples of areas where 'less-than-expert' users experience difficulty with traditional CACSD software. The expert system user interface aims to deal with these issues.

The interface is largely command-driven for ease of use by proficient users; the expert system gives the experienced designer almost direct access to the underlying CACSD package, with only brief comments on the design progress. To assist novice and infrequent users there are also commands such as HELP, SUGGEST, and NEXT STEP. The concept of an expert system intercepting these types of commands has been termed 'command spy' by Larsson and Persson [L1]. This

structure allows users to request advice and assistance when desired. As the user's knowledge and experience with the system grows, they will tend to use the advice facilities less frequently.

Pang and MacFarlane [P1] propose that matters of engineering judgement be left to the designer. This application follows the same philosophy, with the responsibility for all the final design decisions resting on the designer. The expert system has little knowledge of the physical system to be controlled; for example it does not know whether a high frequency gain of +10 dB on the NU response is excessive or not. While it can and does generate warnings and reminders, this type of decision is ultimately left to the designer.

On the whole, the expert system does not question the specifications placed by the designer; it is assumed that the designer has some constructive purpose for each specification. However there are certain situations where the designer is warned of 'suspicious' specifications. Similarly, the expert system does little in checking that the advice given to the user is applied; it is assumed that the designer may have a valid reason for ignoring it.

The expert system has no automatic learning capabilities. Nevertheless, after many design sessions and through the skill of the knowledge engineer, the expert system has undergone a substantial indirect 'learning' experience, resulting in the current version of the design tool being significantly more capable than earlier versions [T4,T6]. One of the strengths of the expert systems approach is the relative ease with which this learning, or extension of the knowledge base, can be effected [T1].

4.3. EXPLAINING THE DESIGN LANGUAGE AND METHODOLOGY.

The design language, terminology and the overall design philosophy for the CACSD design method may be unfamiliar to novice or infrequent users; MacFarlane *et al.* [M2] point out that expert systems may contribute towards making design packages more easily accessible in these situations. The expert system contains a large amount of information on the use of the design tool, covering the design language and notation, the syntax and use of the commands, and the general philosophy for using this design method. While much of this is a simple menu-driven help system as shown below, or is programmed into the expert system explanation facility, a small measure of reasoning is used in generating the context-sensitive help. One of the objectives of the interface is to tutor the user in applying the design tool.

```
DY[1,1](frequency) >HELP
Help is available in the following areas :

1.      The design language and philosophy.
2.      A particular command.
3.      The step-by-step design guide.
4.      Refining your design.
5.      Dealing with conflicting constraints.
6.      Checking that the design is complete.

Which of these do you need help with ? 1

Choose one of the following :

1.      The design method overview.
2.      Terminology used in this program.
3.      How to plan your design.
4.      Selecting a response to work on.
5.      Plotting a step or frequency response.
6.      Setting constraints on a response.
7.      Setting up an optimization function.
8.      Finding a suitable controller.
9.      Selecting the design parameters.

....
```

Genesereth [G1] shows how an intelligent system may reconstruct the user's plan of action in order to provide advice on the correct usage of MACSYMA, a symbolic mathematics program. Jackson and Lefrere [J1] develop this

concept further. Larsson and Persson [L1] store typical command sequences in scripts, and match these against the history of commands issued by the user in an attempt to understand the user's intentions. Similar types of analysis could also be used to assist the control system designer. For example, when an invalid command is issued, the expert system could attempt to identify the designer's intentions from the history of design steps, and then explain the correct usage of the commands. At present it is felt that these approaches are not justified, in light of the commands seldom being chained into distinct sequences but rather being used somewhat independently. A simple notification of the error, combined with suggestions on how to obtain further help, suffices.

Nevertheless the expert system does monitor how it is being used by the designer. This information is useful, for example, when deciding whether or not certain features of the design system deserve explanation.

4.4. PRESENTATION OF THE DESIGN DATA.

One of the most important functions of any design system is that of providing the designer with information on the status of the design; this information should be both comprehensive and easily comprehended.

The CACSD package on its own does not provide comprehensive information on the design in the sense that only information relating to the specifications is made available, and the range of specifications permitted is limited. The expert system, often using supporting numerical software, attempts to effectively increase the range of specifications available, and also performs further analyses on the design, supplementing the information available. This aspect of the expert system is dealt with in detail later.

The expert system also attempts to provide the design information in an easily comprehended form. An example of this is the analysis of the Lagrange multipliers from the quadratic programming algorithm. From these it is possible to determine which of the constraints were satisfied, active or could not be satisfied; a constraint is said to be active when it evaluates to its boundary value. The expert system passes this information on to the designer, and also uses it to suggest approaches for dealing with conflicting specifications or improving the design. A typical message generated by the expert system to describe a conflict in the specifications, following an analysis of the Lagrange multipliers (from the quadratic programming algorithm QPSOL), is shown below :

```
The frequency domain constraints on DY[2,1] could not
be satisfied as they conflict with :
  the frequency domain constraints on DY[1,1]
  the time domain constraints on NU[1,1]
Of these, the Lagrange multipliers hint that the
frequency domain constraints on DY[1,1] are the most
probable cause of the conflict.
It is unlikely that the asymptotic constraints are
responsible for the conflict.
```

4.5. FORMULATING THE DESIGN SPECIFICATIONS.

To assist the user in drawing up and refining a specification, the expert system considers a list of common design attributes, characteristics of the plant, and the current specification. These are implemented as a collection of frames [W1]. The expert system does not have knowledge of many physical requirements of the control system, such as the limitations of the actuator. For example, while the expert system knows that a good design will often constrain the magnitude of the control signal response following a disturbance, and that to improve the speed of response it may be necessary to relax this constraint, it does not know whether a particular value for the constraint is satisfactory or not. Thus it can only remind the designer of

these issues; it is then up to the designer to take the engineering decisions.

There are two mechanisms to assist the designer in formulating a specification. These are invoked by the NEXT STEP and SUGGEST commands, and are discussed in detail below. In addition, the SUGGEST command provides advice on dealing with conflicts in the specification.

4.5.1 The NEXT STEP command.

A step-by-step design mode has been implemented to help novice designers produce a reasonable design, and at the same time assist them in learning to use the design tool effectively. This procedure guides the user through a sequence of design steps; the process is initiated and perpetuated by repeating the NEXT STEP command, which then recommends the next step to be taken, explaining why it may be necessary. The designer is free and encouraged to experiment with other commands during this process. Usually the recommended design step is not executed automatically, but the designer is only shown how to perform it, and is then expected to follow the instructions if required. In this way it is hoped that the user will learn to use the design tool independently.

An example of the use of the NEXT STEP command is shown below :

```
DY[1,1](frequency) >NEXT STEP
Do you want the amplitude of the control signal
to be limited following a step input ? /EXPLAIN?

Constraints on the RU and NU step responses limit
the control action taken following a command
input change. This in turn slows down the closed
loop system.  Since real actuators cannot provide
arbitrarily large control actions, it is good
practice to constrain this response; then
(hopefully) the design will be less susceptible
to saturation problems when implemented.

Do you want the amplitude of the control signal
to be limited following a step input ? YES
```

```
You need some specifications which will limit the
control signal at output U following a step input
at R. The best way to do this is to constrain the
time response of the RU signals; usually it is
necessary to place constraints on only the
initial part of this response. Another way to
achieve this is to constrain the frequency
response, in particular the high frequency
portion. The sequence of commands to use is
          >RU
          >TIME
          >EDIT ALL
```

In this example, the suggested sequence of commands first selects the RU step response. The EDIT command is then used to enter or modify the constraints on this response, which are presented in a spreadsheet format by the CACSD package. The ALL qualifier indicates that the constraints set will apply to all elements RU[i,j] of the response.

4.5.2 The SUGGEST command.

Similar to the step-by-step design mode, the SUGGEST command provides advice to the user on how to improve the design. Guided by appropriate questions, and considering the state of the design in terms of the current specifications and performance, the system can offer suggestions on how to deal with conflicting constraints, which specifications to modify in order to improve performance, or how to meet specifications not explicitly covered by the CACSD package. For example,

```
RY[1,1](time) >SUGGEST
Checking plant DC gain.
A steady state value of at least 2 will be
required at some input of the plant to achieve
asymptotic tracking following a unit step input.
Is this value acceptable ? YES
Do you require asymptotic rejection of
disturbances at input D ? YES

Complete asymptotic disturbance rejection
specifications have not yet been set.  These may
be specified using the 'SET ASYM REJECTION'
command.

....
```

```
DY[1,1](frequency) >SUGGEST
Do you wish to improve the closed loop
performance of the design ? YES
The design is completed in two independent
sections, refered to as Q1 and Q2. These
correspond to the closed loop responses as
follows :
   Q1 : responses RY and RU
   Q2 : responses DY, NY, NU, VY, and VU
Which of these sections (1 or 2) are you
interested in ? 2
Do you want me to check the design for unusual
combinations of optimization specifications ? YES

I did not find any abnormalities in the
optimization specifications.

Which column of the closed loop response are you
interested in ? 2
Are you prepared to relax any of the constraints
in order to improve the design performance ?
/EXPLAIN
The analysis of the Lagrange multipliers from the
last 'SOLVE' command indicates that some of the
constraints are limiting the design performance.

Are you prepared to relax any of the constraints
in order to improve the design performance ? YES
It should be possible to improve the disturbance
rejection performance by relaxing the time domain
constraints on column 2 of the NU responses.
Would you like to try this ? YES

Use the 'EDIT' command to relax the time domain
constraints on NU[1,2].
```

In most cases a particular suggestion is made only once; the designer is expected to take note of it at that time, and keep it in mind during the remainder of the design process. The decision to do this was taken in view of those situations where the designer does not wish to follow the recommendation given, and should not be repeatedly reminded of it. This should suit experienced designers, but not the novice designer who inadvertently ignores important advice. Some relief for this latter case is available, however; after all the relevant suggestions have been made, the designer is given the option of cycling through the suggestions once more.

There are two other more specific SUGGEST commands. The SUGGEST OPT command provides advice on using the optimization facilities to achieve the required performance. The SUGGEST EDIT command provides advice on the possibilities for changing the performance by placing constraints on the selected response. Examples of these commands are shown below.

```
NY[1,1](time) >SUGGEST OPT
I assume that you want to optimize the command
tracking, disturbance rejection, and decoupling
properties of the closed loop system.

I suggest that you use the following sequence of
commands to set optimization for the Q1 part of the
design.
          >RY
          >TIME
          >SET OPT 1 ALL
          >SET OPT 0.1 DIAG
This sets an optimization weight of 0.1 for the
diagonal elements, and a weight of 1 for the others.
You may need to refine this ratio of weights to
tradeoff the tracking response versus the cross-
coupling performance. Increasing the weight of the
diagonal elements will improve the tracking
performance at the expense of worse decoupling, and
vice-versa.

....

DY[1,1](frequency) >SUGGEST EDIT
The DY response relates to the closed loop system's
ability to reject disturbances at input D. The
designer should ensure that low frequency
disturbances are adequately suppressed; -20 dB is a
typical requirement for this range. At mid
frequencies the designer must ensure that the
responses do not peak excessively; anything over +3
dB is cause for concern. Robustness constraints on
the singular values may also be set; use the
'EDIT SVD' command for this.

In general the ability to reject high frequency
disturbances is limited. The gain of the diagonal
elements should be no more than about +3 dB over this
range; this can also be reduced by constraining the
high frequency response of the diagonal elements of
NY. The responses of the off-diagonal elements will
depend strongly on the high frequency behavior of the
plant, but should generally be less than 0 dB. A good
way to get reasonable behavior of this high frequency
range is by limiting the gain of the high frequency
portion of the NU response.
```

Note that the NY and DY step responses are closely related by the equation

NY = I - DY

where I is the identity matrix; constraints on the off-diagonal elements of these responses are therefore completely equivalent.

4.5.3 Checking the design for completeness.

The expert system is able to check that the design is complete. This is done by analyzing the specifications given to see that all the important design aspects have been covered. The designer is warned of possible omissions in the design specification, and advice is provided on how the design may be updated. In some cases the designer is simply required to check that a particular response is satisfactory.

The checking facility is initiated using the COMPLETE command; it may also be accessed via the HELP system. Some typical dialogue initiated by the COMPLETE command is shown below.

```
NY[1,1](frequency) >COMPLETE
Do you require a robust design ? YES

To ensure some robustness of the design, the SVD
plot of the NY frequency response should not show
any excessive peaking, and the high frequency
gain should taper off. Robustness specifications
can be entered as constraints on the maximum
singular values of the NY frequency response.
Details of this approach can be found in the
paper by J.C.Doyle and G.Stein "Multivariable
Feedback Design : Concepts for a Classical/Modern
Synthesis", IEEE Trans.AC, vol 26, no. 1, Feb
1981, pp. 4-16. The sequence of commands to do
this is
        >NY
        >FREQ
        >EDIT SVD
The singular values of a frequency response may
be plotted using the 'SVD' command. Similar
constraints on the singular values of the DY
response can also be used.

....
```

NY[1,1](frequency) >**COMPLETE**

The time domain response of RY will be displayed now. Answer YES to the question following if you are happy with the response.

[*At this stage the RY step response is displayed.*]

Were you satisfied with the response shown ? **YES**

....

4.6. EXPANDING THE SCOPE OF THE CACSD PACKAGE.

The expert system expands the scope of the CACSD package through checking on aspects of the design not explicitly covered by the specification, and effectively extending the range of specifications which may be used.

In every design method, some specifications are treated explicitly, and others implicitly. The latter can only be satisfied by judicious choice of the former, making design an art. For example, the CACSD package used deals with constraints on individual closed loop responses directly; to meet specifications on the singular values of the open or closed loop frequency responses requires careful design of the individual closed loop responses.

There may also be some specifications which a given design method cannot handle; for example the CACSD method used here cannot deal with requirements on the controller structure. A higher level expert system could be used in turn to select the most appropriate design method in view of the characteristics of the plant and specifications.

The expert system provides advice on how to meet some of the implicit constraints, and checks that the design actually satisfies them. For example, the designer may wish to impose constraints on the singular values of the open loop frequency response. These implicit constraints are treated

indirectly in the sense that they are not considered by the CACSD package when solving for a controller. However, when set, the expert system can provide advice on how the individual closed loop responses should be constrained in order to meet them. In addition, after a new controller has been computed, the expert system checks that they were satisfied. If not, advice on possible modifications to the specification to account for them is also available if desired. The example below shows the expert system checking on the constraints set on the singular values (SVD) of the open loop (GK) frequency response :

```
DY[1,1](frequency) >SUGGEST
The SVD constraints on GK have not been
satisfied. Do you want help with these ? YES
Is the problem over the low or high frequency
part of the SVD plot ? (enter '/EXPLAIN' to see
the graph) : LOW

Constraints (minimum of the minimum singular
value) on the low frequency part of the GK
response can be met by placing constraints on the
low frequency part of the DY response. In
general, it is necessary to improve the
performance of the Q2 section of the design; this
goal may also be achieved through optimization of
the DY or NY step responses.
```

4.7. OPTIMIZING THE USE OF THE CACSD SUBROUTINES.

In general terms, the commands presented to the expert system interface are translated into a sequence of calls to the numeric software. In some instances, intermediate results from previous computations may be available and convenient to use in executing the current command, thus improving the efficiency of the design package. The expert system contains knowledge of these situations, and is programmed to exploit them.

An example of this is in the computation of the controller using the SOLVE command. Firstly, only those elements of the Q matrix relating to specifications which have changed since

the previous controller computation require re-evaluation. Secondly, when sufficient memory is available, it is possible to store the constraints and optimization functions generated when the specifications are translated into the Q domain. Then, only those specifications which have changed since the previous controller computation need re-translation.

The expert system is also useful in dealing with exception conditions in the numerical software; for example, there may be a memory allocation failure when the design tool is used on large problems, or the SOLVE operation may have been interrupted prematurely by the user. The expert system is able to identify these situations, and advise the user on how best to proceed.

4.8. SUMMARY.

The most important functions of the expert system have been outlined in this chapter. The expert system forms an interface to the CACSD package, and assists both novice and experienced designers in using the design method. Based upon a database of common design features, the NEXT STEP and SUGGEST commands are able to guide and assist the user in formulating and refining the design specification, and in dealing with conflicting performance constraints. Similarly the COMPLETE command helps the user to check that the design is complete. The expert system has also been used to effectively extend the scope of the design method, as well as to integrate information from analyses of the design.

CHAPTER FIVE. IMPLEMENTATION OF THE EXPERT SYSTEM.

5.1. INTRODUCTION.

The expert system runs under CXS, a backward-chaining rule-based expert system shell for the MS-DOS operating system on IBM-PC type computers. At present the knowledge-base contains are approximately 400 rules; much of the 145k bytes of this program is text displayed to advise and guide the user.

This chapter discusses the implementation of the expert system using the facilities provided by the CXS shell.

5.2. CHOICE OF THE EXPERT SYSTEM SHELL.

The following properties were considering important in selecting an expert system shell for implementing an intelligent design system :

a. The shell should provide an efficient and effective method of communication with external, preferably memory-resident, programs. Numerical analysis algorithms are best implemented in a conventional programming language, and the expert system should be able to interact with these external programs easily.

b. The shell must be suitable for implementing non-monotonic logic systems. Being a feedback process [M2], control system design forms an inherently non-monotonic logic system [J2].

c. The shell should provide flexible knowledge representation capabilities. The nature of the multivariable problem makes database facilities attractive.

d. The shell should have flexible user interface facilities. A significant part of the expert system is devoted to interaction with the designer, and this portion should be both easy to use and program.

e. The shell must support a large knowledge base, and should execute efficiently. Control system design is a complex procedure, and thus requires a considerable knowledge base.

f. The memory requirements (RAM) of the expert system should be modest. Ideally it should be co-resident with the CACSD package in RAM.

g. The shell should support at least simple arithmetic computations.

h. The shell must run under the MS-DOS operating system, and must be very modestly priced. These two constraints are demanded by limited research funding.

None of the existing expert system shells examined for the IBM-PC (VP-Expert, Synapse, K-Shell, dmX, and Personal Consultant) satisfied all of these requirements. A solution was eventually obtained by extending the CXS expert system shell written previously by the author [T8]. Some of the special features of this shell, and how they are used in building the expert system, are detailed below.

5.3. COMMUNICATION WITH EXTERNAL PROGRAMS.

An efficient interface to external software is essential for a high-performance design system. CXS allows linking to external memory resident programs through a message passing scheme based on DOS interrupts. The external programs have access to an array of expert system variables, with elements named T0, T1, ... T99, each of which may contain symbolic or numeric values. This interface is used by the expert system to execute functions such as loading and saving transfer functions, plotting step or frequency responses, translating the specification into a quadratic programming problem, or examining the status of a performance constraint. Details of this interface are given in appendix B. A typical set of rules using this interface, in this case to plot the singular value (SVD) frequency response, is shown below.

```
IF LINE cmdline ["svd"]     ;If the SVD command was entered
THEN
  FIND ^set_response        ;Use the rule below to assign the currently selected
                            ;response to the array variables.
  T0 = "freq"               ;Always use the frequency response
  INTR 96,35,0              ;Call interrupt 96, function 35 to select the response
  INTR 96,1021,0            ;Call interrupt 96, function 1021 to plot the SVD
                            ;response
  cmd_action is "done"
END

CALC                        ;CALC = IF with no conditions
  T0 = domain
  T1 = resp                 ;Assign the currently selected response
  T2 = i_num                ;to the array variables
  T3 = o_num
  set_response is "done"
END
```

The ^ operator used in the 'FIND ^set_response' statement above signifies that a new value for the variable 'set_response' must be computed. This operation, similar to the RESET statement, is used frequently in this non-monotonic logic system.

5.4. DATABASE FACILITIES.

The expert system stores knowledge in rules and facts; in addition facts may be collected into databases, using the Prolog-style database facilities which CXS provides. Five databases are used within the expert system, and these are examined below.

5.4.1 The RESP_DB database.

This database holds the symbols corresponding to each closed loop response, and the domain, on which performance constraints may be placed. These are the RY, RU, DY, NY, and NU responses; each of these appears twice for the time and frequency domains.

This database is used internally by the expert system, mainly to access sets of responses using a common rule.

5.4.2 The SPEC_DB database.

Much of the design specification is stored in this database; the constraints on the responses are stored within the CACSD package. The records of the SPEC_DB database contain the following fields :

DOMAIN	"Time" or "frequency".
RESP	The particular closed loop response, for example "DY".
COL	The input number, 1...n
ROW	The output number, 1...n
Q	The Q matrix corresponding to this response, i.e. 1 for Q1 or 2 for Q2.
STATUS	The status of the constraints on this response : "satisfied", "active", "unsatisfied", or "unknown".
ASYM	The asymptotic value required for this response. If none, then the value "none" is stored.

OPT	The optimization weight for this response. If none, then the value "none" is stored.
RELAX	If the designer has indicated that constraints on this response cannot be relaxed, the value "no" is stored; otherwise "unknown".

The CXS shell also provides pattern matching facilities which are useful for analyzing the specification. A typical rule from the knowledge base, using this database and a pattern matching condition, is :

```
IF can_relax_some is "yes"
AND opt_q is 1
AND q1_objective is "tracking"
AND opt_col isnt "unknown"
AND FIND spec_db [=T0,"RU",opt_col,*,*,"active",*,*,"unknown"]
AND can_relax_ru isnt "no"
THEN
  sugst_opt is "done"
  DISPLAY
Use the 'EDIT' command to relax the \%T0% domain constraints on column \%opt_col% of
the RU response. This should improve the tracking performance.
END

ASK can_relax_ru ["yes","no"]
It should be possible to improve the tracking performance by relaxing the \%T0%
domain constraints on column \%opt_col% of the RU response. Would you like to try
this ?
```

This rule is used to suggest that, if the designer is prepared to relax some constraints, and wants to improve the tracking performance, it may be necessary to relax the constraints on the RU response. Note that if the FIND clause finds a record in the SPEC_DB database with fields

```
     RESP   = "RU",
     COL    = the current value of the variable opt_col,
     STATUS = "active",
and  RELAX  = "unknown",
```

then the value of the DOMAIN field is assigned to the variable T0. Also shown above is the corresponding question which is put to the user when searching for a value for the

variable 'can_relax_ru'. The sequence \%var_name% in the text displays the value of the particular expert system variable (T0 and opt_col in this example).

The expert system uses rules of the form shown above, together with knowledge of the design constraints and present performance, to analyse the specification and current state of the design. While the actual constraints on the responses are stored within the CACSD package, the expert system has efficient access to these through the message passing facility. Similarly, the performance of the design is analyzed within the CACSD package, and the results then transferred to the expert system. In some cases the designer is shown a response graph, and then asked qualitative questions about it, for example whether or not the high frequency gain is considered excessive for the particular application.

5.4.3 The SOLVE_DB database.

The status of each column of the Q1 and Q2 sections of the design are stored in this database. In particular, and for each of these columns, the database records whether or not the specification has been changed since the last SOLVE command, the result of the last SOLVE command (if it was successful, or the nature of the problem), the particular response which could not be satisfied (if any), and the likelihood of the asymptotic constraints being responsible for the conflict.

5.4.4 The PARAM_DB database.

This database records the values of various design parameters; it also contains the function code used to pass the value on to the CACSD package, and an indication of whether or not the value has been changed during the design session. At present the parameters stored are the sample time, the maximum modulus allowed for the closed loop poles, the number of points at which frequency responses are evaluated, the number of samples considered for the step

responses, and the QSTEP and NVARS parameters.

5.4.5 The FEATURE_DB database.

The records in this database are used as frames [W1] to capture knowledge relating to various design features, and have the following slots :

WANTED	The symbol to use to check whether or not the designer wants this feature. Where this feature is always required, "yes" is stored.
SPECIFIED	The symbol to use to check if the current specifications already account for this feature.
Q	The section of the design (Q1 or Q2) to which this feature corresponds.
SATISFIED	Whether or not the designer is happy with the current performance of the design with respect to this feature.
ADVISED	Whether or not this advice has already been given.
ADVICE_RULE	The rule to use to display the relevant advice.
STEP	The step number for use in the NEXT STEP module. If not to be used there, then "none".
POSSIBLE	The symbol to use to check whether or not this feature is possible. Where this feature will always be possible, "yes" is stored.
RESPONSE	The closed loop response in question.
DOMAIN	The domain (time or frequency) of the response above.
SVD	Whether or not the feature is based on the singular values of the frequency response.

At present the following design features are represented in this database. The list is by no means complete; it could be argued that no such list would ever be complete. Unfortunately the strict memory limitations of MS-DOS prevent it from being expanded much more in the present expert system. FEATURE_DB is used by the NEXT STEP, SUGGEST, and COMPLETE modules of the expert system. The design features considered are :

a. *Asymptotic properties.* If possible and desired, the design should include asymptotic tracking of command signals, asymptotic rejection of disturbances, and asymptotic decoupling. It may not always be possible to achieve this on all responses simultaneously, depending on the dc gain of the plant. In this case the designer is advised to set the asymptotic requirements on the individual responses as required.

b. *Disturbance rejection properties.* The design should ensure that low frequency disturbances are adequately rejected. To achieve this, the design should include constraints on the low frequency sections of the DY frequency responses.

c. *Control signal magnitude.* The design should limit the magnitude of the control action. This is often desirable to reduce actuator saturation problems when implemented, and generally also improves the robustness of the design. Usually the RU and NU step responses are constrained; it is also possible to achieve the same effects by constraining the high frequency sections of these responses.

d. *Optimization.* Each column of the design should contain some optimization specification, to ensure that all available degrees of freedom in the design method are used.

e. *Robustness.* To give the design some robustness to modelling errors, the singular values of the NY frequency response should not exhibit excessive gain, particularly at high frequencies. The singular value response should be constrained; constraints on the individual elements of the NY frequency response will probably be necessary to achieve this.

f. *The DY singular values.* The user should check the singular values of the DY frequency response to ensure that the design has adequate low frequency disturbance rejection. In addition and similar to the feature above, the gain at mid and high frequencies should not be excessive, for robustness considerations.

g. *The RY and NY step responses.* If these step responses do not exhibit the desired closed loop performance, they should be constrained as necessary.

h. *The NVARS and QSTEP parameters.* The values of these parameters can sometimes have a profound effect on the quality of the design. Large values of NVARS, the number of decision variables in each element of the **Q** matrices, produce better designs but increase the time required to find the solution. In general, large values of QSTEP, used in the parametrization of the **Q** matrices, suit slow control systems; the designer should experiment with different values. The user is reminded of their importance if these parameters have not been changed during the design. Otherwise it is assumed that the user has already found suitable values.

5.5. STRUCTURE OF THE EXPERT SYSTEM PROGRAM.

Figure 5.1 illustrates the overall structure of the expert system program. The command line interpreter constitutes the bulk of the program. This in turn may be divided into four main components: the help module, the suggest and next step modules, the module to explain conflicts in the design specification, and the module to check that the specification is complete. The command line interpreter also deals with commands which are to passed on to the CACSD package.

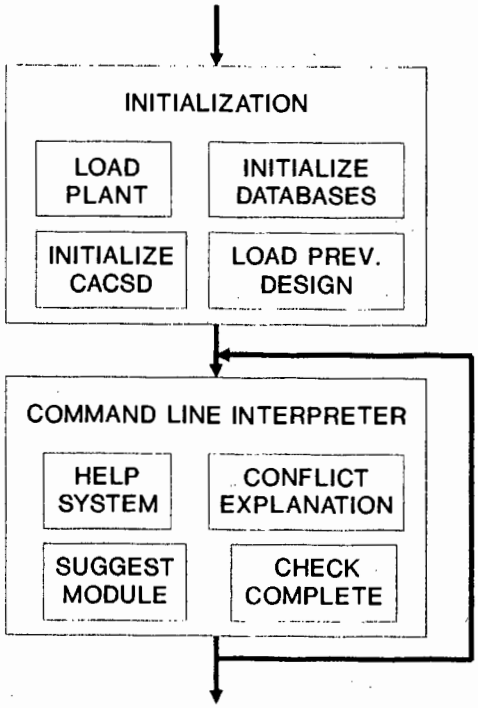


Figure 5.1. Structure of the expert system program.

5.5.1 Initialization.

The initialization process depends to a large extent on whether a completely new design is required, or the design is a continuation of some previous session.

For a new design, the user must enter the plant transfer function, and that of a nominal stabilizing controller if necessary. The expert system checks that the plant is

suitable (for example the transfer function must be strictly proper), and initializes its databases to reflect no specifications on any of the closed loop responses.

It is also possible to continue with a previously saved design. Here the relevant transfer functions are retrieved from their files, and databases are loaded with their previous values.

Some initialization is common to both cases. This includes computing the matrix fractions and the nominal step and frequency responses according to the formulae listed in table 3.2, for example. The CACSD package, which is called to perform many of the tasks mentioned above, also requires initialization.

5.5.2 The command line processor.

Pattern matching facilities, similar to those for databases, are used to analyse the user command line. For example, the 'GROUP PLOT' command, and its abbreviation 'GP', are treated by the rule

```
IF LINE cmdline["group","plot"]
OR LINE cmdline["gp"]
THEN
  INTR 96,1006,0                ;Plot the group of responses
  cmd_action is "done"
END
```

The more complex commands are treated in a similar fashion. For example, the 'OPT' command used to enter the optimization specifications, has the syntax

```
action OPT [value] [group]
```

where action is 'SET' or 'RESET',

value is the optimization weight (specified for the SET action only),

and group is 'ALL', 'DIAG', 'OFFDIAG' or 'COLUMN', if specified.

Again the pattern matching facilities provided by CXS in the form of the LINE condition are invaluable for decoding these commands. In this case, the rule

```
IF LINE cmdline[=T0,"opt"]
AND T0 isnt "set"
AND T0 isnt "reset"
THEN
  ....
END
```

could be used to check for an illegal 'OPT' command.

5.5.3 The HELP module.

The menu driven online help system, discussed in the previous chapter, is initiated by the 'HELP' or 'HELP topic' command. A useful feature of the CXS shell is that text for display purposes need not be stored in memory, but can be recalled from disk when required. This makes it possible to provide a large amount of information on a wide range of topics, even with the restrictive MS-DOS memory limitations.

Further help on questions posed to the user is also available by typing '/EXPLAIN' in response to that question.

5.5.4 The SUGGEST and NEXT STEP modules.

The SUGGEST and NEXT STEP modules are programmed similarly. The expert system stores a list of design features which an experienced designer would consider, and compares these against the current specifications and requirements of the user. In some cases, data from an analysis of the plant is also considered. Using this knowledge, the expert system can

determine whether or not a particular design feature should be added to the design specification, and advise the designer accordingly. The advice generally includes the set of commands which must be used to accomplish this.

Rules in the SUGGEST module have the generic structure shown below.

```
IF this advice has not previously been given
AND the design feature is possible with the given
      plant
AND the design feature is not accounted for by the
      current specifications
AND the feature is probably required
THEN
  Note that the advice has been given.
  Display the advice.
END
```

The NEXT STEP rules are very similar; the main difference is the step_state expert system variable which is used to record the progress of the advice given. The generic structure of these rules is

```
IF the previous steps in the design process have
      been considered
AND the design feature is possible with the given
      plant
AND the design feature is not accounted for by the
      current specifications
AND the feature is probably required
THEN
  Note that this step has been suggested.
  Display the advice.
END
```

The similar structures of the rules in these two modules can be exploited to give efficient coding. This has been

achieved using the FEATURE_DB database of design features described in section 5.4.5. Both modules have rules which search through this database to identify design features which the user should consider. For example the record

```
["limit_u_reqd", "ru_spec_ok", 1, "unknown", "no",
 "rut1", 5, "yes", "ru", "time", "no"]
```

is used to check that specifications have been placed on the control signals following a step response at input R. This in turn relates to the rules given below. A value for the variable 'limit_u_reqd' is found by asking the user. The condition clauses of the rule for 'ru_spec_ok' check that there is some element of the RU response which has constraints on neither the initial part of the step response, nor on the high frequency response; this information is obtained via the CACSD module. Finally the rule for 'rut1' is invoked to display the advice if all the conditions have been satisfied.

```
ASK limit_u_reqd ["yes","no"]
Do you want the amplitude of the control signal to be limited following a step input
at R or D ?
EXPLAIN
Constraints on the RU and NU step responses limit the control action taken following
a command input change. This in turn slows down the closed loop system. Since real
actuators cannot provide arbitrarily large control actions, it is good practice to
constrain this response; then (hopefully) the design will be less susceptible to
saturation problems when implemented.

IF FINDSOME spec ["time","RU",=T2,=T3]
AND ^low_t_max_spec isnt number        ;no constraints on maximum step response
AND ^low_t_min_spec isnt number        ;no constraints on minimum step response
AND ^high_f_spec isnt number           ;no constraints on max high frequency response
THEN
  ru_spec_ok is "no"
END

CALC
# rut1
  DISPLAY
You need some specifications which will limit the control signal at output U
following a step input at R. The best way to do this is to constrain the time
response of the RU signals; usually it is necessary to place constraints on only the
initial part of this response. Another way to achieve this is to constrain the
frequency response, in particular the high frequency portion. The sequence of
commands to use is
       >RU
```

```
        >TIME
        >EDIT ALL
END
```

In addition to the types of advice mentioned above, there are also the SUGGEST OPT and SUGGEST EDIT commands. SUGGEST OPT deals with the optimization of responses. The designer may choose to optimize either the RY response (for tracking performance) or the RU response (for low control power); a similar choice is made between NY (or DY) and NU. Given this choice, the expert system can also check that the optimization specifications do not contradict the choice of objective. Optimization of step responses generally produces better results than optimization of the corresponding frequency responses; the expert system points this out if necessary, and advises (but does not compel) the user accordingly. The command also provides advice on how the performance of the response selected for optimization may be improved further. This involves relaxing those constraints which are limiting the performance, adjusting the relative values of the optimization weights, or experimenting with the NVARS and QSTEP parameters.

A typical rule from this knowledge base, which deals with the absence of any optimization on column qlopt of the Q1 section of the design, is shown below.

```
IF qlopt is number                ;No optimization has been specified for column qlopt
AND q1_objective is "tracking"
THEN
  sugst_opt is "done"
  DISPLAY
I suggest that you use the following sequence of commands to set optimization for column
\%qlopt% of the Q1 part of the design.
        >RY
        >TIME
        >\%qlopt% \%qlopt%
        >SET OPT 1 COLUMN
        >SET OPT 0.1
This sets an optimization weight of 0.1 for the element on the diagonal, and a weight of 1
for the others. You may need to refine this ratio of weights to tradeoff tracking response
versus cross-coupling performance. Increasing the weight of the diagonal elements will
improve the tracking performance at the expense of worse decoupling, and vice-versa.
END
```

SUGGEST EDIT gives information about the currently selected response, indicating its relevance to the final design, and discussing why the designer may wish to consider placing constraints on it. A typical example of this rule, dealing with the case when the DY frequency response is currently selected, is

```
IF resp is "DY"
AND domain is "freq"
THEN
  sugst_edit is "done"
  DISPLAY
The DY response relates to the closed loop system's ability to reject disturbances at
input D. The designer should ensure that low frequency disturbances are adequately
suppressed; -20 dB is a typical requirement for this range. At mid frequencies the
designer must ensure that the responses do not peak excessively; anything over +3 dB is
cause for concern. Robustness constraints on the singular values may also be set; use the
'EDIT SVD' command for this.

In general the ability to reject high frequency disturbances is limited. The gain of the
diagonal elements should be no more than about +3 dB over this range; this can also be
reduced by constraining the high frequency response of the diagonal elements of NY. The
responses of the off-diagonal elements will depend strongly on the high frequency behavior
of the plant, but should generally be less than 0 dB. A good way to get reasonable
behavior of this high frequency range is by limiting the gain of the high frequency
portion of the NU response.

Note that the NY and DY step responses are closely related by the equation
     NY = I - DY
where I is the identity matrix; constraints on the off-diagonal elements of these
responses are therefore completely equivalent.
END
```

5.5.5 The module checking for completeness.

The purpose of this module is to check that the design is complete in the sense that a list of design aspects have been considered. The structure of rules in this module, as shown below, is similar in many ways to those of the SUGGEST and NEXT STEP modules. Here again the design specifications are compared against a list of common design features in the FEATURE_DB database, and the user advised accordingly. The rules in this module have the generic form shown below.

```
IF the design feature is not accounted for by the
      current specifications
AND the design feature is possible with the
      given plant
AND the feature is probably required
AND the designer is not happy with the current
      performance
THEN
  Display advice on how to update the specification.
END
```

5.5.6 The module explaining specification conflicts.
Conflicts in the specification are detected at two levels by the CACSD package, in both cases as conflicts in the set of linear constraints generated for the particular design. The expert system has the task of explaining these conflicts, and of suggesting ways for dealing with them. The expert system also deals with other complications which preclude a solution to the design; for example there is the possibility of insufficient RAM memory being available for large design problems. In this instance the user is advised on ways to reduce the memory requirements.

Some conflicts are detected when the design specification is translated into linear constraints. These conflicts relate to linear constraints in the form

$$\mathbf{a}^T\mathbf{x} \leq \alpha < 0,$$

with $\mathbf{a} = 0,$

which cannot be satisfied by any vector $\mathbf{x}$. In the time domain this situation typically arises with a rise time specification faster than the plant dead time. In the frequency domain this type of conflict domain results from poles or zeros of the plant on the unit circle fixing the system response at those frequencies. Similar conflicts also arise occasionally in the asymptotic specifications.

When the CACSD package detects this type of conflict, the expert system is notified of the response concerned. The messages displayed to the user, for the time and frequency domain cases respectively, are shown below. In each case the designer must relax or remove the offending specification.

> It is not possible to satisfy the specifications on the RY[1,1] time domain response, due to the characteristics of the plant (usually the plant dead time). You will have to relax the specifications on this response.
>
> It is not possible to satisfy the specifications on the DY[2,1] frequency domain response, due to the characteristics of the plant (generally a pole or zero of the plant on the unit circle). You will have to relax the specifications on this response. It may be possible avoid this problem by placing the specifications over ranges which exclude this pole or zero frequency.

Most conflicts are detected when QPSOL, the quadratic programming algorithm, finds no feasible solution to the quadratic programming problem. To facilitate analysis of these conflicts, the linear constraints are grouped according to the design specifications that they represent. Thus all constraints relating to specifications on the RY[2,2] step response will be in one group, and all those on the NU[1,2] frequency response will be in another. QPSOL tackles phase I of the quadratic programming problem by finding a vector **x** which satisfies all constraints in the first group, then one which satisfies the constraints in the first two groups, and so on, until all groups have been considered, or a conflict has been found.

The expert system has access to the status of each of these groups following a call to QPSOL; this information is requested from the CACSD package and stored in the SPEC_DB database. The linear constraints are given a status of active, satisfied, or not satisfied, according to the terminology of the active set method used (3.x). Consider the constraint

$$c(\mathbf{x}) \leq \alpha.$$

This constraint, when evaluated at $\mathbf{x}_1$, is assigned a status according to table 5.1.

$c(\mathbf{x}_1)$	status
$= \alpha$	active
$\leq \alpha$	satisfied
$> \alpha$	not satisfied

Table 5.1. Status of constraint $c(\mathbf{x}_1)$.

Note that only those constraints appearing in the active set (3.5) are considered active, and that the active status implies that it is also satisfied. Thus it is possible that some constraints may be met with equality, but not be considered active. The status of a group is determined from the status of the corresponding linear constraints, using the following rules :

1. If the group has not yet been considered by QPSOL, then its status is unknown.
2. If any linear constraints are not satisfied, then the group is not satisfied.
3. If any linear constraints are active, then the group is active.
4. The status is satisfied.

The expert system uses this data to inform the designer of the conflict. A typical message, generated by the expert system to describe this conflict, is shown below :

```
The frequency domain constraints on DY[2,1] could
not be satisfied as they conflict with :
   the frequency domain constraints on DY[1,1]
   the time domain constraints on NU[1,1]
Of these, the Lagrange multipliers hint that the
frequency domain constraints on DY[1,1] are the
most probable cause of the conflict.
It is unlikely that the asymptotic constraints are
responsible for the conflict.
```

The likelihood of the asymptotic constraints contributing to the conflict, as well as the specification most probably causing the conflict, is determined by analyzing the Lagrange multipliers computed in the QPSOL algorithm. Although subject to scaling, the more positive the value of the multiplier, the more likely that constraint is a limiting factor in the design. The inverse argument is commonly used active set methods, where the constraint with the most negative multiplier is assumed to have the least impact on the solution, and is deleted from the active set.

In addition to explicitly describing the cause of the conflict as shown above, the expert system can also provide the user with some assistance in resolving the conflict. One way this is done is by diagnosing some particular conflicts. For example, given the relationship between the NY and DY responses, specifications placed on both responses may conflict. This is dealt with by rules of the form shown below; the example given detects a specification of less than unity gain at low frequencies on a diagonal element of NY, when asymptotic disturbance rejection has been specified on the corresponding element of DY.

```
IF spec.resp is "NY"
AND spec.col is spec.row
AND spec.domain is "freq"
AND FIND spec_db ["freq","DY",spec.col,spec.row, *,*,number]
AND ^ny_dy_asym_clash is "yes"
THEN
  sugst_conflict is "done"
  DISPLAY
You must change the specifications on the frequency response of
NY[\%spec.row%,\%spec.col%], or remove the asymptotic rejection specification on
DY[\%spec.row%,\%spec.col%], to make them consistent. Of these, the constraints
on the NY response seem strange to me.
END

CALC
  T0 = "freq"
  T1 = "NY"
  T2 = spec.col
  T3 = spec.row
  INTR 96,35,0                ;select NY[i,i](freq)
  T5 = 0
  T6 = 0.004
```

```
  INTR 96,2025,1              ;get the minimum value of the constraints
                              ; at low frequencies
  IF T8 < 1
    ny_dy_asym_clash is "yes"
  END
END
```

Sometimes it is possible that a better choice of the NVARS and/or QSTEP design parameters could resolve the conflict. The CACSD method uses these parameters in the parameterization of the Q matrices, which can influence the design significantly. In this case the designer is advised on how the parameters may be changed.

Further advice is also available on resolving conflicts in the specifications on the singular values of the frequency responses. These specifications are not translated into equivalent linear constraints, but the expert system can provide advice on how to modify the other closed loop specifications in order to meet them. For example, the rule below treats conflicts in specifications on the high frequency region of the open loop (GK) frequency response.

```
IF svd_resp_c is "GK"
AND gk_svd_prob is "high"          ;the problem is with the high frequency part of the
                                                          ;response
THEN
  sugst_svd is "done"
  DISPLAY
Constraints (maximum of the maximum singular value) on the high frequency part of the GK
response can be met by placing constraints on the high frequency part of the NY response.
In general, it is necessary to slow the performance of the Q2 section of the design, thus
making it more robust. Another way to achieve this is by placing constraints on the high
frequency part of the NU response.
END
```

5.6. SUMMARY.

The issues involved in implementing the expert system have been described in this chapter. These include the selection of an appropriate expert system shell, communication with the CACSD package, and the use of databases to store common design features and the status of the design. The overall

structure of the expert system has been presented, as well as details of the rules used in the NEXT STEP, SUGGEST, and COMPLETE modules. It has also been shown how the active set and Lagrange multipliers of the quadratic programming problem may be analysed to identify the cause of conflicts within the design specification, and to assist the user in dealing with these.

CHAPTER SIX. SAMPLE DESIGN SESSIONS.

6.1. MINE MILLING PLANT.

The expert system has been applied to a model of a gold mine milling unit [H3]. The z domain transfer function of the plant, with a normalized sampling rate of 1 Hz, is given below.

$$\begin{bmatrix} \dfrac{0.6512}{z-0.9031} & \dfrac{1.190}{z-0.05306} & \dfrac{0.0503}{z-0.7304} \\ \dfrac{7.0370}{z-0.9130} & \dfrac{-3.296}{z-0.08696} & \dfrac{0.2055}{z-0.7594} \\ \dfrac{-0.8206}{z-0.8436}\,z^{-5} & \dfrac{0.5127z-0.3876}{z^2-0.5837z+0.07420}\,z^{-1} & \dfrac{0.2932z-0.2640}{z^2-0.6603z+0.01052} \end{bmatrix}$$

The design for this plant is carried out using the expert system for guidance. A transcript of the design session is given below. Commands and answers entered by the user are shown in **BOLD** type. Comments added to this transcript are printed in *italics* and enclosed in square brackets.

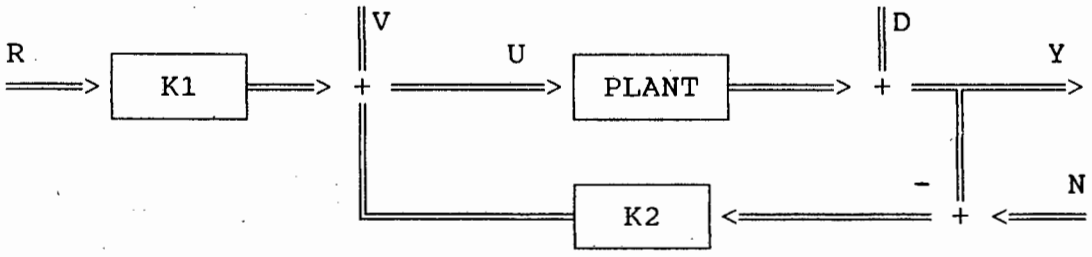
Welcome to MV-CXS, the Expert System design tool for linear multivariable control system design.

The file MILL.GZ has been found.
Do you wish to load the plant transfer function ? **YES**
The plant transfer function has been loaded from file MILL.GZ
Do you wish to modify the plant transfer function ? **NO**

[For the purposes of demonstration this design will be started from scratch. Therefore the two questions following are answered with NO.]

Do you want to load the previous design ? **NO**
The file MILL.PRM has been found.
Do you want to load the previous design parameters from it ? **NO**
Initializing the databases.....
Checking that individual elements of the plant transfer matrix are coprime and strictly proper...
Ok
What is the sampling time (seconds) ? **1**
The plant poles have a maximum modulus of 0.913. Is this maximum modulus acceptable for the closed loop poles ? **YES**
Would you like help in setting up a specification ? **YES**

The following block diagram illustrates the type of system under consideration. You will need to remember the meanings of the inputs labeled R, N, D and V, and the outputs Y and U.



For more help with the design system, type 'HELP'. For help on selecting a response to work on, type 'HELP SELECT'. Remember that you may also type '/EXPLAIN' in response to a question if you need assistance.

Finally, to help you further the command 'SUGGEST' can often offer advice on what to do next, and the 'NEXT STEP' command can help you get started with a design.

NY[1,1](time) >**NEXT STEP**

The step-by-step design mode has been selected. This will give you recommendations on what you should do next; it is then up to you to enter the commands if you wish. Note that any of the other commands can also be used; for example you may examine the various responses using the 'PLOT' command. Once you have completed the step, the 'NEXT STEP' ('NS' for short) command should be used once more to recommend the following step to be taken.

Checking plant DC gain.
A steady state value of at least 8.369168 will be required at some input to the plant to achieve asymptotic tracking and decoupling following a unit step input. Is this value acceptable ? **YES**
Would you like asymptotic rejection of disturbances at input D ? **/EXPLAIN**
Asymptotic rejection of disturbances at input D
means that, at DC, the frequency response matrix Hdy is zero. Thus the disturbances will be rejected completely, and there will be no steady state cross-coupling (interaction). Usually this is desired; unfortunately it sometimes requires prohibitively large control signal inputs to the plant. If you think you might like asymptotic rejection, answer YES.

Would you like asymptotic rejection of disturbances at input D ? **YES**

It is possible to achieve asymptotic rejection of disturbances at input D.

You can enter asymptotic disturbance rejection requirements using the 'SET ASYM REJECTION' command. If desired, this specification may be lifted later using the 'RESET ASYM REJECTION' command. For more information on the asymptotic specifications, type 'HELP ASYM'.

DY[1,1](frequency) >**SET ASYM REJECTION**
DY[1,1](frequency) >**NEXT STEP**
Would you like asymptotic tracking of the command signal R ?
YES

You can enter asymptotic tracking/decoupling requirements using the 'SET ASYM TRACKING' command. If necessary, this specifications may be lifted later using the 'RESET ASYM TRACKING'. For more information on the asymptotic specifications, type 'HELP ASYM'.

RY[1,1](frequency) >**SET ASYM TRACKING**
RY[1,1](frequency) >**NEXT STEP**
Do you require rejection of low frequency disturbances at input D ? **YES**

Rejection of low frequency disturbances has not been completely specified. You should use the 'EDIT' command to enter constraints on the low frequency response of the DY signals to ensure reasonable disturbance rejection properties for the closed loop system. The sequence of commands to use is

>DY
>FREQ
>EDIT ALL

A starting constraint could be -20 dB over the range 0 to 0.0025 Hz.

DY[1,1](frequency) >**EDIT ALL**
[*Using the spreadsheet style constraint editor shown below, the maximum value of this response is constrained to no more than -20 dB over the range 0 to 0.0025 Hz.*]

Frequency domain constraints : DY[1,1] max = 0.5 Hz

Maximum

Value (db)	From (Hz)	To (Hz)
-20	0	0.0025

Select using [Ctrl] cursor keys : ESC to exit

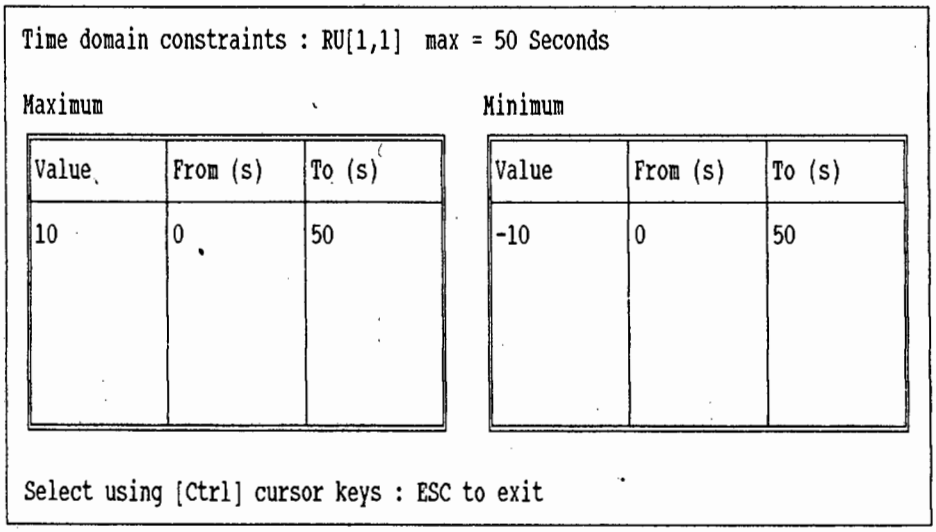
DY[1,1](frequency) >**NEXT STEP**
Do you want the amplitude of the control signal to be limited following a step input at R or D ? **YES**

You need some specifications which will limit the control signal at output U following a step input at R. The best way to do this is to constrain the time response of the RU signals; usually it is necessary to place constraints on only the initial part of this response. Another way to achieve this is to constrain the frequency response, in particular the high frequency portion. The sequence of commands to use is

>RU
>TIME
>EDIT ALL

RU[1,1](time) >**EDIT ALL**

[*Using the constraint editor, the value of the control signal at U is constrained to lie between -10 and 10 units over the range 0 to 50 seconds, following a unit step at input R.*]



RU[1,1](time) >**NEXT STEP**

You need some specifications which will limit the control signal at output U following a step input at N. The best way to do this is to constrain the time response of the NU signals; usually it is necessary to place constraints on only the initial part of this response. Another way to achieve this is to constrain the frequency response, in particular the high frequency portion. The sequence of commands to use is

>NU
>TIME
>EDIT ALL

NU[1,1](time) >**EDIT ALL**

[*Using the constraint editor, the value of the control signal at U is constrained to lie between -10 and 10 units over the range 0 to 50 seconds, following a unit step at input N.*]

NU[1,1](time) >**NEXT STEP**

You should now use the 'SOLVE' command to take the changes to the specification into account.

NU[1,1](time) >**SOLVE**
Solving for column 1 of Q1.

```
Solving for column 2 of Q1.
Solving for column 3 of Q1.
Solving for column 1 of Q2.
Solving for column 2 of Q2.
Solving for column 3 of Q2.
NU[1,1](time) >NEXT STEP
```

The optimization specifications are not yet complete. Use the 'OPT' commands to enter your optimization requirements. For more information on these, type 'HELP OPT'. The 'SUGGEST OPT' command should be able to help you with this, and will be started for you now.

I assume that you want to optimize the command tracking, disturbance rejection, and decoupling properties of the closed loop system.

I suggest that you use the following sequence of commands to set optimization for the Q1 part of the design.

```
>RY
>TIME
>SET OPT 1 ALL
>SET OPT 0.1 DIAG
```

This sets an optimization weight of 0.1 for the diagonal elements, and a weight of 1 for the others. You may need to refine this ratio of weights to tradeoff the tracking response versus the cross-coupling performance. Increasing the weight of the diagonal elements will improve the tracking performance at the expense of worse decoupling, and vice-versa.

```
NU[1,1](time) >RY
RY[1,1](time) >SET OPT 1 ALL
RY[1,1](time) >SET OPT 0.1 DIAG
RY[1,1](time) >NEXT STEP
```

You should now use the 'SOLVE' command to take the changes to the specification into account.

```
RY[1,1](time) >SOLVE
Solving for column 1 of Q1.
Solving for column 2 of Q1.
Solving for column 3 of Q1.
RY[1,1](time) >NEXT STEP
```

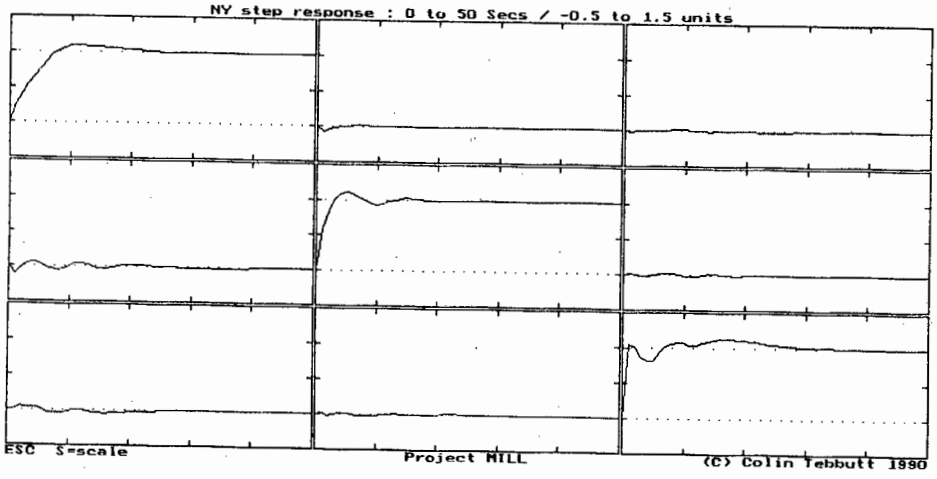
The optimization specifications are not yet complete. Use the 'OPT' commands to enter your optimization requirements. For more information on these, type 'HELP OPT'. The 'SUGGEST OPT' command should be able to help you with this, and will be started for you now.

I suggest that you use the following sequence of commands to set optimization for the Q2 part of the design.

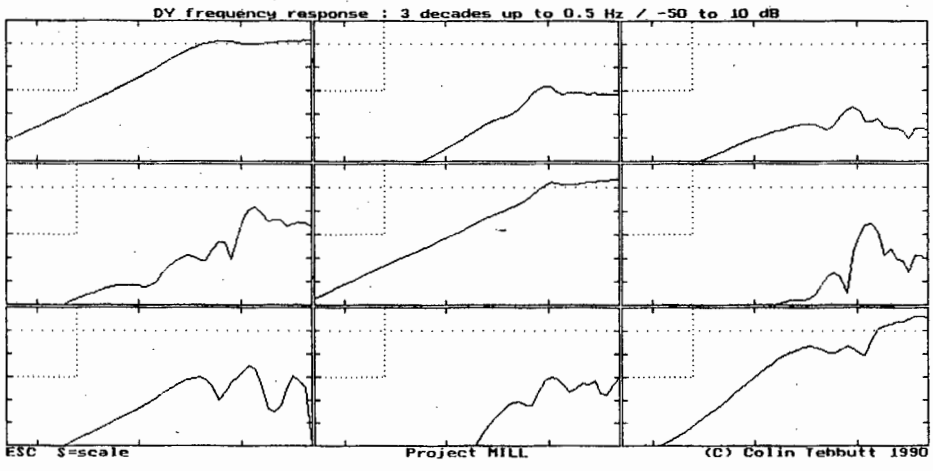
```
>NY
>TIME
>SET OPT 1 ALL
>SET OPT 0.1 DIAG
```

This sets an optimization weight of 0.1 for the diagonal elements, and a weight of 1 for the others. You may need to refine this ratio of weights to tradeoff tracking response versus cross-coupling performance. Increasing the weight of the diagonal elements will improve the tracking performance at the expense of worse decoupling, and vice-versa.

```
RY[1,1](time) >NY
NY[1,1](time) >SET OPT 1 ALL
NY[1,1](time) >SET OPT 0.1 DIAG
NY[1,1](time) >SOLVE
Solving for column 1 of Q2.
Solving for column 2 of Q2.
Solving for column 3 of Q2.
NY[1,1](time) >GROUP PLOT
```



NY[1,1](time) >**DY**
DY[1,1](time) >**FREQUENCY**
DY[1,1](frequency) >**GROUP PLOT**



DY[1,1](frequency) >**NEXT STEP**

The 'NEXT STEP' sequence of step is complete. Now you may wish to look at some of the other performance specifications; the 'SUGGEST' command could possibly give you some further advice. If you wish to know more about tailoring a particular response, select that response and then type 'SUGGEST EDIT'.

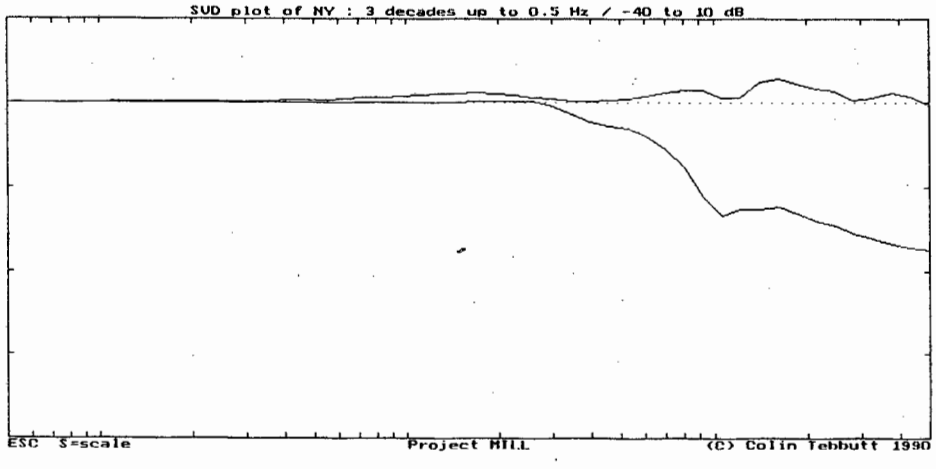
DY[1,1](frequency) >**SUGGEST**
Do you require a robust design ? **/EXPLAIN**
The robustness of a design is a measure of its ability to handle inaccuracy in the plant model. Since every model of a real plant contains some inaccuracy, every control system should have some measure of robustness. So in short, answer YES!
Do you require a robust design ? **YES**

To ensure some robustness of the design, the SVD plot of the NY frequency response should not show any excessive peaking, and the high frequency gain should taper off. Robustness specifications can be entered as constraints on the maximum singular values of the NY frequency response. Details of this approach can be found in the paper by J.C.Doyle and G.Stein "Multivariable Feedback Design : Concepts for a Classical/Modern Synthesis", IEEE Trans AC, vol 26, no. 1, Feb 1981, pp. 4-16. The sequence of commands to do this is

```
>NY
>FREQ
>EDIT SVD
```

The singular values of a frequency response may be plotted using the 'SVD' command. Similar constraints on the singular values of the DY response can also be used.

NY[1,1](frequency) >**SVD**



NY[1,1](frequency) >**EDIT SVD**

[*The following constraints are entered on the singular values of the NY frequency response :*

SVD ≤ +3 dB, 0 ≤ f < 0.1 Hz,

SVD ≤ -3 dB, 0.1 ≤ f < 0.2 Hz,

SVD ≤ -6 dB, 0.2 ≤ f < 0.5 Hz.]

Would you like advice on how to satisfy these specifications ? **YES**

It is necessary to place/tighten constraints on the individual elements of the NY frequency response in order to meet the SVD constraints on the NY response, over the required range of frequencies. Note that usually the gains of the off-diagonal should be lower than those of the diagonal elements. If there are some elements with relatively high gains, try constraining these first.

NY[1,1](frequency) >**EDIT ALL**

[*The maximum value of the NY frequency response is constrained to no more than -20 dB over the range 0.1 to 0.5 Hz.*]

NY[1,1](frequency) >**SOLVE**

Solving for column 1 of Q2.

Solving for column 2 of Q2.

Solving for column 3 of Q2.

The frequency domain constraints on DY[3,3] could not be satisfied as they conflict with :

the frequency domain constraints on DY[2,3]

the frequency domain constraints on NY[1,3]

the frequency domain constraints on NY[2,3]

the frequency domain constraints on NY[3,3]

Of these, the Lagrange multipliers hint that the frequency domain constraints on NY[3,3] are the most probable cause of the conflict. It is unlikely that the asymptotic constraints are responsible for the conflict.

NY[1,1](frequency) >**EDIT ALL**

[The constraints on the NY frequency response are revised as follows :

NY(f) ≤ -6 dB, 0.1 ≤ f < 0.2 Hz,

NY(f) ≤ -12 dB, 0.2 ≤ f < 0.5 Hz.]

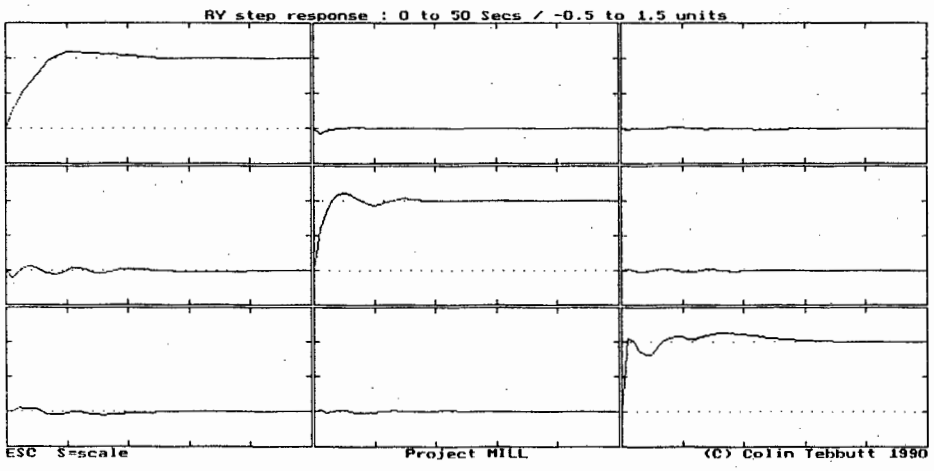
NY[1,1](frequency) >**SOLVE**

Solving for column 1 of Q2.
Solving for column 2 of Q2.
Solving for column 3 of Q2.

Checking the SVD constraints for section Q2...
All of the SVD constraints have been satisfied.

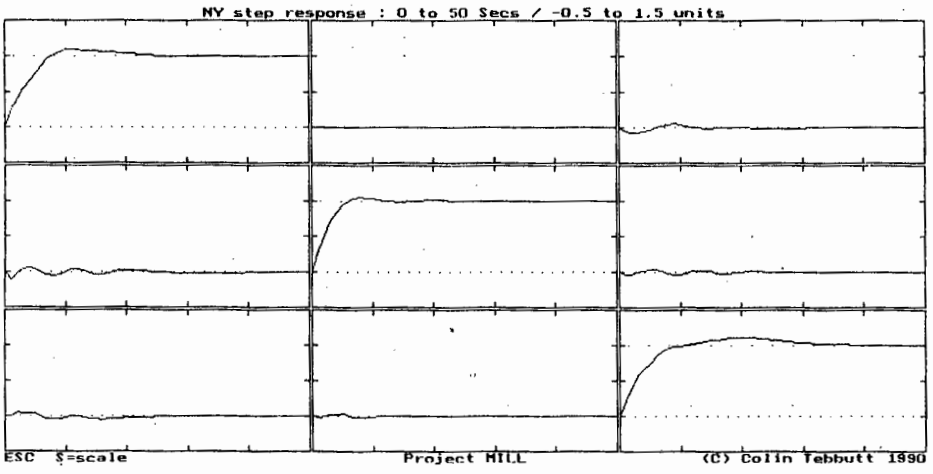
NY[1,1](frequency) >**SUGGEST**

The time domain response of RY will be displayed now.
Answer YES to the question following if you are happy with the response.



Were you satisfied with the response shown ? **YES**

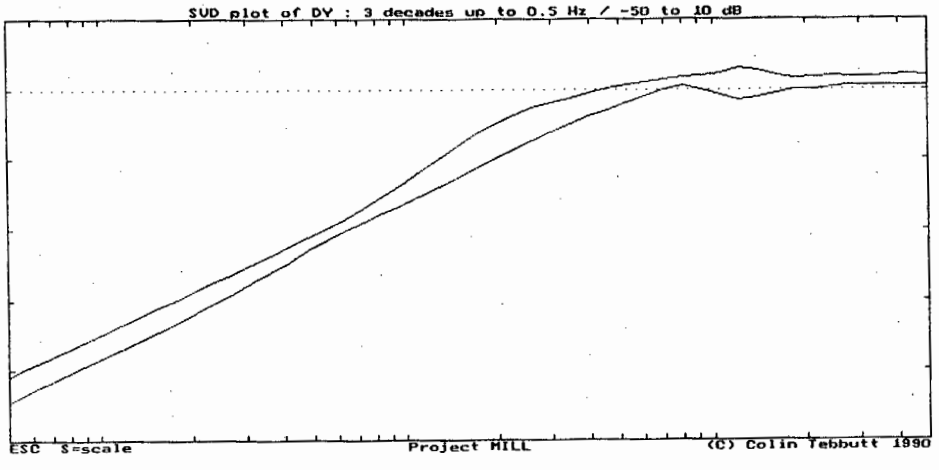
The time domain response of NY will be displayed now.
Answer YES to the question following if you are happy with the response.



Were you satisfied with the response shown ? **YES**

To ensure good rejection of low frequency disturbances, the low frequency gain of the DY response should be small. In addition, the SVD plot of the DY frequency response should not show any excessive peaking.

The frequency domain response of DY will be displayed now. Answer YES to the question following if you are happy with the response.



Were you satisfied with the response shown ? **YES**

You can try experimenting with the value of the QSTEP parameter. Larger values for QSTEP suit slower control systems, and smaller values suit fast systems. The present value of QSTEP is 5, and this may be changed using the 'QSTEP n' command. Type 'HELP QSTEP' for more information.

NY[1,1](frequency) >**HELP QSTEP**

The parameter QSTEP is used internally in the numeric software. The usual range is from 1 to about 20; usually the upper limit should satisfy the inequality QSTEP*NVARS <= 100. The larger the value of QSTEP, the slower the control system response tends to be.

For stable plants, the control signal will over the first QSTEP*NVARS sample periods only; thus the combination of QSTEP and NVARS describe a time window over which the control action takes place. Therefore QSTEP and NVARS should be chosen to adequately cover this time window. For unstable plants the interpretation of these parameters is more complex, but the same principles should be applied. Note that the QSTEP and NVARS parameters should be dealt with as a unit; the product QSTEP*NVARS gives the time window, and NVARS the number of subdivisions in this window. For help with NVARS, type 'HELP NVARS'.

The present value of QSTEP may be found by typing 'QSTEP', and the value can be changed using the command 'QSTEP n'. Note that the responses are re-initialized after QSTEP is changed, and so you must use the SOLVE command to find the new controller.

NY[1,1](frequency) >**HELP NVARS**

NVARS is the number of decision variables used in each element of Q in the search algorithm. The higher the value, the better the solution will be, but the longer it will take to find the solution. The maximum value must not exceed 8;

the minimum useful value is 2, but generally a value of 5 is reasonable.

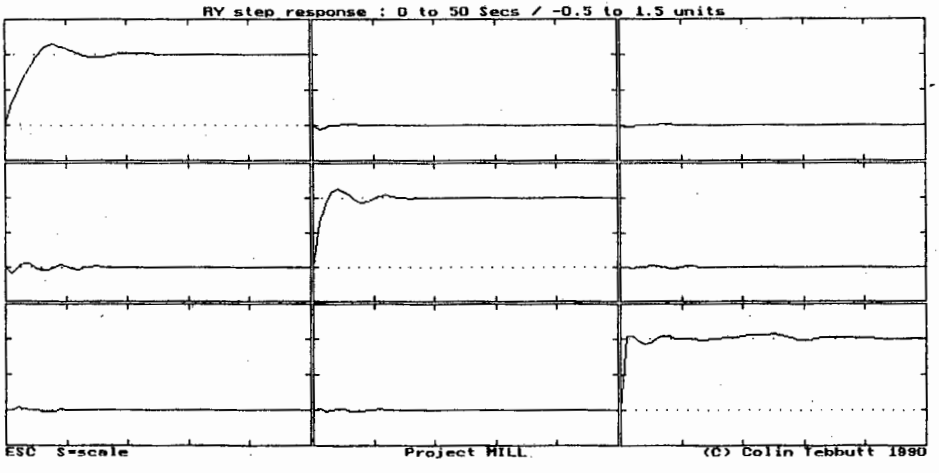
For stable plants, the control signal will over the first QSTEP*NVARS sample periods only; thus the combination of QSTEP and NVARS describe a time window over which the control action takes place. Therefore QSTEP and NVARS should be chosen to adequately cover this time window. For unstable plants the interpretation of these parameters is more complex, but the same principles should be applied. Note that the QSTEP and NVARS parameters should be dealt with as a unit; the product QSTEP*NVARS gives the time window, and NVARS the number of subdivisions in this window. For help with QSTEP, type 'HELP QSTEP'.

The present value of NVARS may be found by typing 'NVARS', and the value can be changed using the command 'NVARS n'. After changed NVARS, you should use the SOLVE command to find the new controller.

```
NY[1,1](frequency) >NVARS 8
The closed loop responses have been re-initialized. You
should use the SOLVE command to find a new controller.

NY[1,1](frequency) >QSTEP 4
The closed loop responses have been re-initialized. You
should use the SOLVE command to find a new controller.

NY[1,1](frequency) >SOLVE
Solving for column 1 of Q1.
Solving for column 2 of Q1.
Solving for column 3 of Q1.
Solving for column 1 of Q2.
Solving for column 2 of Q2.
Solving for column 3 of Q2.

Checking the SVD constraints for section Q2...
All of the SVD constraints have been satisfied.
```

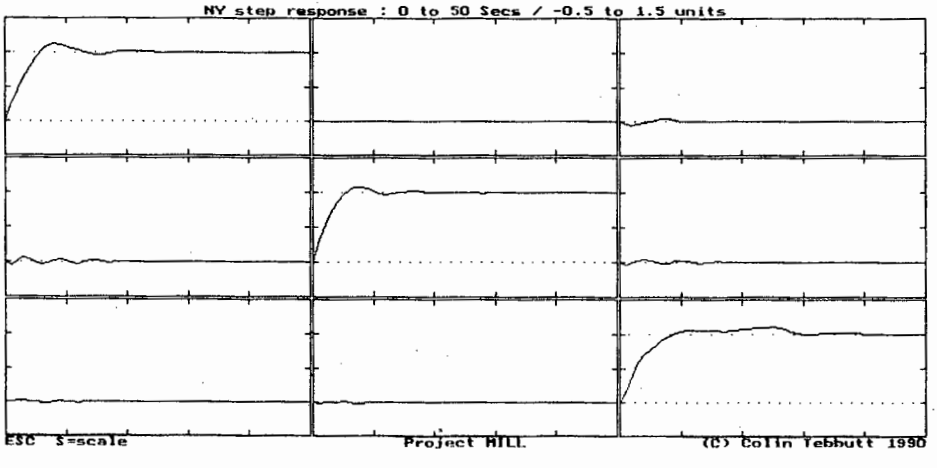
NY[1,1](frequency) >**COMPLETE**

The time domain response of RY will be displayed now. Answer YES to the question following if you are happy with the response.



Were you satisfied with the response shown ? **YES**

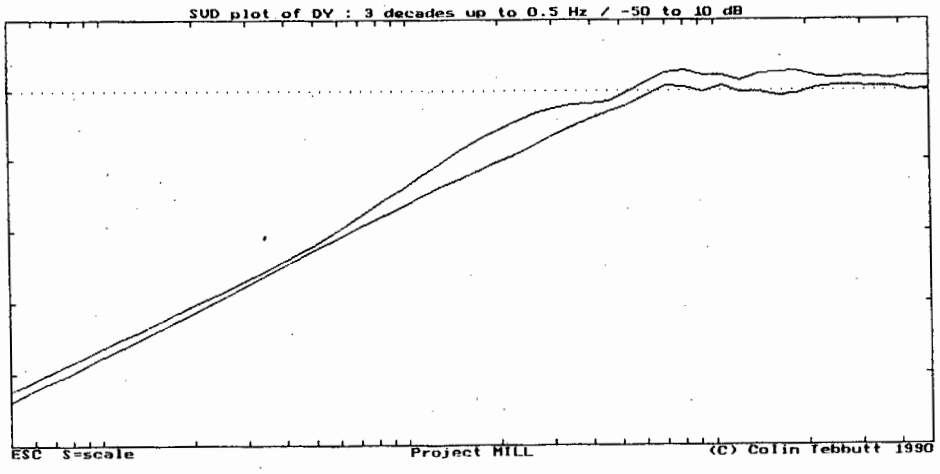
The time domain response of NY will be displayed now. Answer YES to the question following if you are happy with the response.



Were you satisfied with the response shown ? **YES**

To ensure good rejection of low frequency disturbances, the low frequency gain of the DY response should be small. In addition, for robustness of the design, the SVD plot of the DY frequency response should not show any excessive peaking.

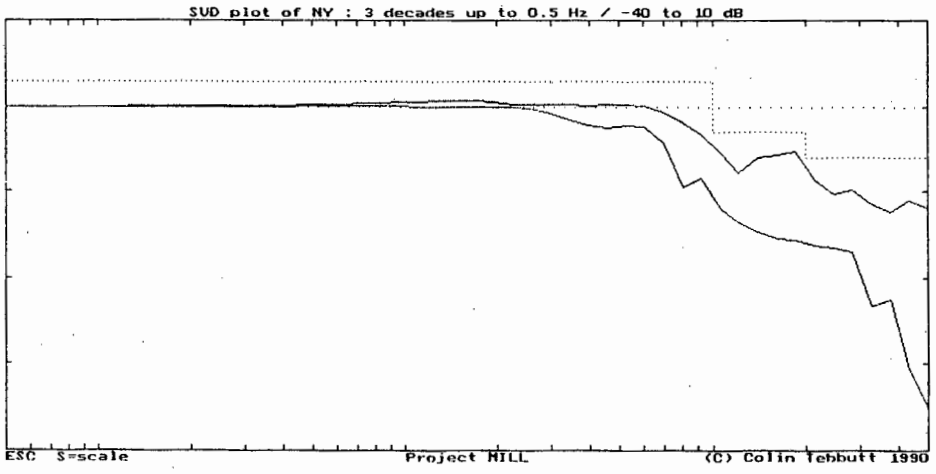
The frequency domain response of DY will be displayed now. Answer YES to the question following if you are happy with the response.



Were you satisfied with the response shown ? **YES**

It seems to me that your design is complete; if you are satisfied with it, use the 'SAVE' command to save it, and then the 'EXIT' command to leave the design session.

NY[1,1](frequency) >**SVD**



NY[1,1](frequency) >**SAVE**
NY[1,1](frequency) >**EXIT**
Do you want a report on the current design ? **NO**

[*The SAVE command saves the current status of the design for implementation or a possible future design session. The report option, which was not selected, produces a printout of the current specification.*]

6.2. GYROSCOPE.

The 2-gimbal gyro studied by Limebeer and Maciejowski [L2] is used to illustrate the use of the design system when the plant is not stable. The dialogue with the expert system to engineer a design specification would be quite similar to that given above; it is within the CACSD package that most of the differences between stable and unstable plants are encountered. Hence this section will concentrate on the solution of the control system design problem given a set of performance specifications.

The z domain transfer function of the plant, when sampled at 50 Hz, is

$$\frac{0.001}{d(z)}\begin{bmatrix} 2.4240z^2 - 1.2298z - 0.8155 & 0.7815z^2 + 2.9061z + 0.2076 \\ -0.7835z^2 - 2.9134z - 0.2082 & 0.5170z^2 - 0.2207z - 0.1680 \end{bmatrix}$$

where $d(z) = z^3 - 0.4359z^2 + 0.2384z - 0.8025$.

Thus the plant has an unstable pole at $z = 1$, and lightly damped poles at $z = -0.2820 \pm j0.8503$. The nominal controller used to stabilize the plant was

$$K0(z) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Applying the diagonal factorization method gives

$$\tilde{D}(z) = \frac{z - 1}{z - 0.95}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

and $\tilde{N}(z) = \tilde{D}(z)G(z)$.

Theorem 2 of chapter 2 can be used to show that this factorization is coprime.

Only the K2 section of the design will be considered below; similar performance may be achieved for command inputs at R. The following constraints were set to limit the control effort, provide robustness for the design, and ensure adequate rejection of low frequency disturbances :

$$| h_{NU}(kT) | \leq 50 \qquad 0 \leq k \leq 50$$

$$| H_{NY}(e^{j2\pi fT}) | \leq 0.1 \qquad 10 \leq f \leq 25 \text{ Hz}$$

$$| H_{DY}(e^{j2\pi fT}) | \leq 0.1 \qquad 0 \leq f \leq 0.1 \text{ Hz}$$

$$| H_{DY}(e^{j2\pi fT}) | \leq \sqrt{2} \qquad 0.5 \leq f \leq 25 \text{ Hz}$$

The cost function to be minimized was based on the time domain response to unit step at the disturbance input D. This in turn gives good decoupling and command tracking performance. For column j the cost is given by

$$J_j = \sum_{k=0}^{50} \sum_{i=1}^{2} \left[a_{i,j} (h_{DY}(kT))^2 \right], \quad j \in \{ 1, 2 \}$$

where

$$a_{i,j} = \begin{cases} 0.1 & i = j \\ 1 & i \neq j \end{cases}$$

Elements in the Q matrix were approximated as discussed in chapter 2; each had ten coefficients, and the QSTEP parameter was set at four. In figure 6.3 it can be seen that the disturbance rejection constraint has not been met exactly; this is mainly the result of evaluating frequency responses at a finite number of points (50 in this case) on the unit circle. Representing the frequency constraints as linear constraints also gives rise to a small approximation error [B6].

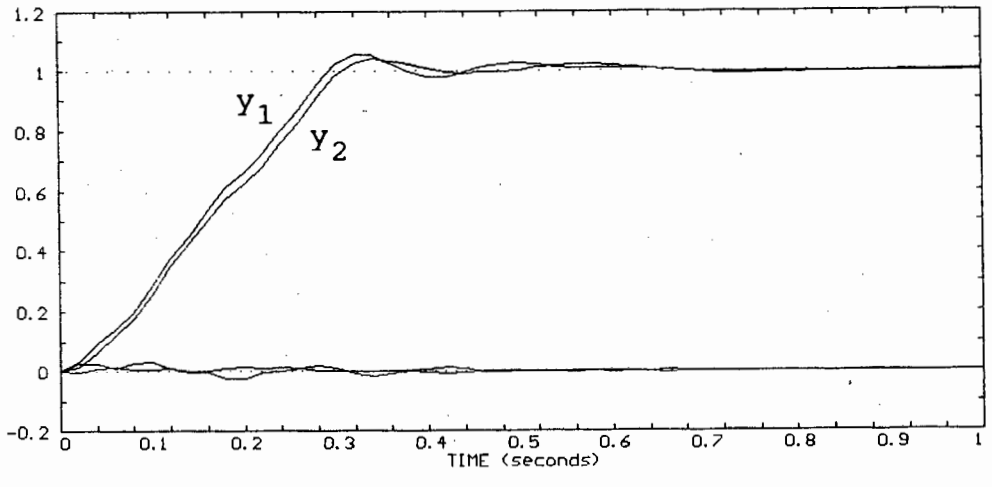


Figure 6.1. h_{NY} step response.

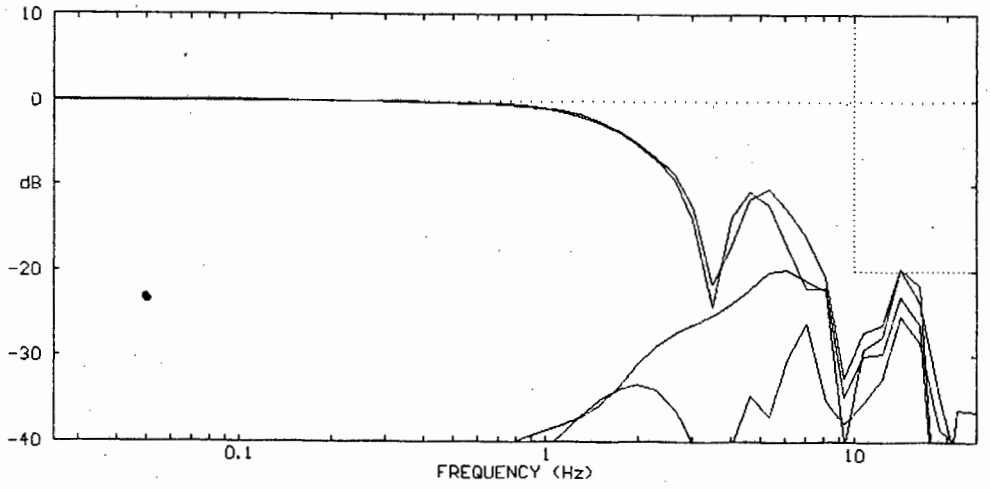


Figure 6.2. H_{NY} frequency response.

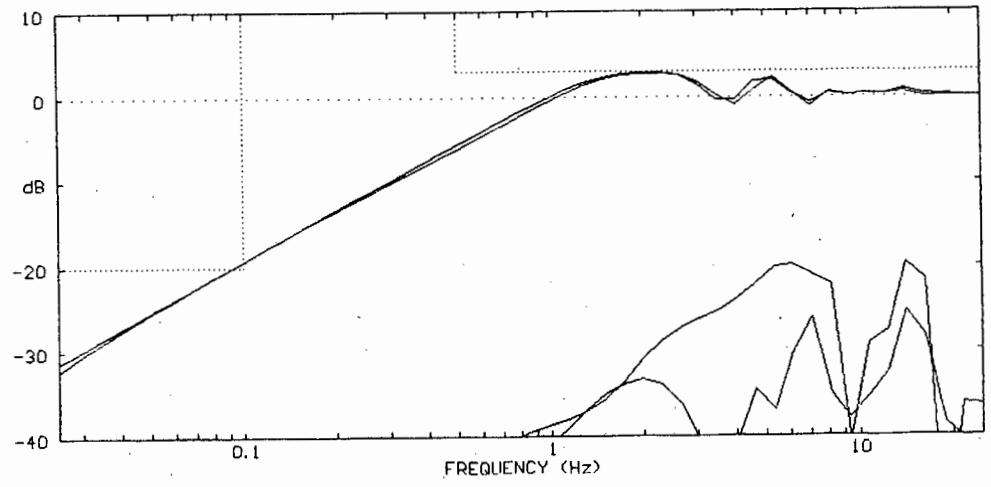


Figure 6.3. H_{DY} frequency response.

The step response of this design (figure 6.1) compares very favourably with that given by Limebeer and Maciejowski [L2], and the design displays good robustness properties (figures 6.2 and 6.3).

6.3. OTHER EXAMPLES.

The design package has been applied to numerous other plants. Chapter 7 contains details of the designs for the 2-input 2-output plant used by students during their course

on multivariable control systems. A 4-input 4-output model of a flotation plant, which included significant dead time, was studied in [T3] (appendix E).

6.4. SUMMARY.

Two design examples have been presented in this section. The first example gives the dialogue with the expert system, demonstrating some of the facilities available to assist and guide the designer. The second example illustrates the use of the design method on an unstable plant. In both cases good performance is achieved with a relative small effort on the part of the designer.

CHAPTER SEVEN. USE OF THE EXPERT SYSTEM.

7.1. INTRODUCTION.

Educational instruction is another useful application for intelligent design systems [B5]. The expert system described here has been used by two groups of students at the University of Cape Town. Although it was not primarily intended for educational purposes, the expert system did prove useful for introducing new concepts to the students. In both instances the students were not familiar with either the design system or design method, and they received minimal explicit tuition on either.

The first group of students, an undergraduate class, used an early version of the software for their control system design project (single variable). The results of this trial were presented at the 1989 RUGSA symposium [T5] (appendix G). The second group used the full MV-CXS design system during their postgraduate course on multivariable control system design.

7.2. THE POSTGRADUATE CONTROL DESIGN PROJECT.

The students were required to design a controller for the plant

$$G(s) = \begin{bmatrix} \dfrac{0.1054}{s + 0.1054} & \dfrac{0.3162}{s + 0.1054} \\ \dfrac{0.2108}{s + 0.1054} & \dfrac{0.8924}{s + 0.2231} \end{bmatrix}$$

which satisfied specifications on the step response and control signal. Appendix C details the instructions given to the students. This plant, a scaled version of that studied by Zhao and Kimura [Z2], is non-minimum phase, with a right half plane zero near s = 0.1809.

The students repeated the design using four different design approaches - initially they used single variable design methods, and then the INA/DNA method, the characteristic loci method and finally the MV-CXS package. Approximately one week was allocated for each method.

The MV-CXS design was performed on the zero-order hold equivalent of G(s), sampled at 1 Hz; the transfer function G(z) is given below. The students received no tuition on using the package other than the instructions given in appendix C.

$$G(z) = \begin{bmatrix} \dfrac{0.1}{z - 0.9} & \dfrac{0.3}{z - 0.9} \\ \dfrac{0.2}{z - 0.9} & \dfrac{0.8}{z - 0.8} \end{bmatrix}$$

The students used a one parameter controller structure for the classical designs. For the MV-CXS designs, a two parameter structure was assumed; to make the comparison with the other methods fair, only the K2 section of the design (i.e. inputs N and D) was considered. The students were, however, expected to design for command inputs at input R

also.

The designs were evaluated in terms of their ability to reduce interaction and reject disturbances, in addition to satisfying the specifications on the tracking performance and control signal. In the tables below, the column labeled 'Satisfied' indicates whether or not the design satisfied the specifications; the 'Interaction' column gives the peak off-diagonal response at the plant output to a unit step on the command input. 'Bandwidth' is the frequency (Hz) below which all disturbances at any single plant output are rejected by at least 3 dB, and 'DYmax' the maximum amplification of any disturbance at the plant output. This last parameter gives an indication of the robustness of the design; the lower the values, the more robust the design should be.

7.2.1 Single variable designs.

The students used single variable methods for the first group of designs. For example, some students used a sequential loop closing method, where a controller $k_{11}(s)$ was designed for $g_{11}(s)$, and then a second controller $k_{22}(s)$ was designed for

$$g_{22}(s) - g_{12}(s)k_{11}(s)(1 + g_{11}(s)k_{11}(s))^{-1}g_{21}(s)$$

In all cases the single variable designs were done using a classical Nyquist diagram method; in general PI (proportional plus integral) controllers were produced. The two controllers were then combined to give

$$K(s) = \begin{bmatrix} k_{11}(s) & 0 \\ 0 & k_{22}(s) \end{bmatrix}$$

Students (1) and (2) achieved good results by re-assigning the plant inputs (i.e. controlled output 1 using input 2, and output 2 using input 1) based on an analysis of the

relative gain array [M3]. In subsequent designs (except those using MV-CXS where this assignment is immaterial) almost all students adopted this approach.

The performance of the single variable designs is shown in table 7.1 below. An error in the program computing the performance of the controllers for the single variable designs may have led some students to believe incorrectly that their designs satisfied the requirements; the initial control signal value u(0) was not included when computing the maximum control signal. However the students could have observed this from the graphical plot of the control signal response, and should have been concerned about the corresponding high gains in the disturbance rejection response. These cases have been labeled as (Yes) in the 'Satisfied' column, and will be considered to have satisfied the design requirements.

Student	Satis-fied	Inter-action	Band-width	DYmax
1	Yes	1.26	0.005	5.08
2	Yes	1.02	0.010	3.20
3	No	0.61	0.005	4.04
4	No	0.60	0.010	4.73
5	(Yes)	2.51	0.003	11.90
6	No	3.11	0.004	22.71
7	No	2.23	0.002	9.83
8	(Yes)	2.32	0.003	10.50
9	(Yes)	2.40	0.003	11.08
10	Unstable			
11	Unstable			
12	(Yes)	2.41	0.003	10.58
13	Close	1.27	0.006	4.67

Table 7.1. Single variable designs.

The design by student (2) was judged to be the best in this group. The step response of this design is shown in figure 7.1 below.

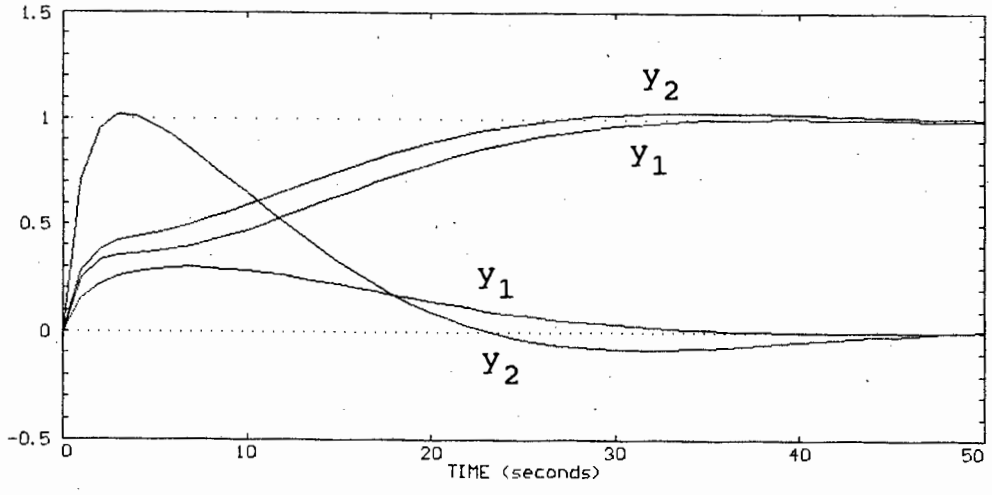


Figure 7.1. Single variable design of student (2).

7.2.2 INA/DNA designs.

Students were free to use either the Inverse Nyquist Array (INA) or the Direct Nyquist Array (DNA) design method in this section. In some cases students submitted designs performed using both methods; here the better of the two designs were used in the analysis below. The results of this group of designs are listed in table 7.2.

Student	Satis-fied	Inter-action	Band-width	DYmax
1	Close	0.99	0.011	3.28
2	No	1.69	0.004	9.14
3	Close	1.25	0.004	5.62
4	No	0.01	0.008	4.87
5	No	0.55	0	2.97
6	No	0.74	0.013	15.61
7	Yes	1.91	0.005	9.33
8	No	0.50	0.004	7.27
9	Yes	1.53	0.004	7.51
10	Unstable			
11	Close	1.62	0.005	8.66
12	No submission			
13	No	1.29	0.010	9.66

Table 7.2. INA/DNA designs.

The design by student (1) was judged to be the best in this group. The step response of this design is shown in

figure 7.2 below.

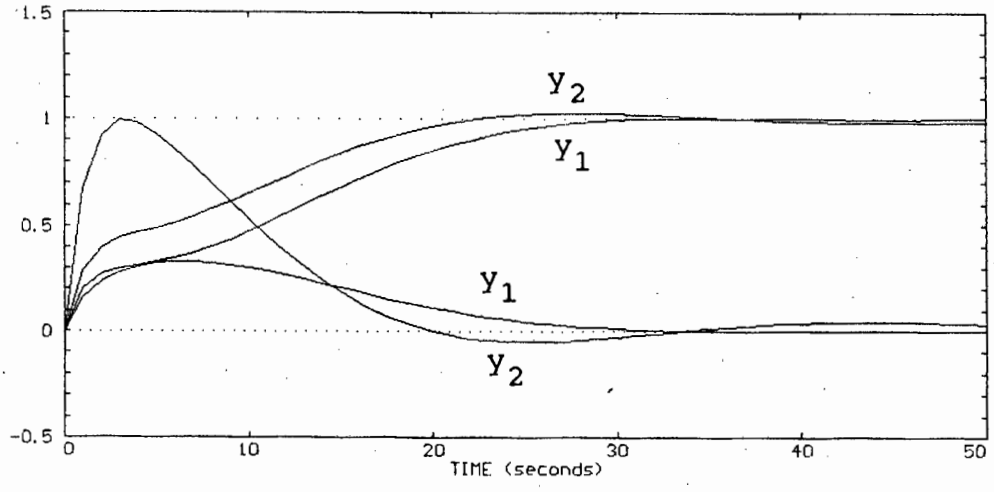


Figure 7.2. INA/DNA design of student (1).

7.2.3 Characteristic Loci designs.

From table 7.3 it is clear that the Characteristic Loci method produced the least successful results; not one student was able to even nearly satisfy the design requirements using this method. Most students found that while the method gave good indications of stability and interaction, it gave little indication of what should be done to improve the design.

Student	Satis-fied	Inter-action	Band-width	DYmax
1	No	0.08	0.004	3.38
2	No	0.02	0.004	3.27
3	No	0.70	0.004	8.13
4	Unstable			
5	Unstable			
6	Unstable			
7	No submission			
8	No	0.65	0.002	3.72
9	No	0.80	0.005	7.09
10	Unstable			
11	No	2.98	0.002	14.59
12	No submission			
13	No submission			

Table 7.3. Characteristic Loci designs.

The design by student (1) was judged to be the best in this group. The step response of this design is shown in figure 7.3 below.

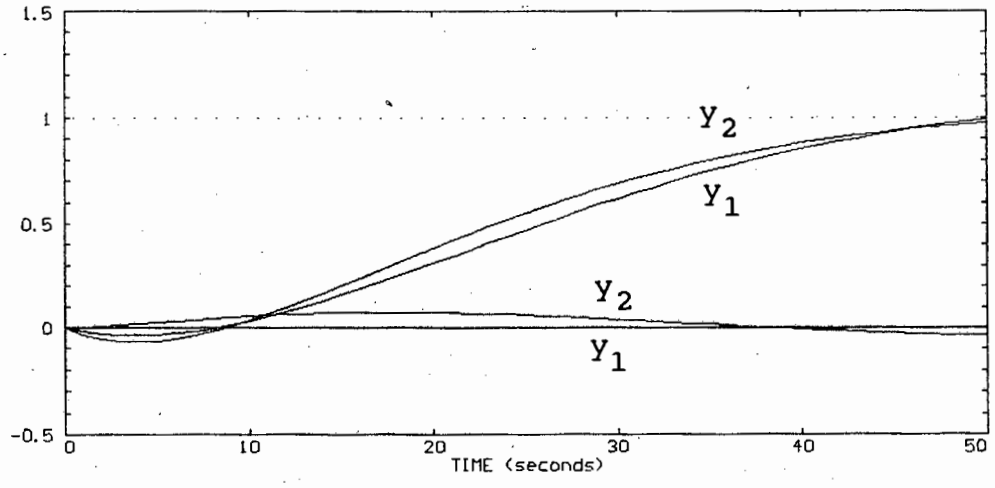


Figure 7.3. Characteristic Loci design of student (1).

7.2.4 MV-CXS designs.

The students used the MV-CXS design system for the final group of designs. Most students found this easy to use; the NEXT STEP facility seemed particularly helpful. The performance of the controllers is given in table 7.4. The version of MV-CXS used by the students used the first order approximation of Q; had the second order version been available it is most likely that the students would have achieved better results.

All students but one were able to meet the design requirements using MV-CXS. Student (6) did not constrain the off-diagonal elements of the RU response, nor any of those of the NU response. Furthermore, this student required the off-diagonal NY responses to meet the conditions intended for the diagonal elements. However, the student did not achieve even nearly acceptable results using any of the other design methods.

Student	Satis-fied	Inter-action	Band-width	DYmax
1	Yes	0.29	0.006	3.89
2	Yes	0.05	0.005	5.14
3	Yes	0.72	0.010	3.32
4	Yes	0.85	0.008	2.17
5	Yes	1.02	0.008	3.27
6	No	1.05	0.000	7.34
7	No submission			
8	Yes	0.20	0.006	4.00
9	Yes	0.20	0.005	3.82
10	Yes	0.50	0.008	2.80
11	No submission			
12	No submission			
13	No submission			

Table 7.4. MV-CXS designs.

The design by student (9) was judged to be the best in this group. The step response of this design is shown in figure 7.4 below; the performance would have been better had the student not placed tight constraints on the undershoot. While the design of student (2) has much lower interaction, it did not achieve asymptotic disturbance rejection.

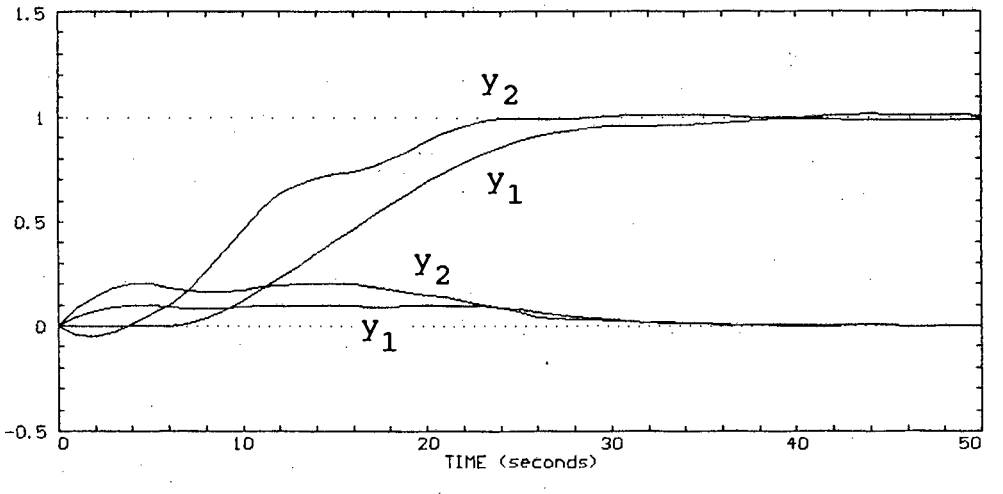


Figure 7.4. MV-CXS design of student (9).

7.2.5 Comparison of design methods.

In general it is difficult to compare designs as there are many possible performance measures. However the students were asked to minimize interaction and maximize the disturbance rejection properties of the closed loop system. The designs are therefore compared on these criteria; in addition the value of the DYmax parameter, which gives an indication of the robustness of the design, is also compared. These criteria are generally complementary; to achieve a lower interaction value it is generally necessary to accept a higher value of DYmax, for example.

The design methods are compared in the tables below; only those designs which satisfied the requirements, or were at least close, are considered. Table 7.5 below gives the method which produced the best results in terms of the interaction, bandwidth and DYmax criteria; table 7.6 gives the average value of each of these criteria. The ratio of designs which satisfied the requirements (at least approximately) to the total number submitted is also given as a percentage in table 7.6. In these tables, S.V denotes the single variable designs, INA those done using the INA/DNA method, and C.L the characteristic loci designs.

Student	Inter-action	Band-width	DYmax
1	MV-CXS	INA	INA
2	MV-CXS	S.V	S.V
3	MV-CXS	MV-CXS	MV-CXS
4	MV-CXS	MV-CXS	MV-CXS
5	MV-CXS	MV-CXS	MV-CXS
6	-	-	-
7	Incomplete		
8	MV-CXS	MV-CXS	MV-CXS
9	MV-CXS	MV-CXS	MV-CXS
10	MV-CXS	MV-CXS	MV-CXS
11	Incomplete		
12	Incomplete		
13	Incomplete		

Table 7.5. Design method giving best performance.

Method	Number of acceptable designs.	Average inter-action	Average band-width	Average DYmax
S.V	7 (54%)	1.89	0.0047	8.14
INA	5 (42%)	1.46	0.0058	6.88
C.L	0 (0%)	-	-	-
MV-CXS	8 (89%)	0.48	0.0070	3.55

Table 7.6. Average performance of acceptable designs.

Tables 7.5 and 7.6 above clearly reveal MV-CXS as the most successful of the design methods. Even though the designs involved trading each performance criterion off against the others, the MV-CXS method often produced the best results for all three criteria. However there are some factors which make the comparison with the classical design methods appear mildly unfair. These are considered below.

Firstly the design requirements were posed in the time domain, and the classical design methods operate primarily in the frequency domain. Probably more significant, however, is that MV-CXS allows the designer to satisfy closed loop performance constraints explicitly, while all the other methods can satisfy closed loop requirements only implicitly. Thus had the requirements been set in the frequency domain, it is quite probable that MV-CXS would remain the method of choice. On the other hand, two of the three performance criteria used to compare the designs are frequency domain parameters.

The second possible objection is that the controllers synthesized by MV-CXS are vastly more complex than those of the classical methods. This may be a real concern for many applications; the options for implementing the controllers are discussed in chapter 8. One of those options is to estimate a low order controller; the step response using the controller estimated from the design of student (9) is shown below. Here the elements of the controller transfer function are ratios of second order polynomials. This response is

still better than the best of those designed using the classical methods.

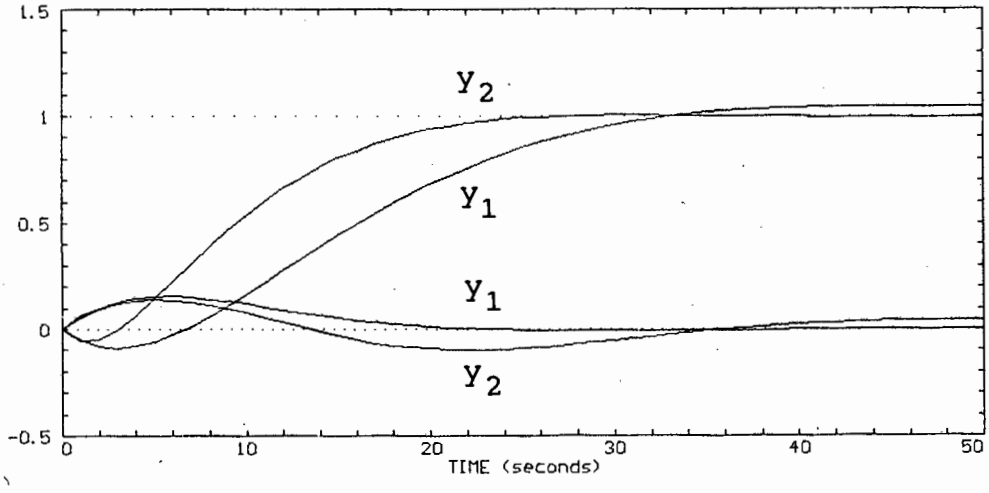


Figure 7.5. MV-CXS design of student (9) with estimated second order controller.

Finally it could be argued that the students had gained a deep understanding of the plant by the time the MV-CXS designs were performed. However any experience from previous designs is certainly not reflected in the results of the characteristic loci designs. Furthermore, the MV-CXS design method is substantially different from the others, making this possible advantage minor. It is also possible that this experience had a negative effect on the MV-CXS results, in that the students may have stopped improving their designs once they had obtained a design better than their previous attempts.

On the other hand it should be remembered that the students were given hardly any instruction on the use of MV-CXS, or on the factorization method for control system design. In contrast the classical methods used above were taught in some detail in their classes. Some students, however, had used an early version of the expert system for single variable designs as undergraduates the previous year.

In addition, it is most probable that the MV-CXS design system would have produced even better results relative to the other methods had the dimension of the plant been higher. The relative increase in the level of skill required of the designer as the dimension of the problem increases is far greater for the classical methods than it is for MV-CXS. Using MV-CXS, the designer is required to select the most appropriate trade-offs for the design; the mechanics of the design process remain largely hidden.

In summary, the results of this experiment were extremely encouraging. The goal of the design system, that of assisting users to produce a good control system designs, has been effectively fulfilled. A significant degree of this success is based on the experience gained during the undergraduate design project [T5].

7.2.6 Alternative designs.

In addition to using the design methods above, two students submitted what they called analytic designs. These were based on an algebraic computation of a compensator to give exact decoupling, followed by two single variable designs for the decoupled plant; the first step is not difficult given the simplicity of the plant transfer function. The non-zero interaction values in the tables below arise from imperfections in the simulation program. Of the two designs, that of student (9) was particularly good. The step response of this design is shown in figure 7.6 below. Similar performance is also possible using MV-CXS; figure 7.7 shows a design generated using the second order approximation for Q, with the QSTEP parameter set at 4, and NVARS set at 12.

Student	Satis-fied	Inter-action	Band-width	DYmax
8	Close	0.01	0.005	4.33
9	Yes	0.01	0.006	4.42
MV-CXS	Yes	0.01	0.006	4.42

Table 7.7 : Analytical designs.

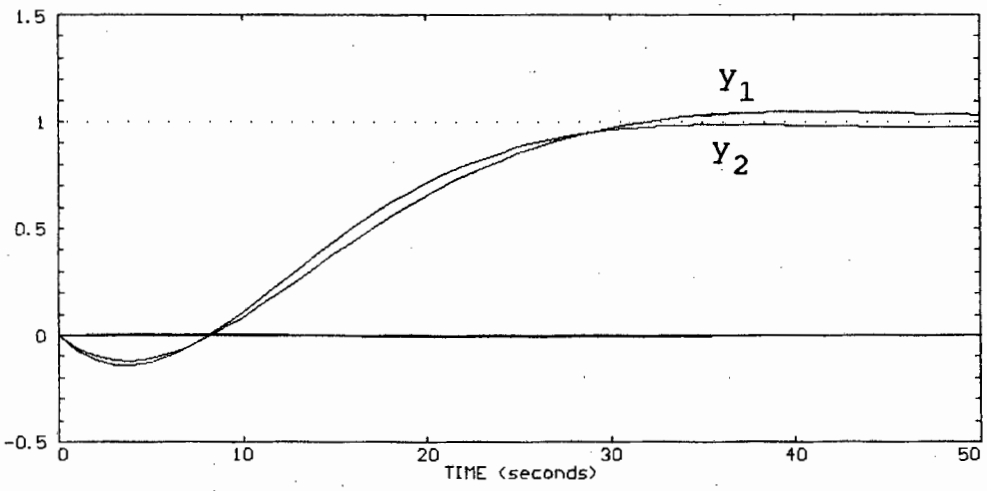


Figure 7.6. Analytical design of student (9).

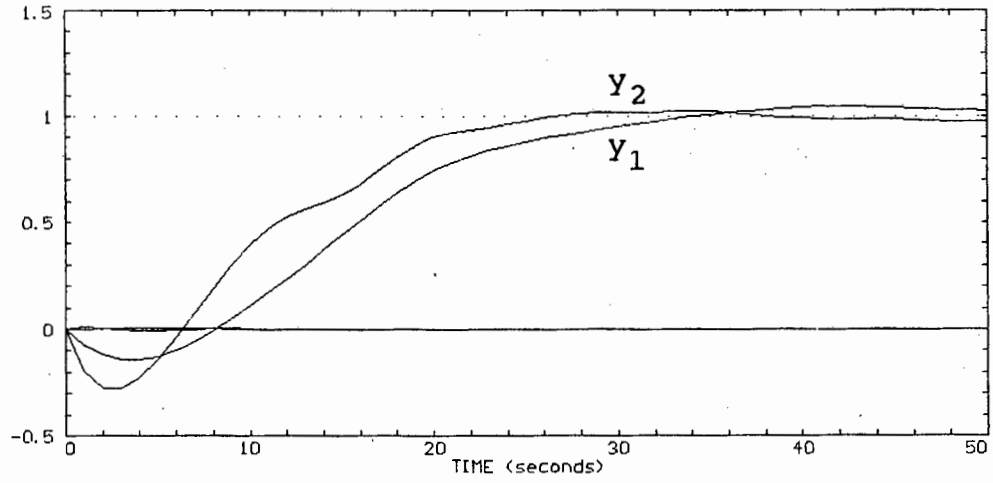


Figure 7.7. MV-CXS design.

7.3. SUMMARY.

The effectiveness of the design system has been tested by two groups of students at the University of Cape Town. An early version, for single-variable design problems, was used by undergraduate students during their control systems design project [T5]. This chapter dealt with the use of the MV-CXS design system by postgraduate students during their course on multivariable systems. The results here were particularly encouraging, with the expert system tool enabling them to produce markedly superior results compared to various classical design methods.

CHAPTER EIGHT. IMPLEMENTING THE CONTROLLER.

8.1. INTRODUCTION.

An unfavourable consequence of the design method is the very high order of the resulting controllers. Boyd *et al.* [B6] give three possible uses for controllers produced by this method; they may be implemented in full, reduced order controllers may be derived from them, or they may be used as a standard against which controllers designed by other methods may be measured. This chapter examines the first two alternatives.

8.2. FULL CONTROLLER IMPLEMENTATION.

Boyd *et al.* [B6] propose an architecture for implementing the controller directly. However, when the plant transfer function is strictly proper, a simpler implementation is possible.

Let the nominal stabilizing controller **K0** have the stable left-coprime factorization

$$\mathbf{K0} = \mathbf{Y}^{-1}.\mathbf{X}. \qquad (8.2.1)$$

Then formula (2.3.4) for the controller can be written as

$$\mathbf{Kc} = (\mathbf{Y} - \mathbf{Q2}.\tilde{\mathbf{N}})^{-1},$$

$$\mathbf{K1} = \mathbf{Kc}.\mathbf{Q1},$$

and

$$\mathbf{K2} = \mathbf{Kc}.(\mathbf{X} + \mathbf{Q2}.\tilde{\mathbf{D}}). \qquad (8.2.2)$$

For clarity it will be assumed that the inputs D, N, and V are all zero. The control signal $\mathbf{u}(z)$ can then be computed as

$$\mathbf{u}(z) = \mathbf{Kc}(z).\mathbf{e}(z)$$

$$= (\mathbf{Y} - \mathbf{Q2}(z).\tilde{\mathbf{N}}(z))^{-1}\mathbf{e}(z), \qquad (8.2.3)$$

where $$\mathbf{e}(z) = \mathbf{Q1}(z).\mathbf{r}(z) - (\mathbf{X}(z) + \mathbf{Q2}(z).\tilde{\mathbf{D}}(z))\mathbf{y}(z). \qquad (8.2.4)$$

Note that the transfer function matrices **Q1**, **Q2**, **X**, and $\tilde{\mathbf{D}}$ are stable, and thus the error signal $\mathbf{e}(kT)$ at time kT is easily computed from the sequences $\{\mathbf{r}(iT), i \leq k\}$ and $\{\mathbf{y}(iT), i \leq k\}$.

Partition $\mathbf{Y}(z)$ into the real matrix $\mathbf{Y}_\infty$ and the strictly proper $\mathbf{Y}'(z)$, where

$$\mathbf{Y}_\infty = \lim_{z\to\infty} \mathbf{Y}(z) \qquad (8.2.5)$$

and $$\mathbf{Y}'(z) = \mathbf{Y}(z) - \mathbf{Y}_\infty. \qquad (8.2.6)$$

Define $$\mathbf{Yc}(z) = \mathbf{Q2}(z).\tilde{\mathbf{N}}(z) - \mathbf{Y}'(z) \qquad (8.2.7)$$

and $$\mathbf{Yi} = \mathbf{Y}_\infty^{-1}. \qquad (8.2.8)$$

Assuming that **Yi** exists (that is, $\mathbf{Y}(z)^{-1}$ exists and is proper), equation 8.2.3 can be rewritten as

$$(\mathbf{Y}_\infty - \mathbf{Yc}(z))\mathbf{u}(z) = \mathbf{e}(z),$$

or $$\mathbf{u}(z) = \mathbf{Yi}[\mathbf{e}(z) + \mathbf{Yc}(z)\mathbf{u}(z)]. \qquad (8.2.9)$$

Note that if the controller is stable, or the diagonal factorization method discussed in chapter 2 is used to compute the fractions **X** and **Y**, then

$$\mathbf{Yi} = \mathbf{Y}_\infty^{-1} = \mathbf{I}. \qquad (8.2.10)$$

For strictly proper **G**, $\tilde{\mathbf{N}}$ is also strictly proper, and thus **Yc**(z).z is proper and can be realized. This gives a direct formula for computing the control signal **u**(kT) at time kT from the error signal **e**(kT), and the sequence {**u**(iT), i < k} representing the control signal history.

$$\mathbf{u}(z) = \mathbf{Yi}[\mathbf{e}(z) - (\mathbf{Yc}(z).z)(z^{-1}\mathbf{u}(z))] \qquad (8.2.11)$$

Thus the controllers may be implemented directly in terms of the stable transfer function matrices **Q1**, **Q2**, $\tilde{\mathbf{N}}$, $\tilde{\mathbf{D}}$, and **Yc**; figure 8.1 illustrates this architecture. The transfer function matrices **Q1**, **Yc**, and (**X** + **Q2**.$\tilde{\mathbf{D}}$), may be approximated by high order FIR filters as suggested by Boyd *et al.* [B6]; note that $\tilde{\mathbf{N}}$ in particular may require a very high order FIR filter for accurate approximation. When the FIR approximation is used, the asymptotic properties of the controller may be preserved by ensuring that the dc gain of the approximation is exactly that of the transfer function form.

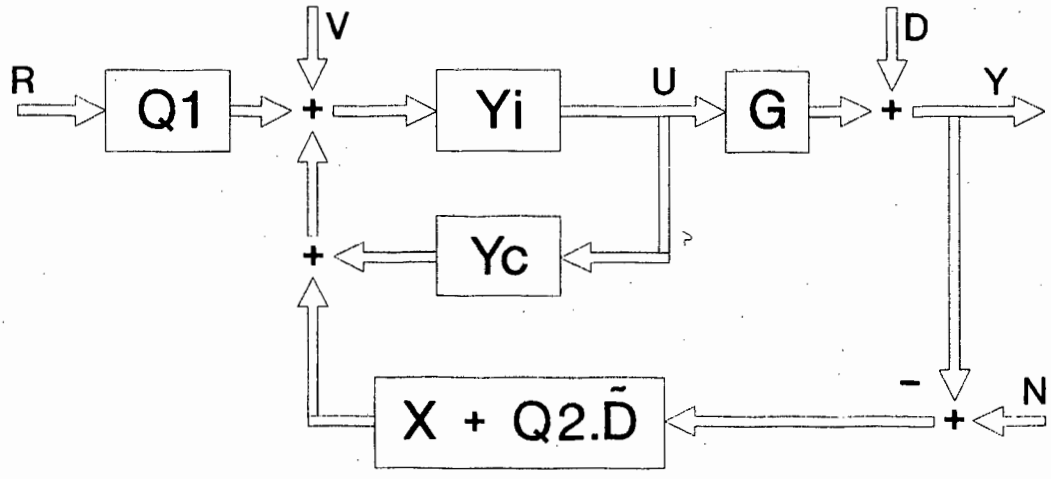


Figure 8.1. Controller structure.

It is of course possible to implement the full controller without using the FIR filter approximation, but implementing $\tilde{\mathbf{N}}$, $\tilde{\mathbf{D}}$, **X** and **Y'** as dynamic systems. **Q1** and **Q2**, by their definition, should still be implemented as FIR filters. An

efficient structure for this form is shown in figure 8.2. This form simplifies considerably when the plant is stable, as $\tilde{N} = G$, $\tilde{D} = I$, and $X = Y' = 0$.

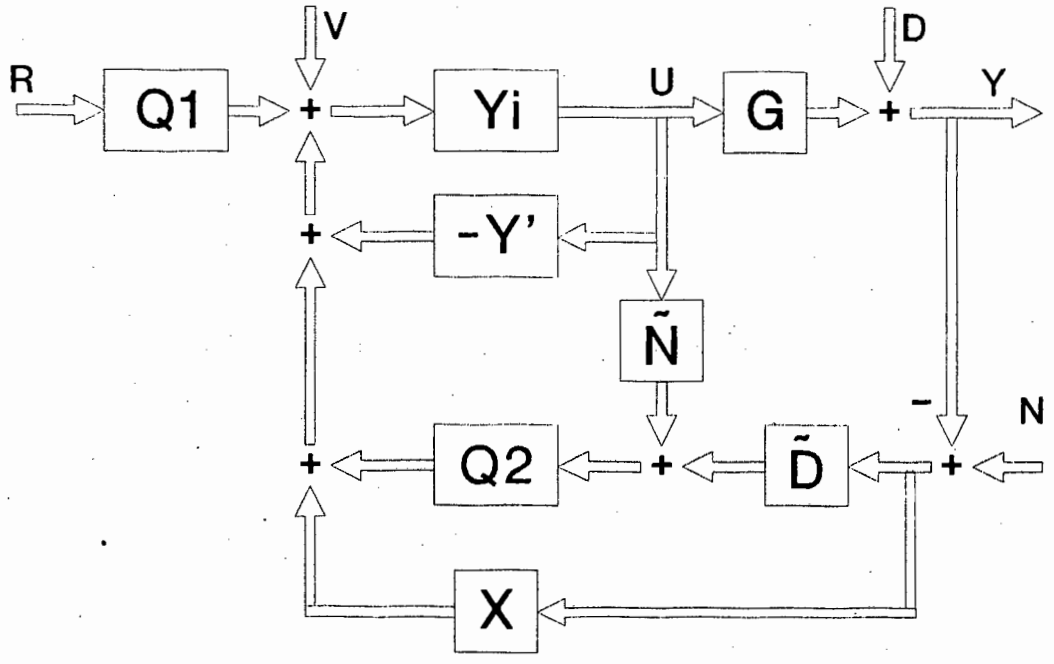


Figure 8.2. Controller structure for full dynamic system implementation.

A consequence of the complexity of the controller is that manual online tuning is virtually impossible; the overall gain of the controller is easily be adjusted, however, by scaling the control signal **u** by a (diagonal) gain matrix. An alternative approach to tuning is 'online' re-design of the control system with appropriate modifications to the design specification. Since the design algorithm is fast, this is quite feasible, particularly where the control law can be downloaded directly from the design system to the controller hardware. An expert system could again be profitably employed to assist the user in deciding how to change the design specifications to achieve the desired online performance.

8.3. REDUCED ORDER CONTROLLER ESTIMATION.

The second approach is to estimate a reduced order controller which approximates the high order synthesized controller. The CACSD package contains a simple estimation algorithm; the algorithm used is by no means the best available, and is included simply to illustrate the utility of this approach. For an good overview of this complex subject, Anderson and Liu [A1] give a discussion of controller reduction principles.

The estimation algorithm used is a least squares fit of a low order transfer function to the controller frequency response. The individual elements of the controller matrix are treated independently, giving n^2 least squares problems for each of **K1** and **K2**. The analysis below relates to the **K2**[a,b] controller element; the procedure is identical for the other elements of **K2**, and very similar for those of **K1**.

Let the controller element be modelled by the equation

$$k = \frac{n(z)}{d(z)} \qquad (8.3.1)$$

where $$n(z) = n_m z^m + n_{m-1} z^{m-1} + \ldots + n_0, \qquad (8.3.2)$$

$$d(z) = z^m + d_{m-1} z^{m-1} + \ldots + d_0, \qquad (8.3.3)$$

and m is the desired order for the controller element. The original full controller is evaluated at the set of points $\{z_i, i=1...N\}$ on the unit circle in the complex plane, with $N > 2m$; let K_i denote the controller response **K2**[a,b](z_i). Then the controller estimation can be formulated as a weighted non-linear least squares problem to minimize

$$\sum_{i=1}^{N} w(z_i) \left[K_i - \frac{n(z_i)}{d(z_i)} \right]^2, \qquad (8.3.4)$$

where w(z) is a frequency dependent function, chosen so that the estimation will be more accurate at critical frequencies. Multiplying each term in the sum by the factor $d(z_i)$ gives

$$\sum_{i=1}^{N} w(z_i)\left[K_i d(z_i) - n(z_i) \right]^2, \qquad (8.3.5)$$

which is linear in the unknown coefficients. Here the effective weight, in terms of equation (8.3.4), is

$$w'(z_i) = \frac{w(z_i)}{d(z_i)} . \qquad (8.3.6)$$

In this form the coefficients d_i, i=0...m-1, and n_i, i=0...m of the reduced order controller are easily estimated using a standard linear least squares algorithm. Note that the solutions to (8.3.4) and (8.3.5) are different in general.

The estimation algorithm implemented in the CACSD package is a combination of the linear least squares problems given by (8.3.4) and (8.3.5), with w(z) = 1. First the problem given by (8.3.5) is solved; the denominator of the controller element is taken from this. A second least squares fit, in the form of equation (8.3.4) and using the denominator d(z) identified above, is then used to find the numerator polynomial. The corresponding element of the K1 part of the controller uses the denominator identified for the K2 part, and solves the equivalent least squares problem given by (8.3.4) to find the numerator. This approach ensures that the two controller sections have the same poles, which simplifies implementation where the controller has unstable poles (for example at z = 1).

8.4. EXAMPLES OF CONTROLLER IMPLEMENTATION.

The first example from chapter 6, the gold mine milling unit with three inputs and three outputs, will be used to illustrate these two approaches to controller implementation. Since the plant is stable, the trivial factorization given by equation 2.4.1 is used. The implementations are compared using the NY step response, observed over the first 100 seconds. The response of the full controller is shown in figure 8.3 below.

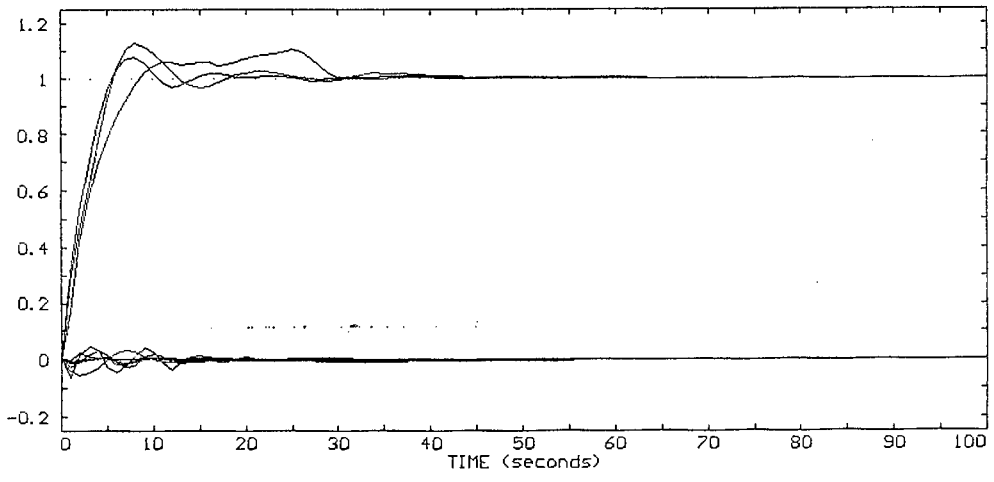


Figure 8.3. Response using the full controller.

The responses for the controller implemented (approximately) using FIR filters are shown in figures 8.4 and 8.5, using 50 and 75 tap filters respectively. Here the FIR filters are the truncated impulse responses of the transfer function matrices

$$\mathbf{F1 = Q1,}$$

$$\mathbf{F2 = X + Q2.D = Q2,}$$

and $$\mathbf{F3 = Yc = Q2.G.}$$

The FIR filters for F1 and F2 in this example are precisely

those of the full order controller; **F3** will differ, as the impulse response of **G** is infinite. The effects of this approximation are clearly seen when using 50 tap filters; in particular the integral action has been lost, giving a steady state tracking error.

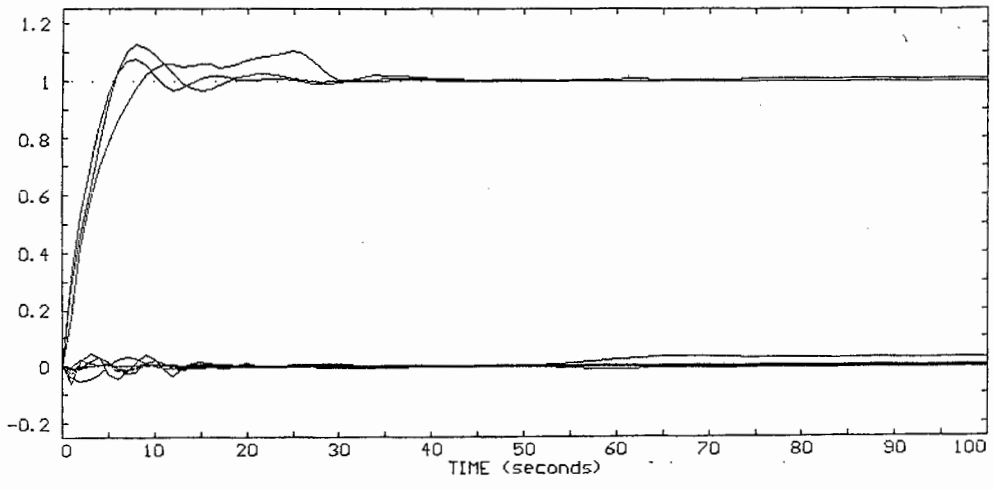


Figure 8.4. Response for controller using 50 tap FIR filters.

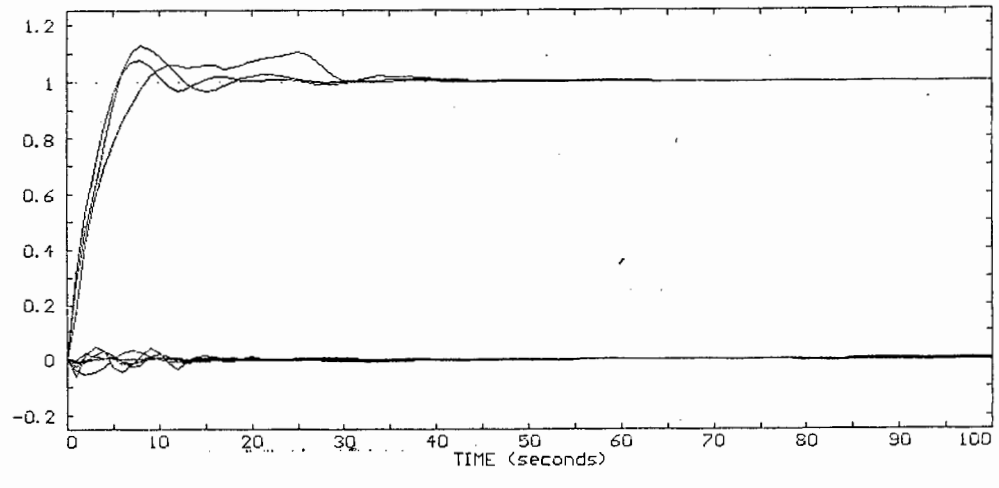


Figure 8.5. Response for controller using 75 tap FIR filters.

Attempts to remedy the situation by adjusting the DC gain of the FIR filters to match those of the full order controller are shown in figures 8.6 and 8.7. Again, these plots are for the 50 and 75 tap filters respectively.

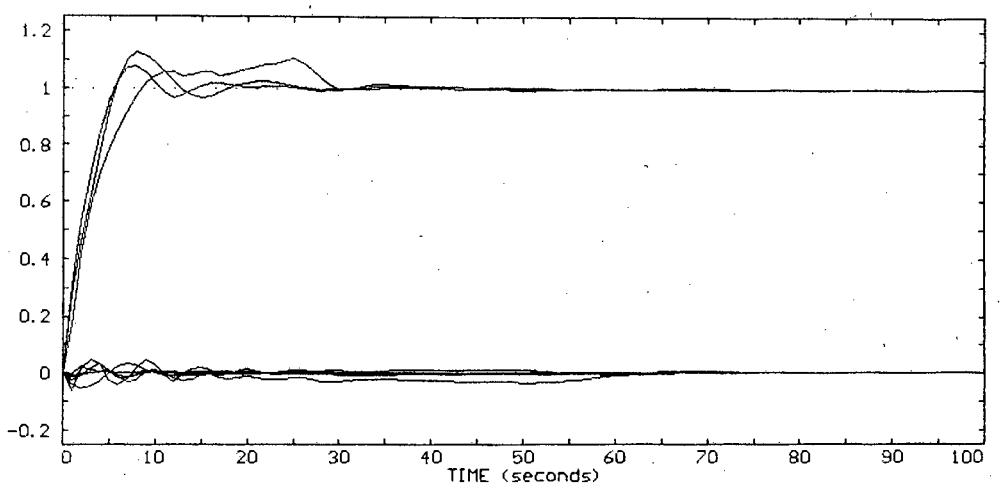


Figure 8.6. Response for controller using 50 tap FIR filters, with DC correction.

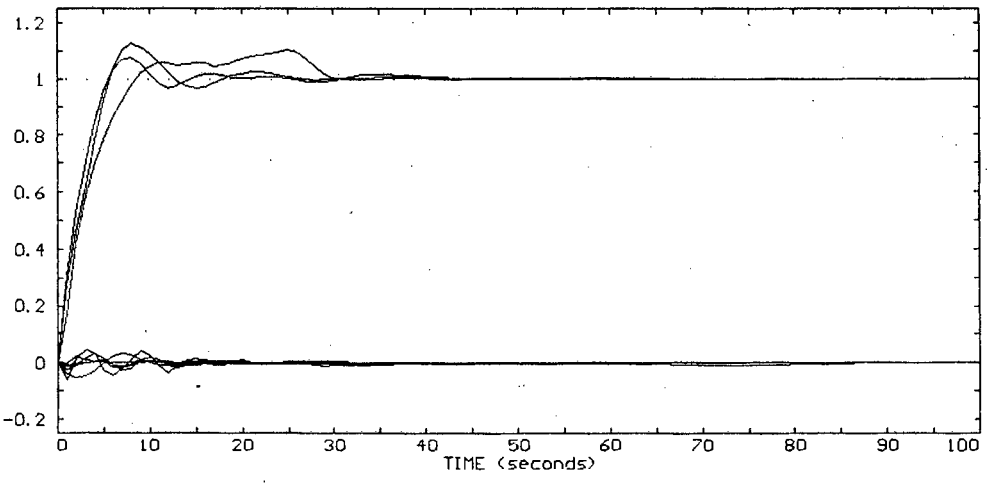


Figure 8.7. Response for controller using 50 tap FIR filters, with DC correction.

Next the estimation method described above was applied to the problem. The estimated controller with first order elements destabilised the system. Figures 8.8, 8.9 and 8.10 show the step response for controllers with elements of order 2, 4 and 6 respectively. It is interesting that, for this example, the step response using fourth order approximation is worse than that using the second order estimate.

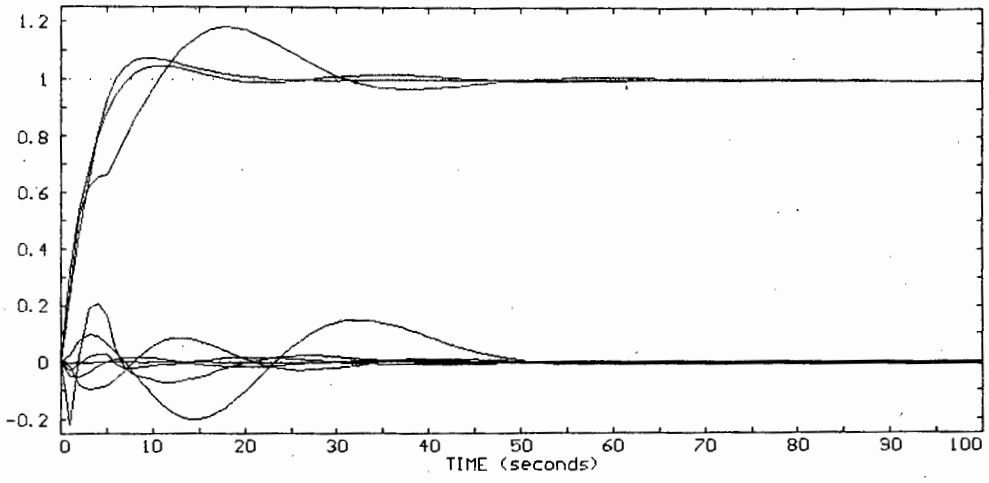


Figure 8.8. Response using estimated controller, m = 2.

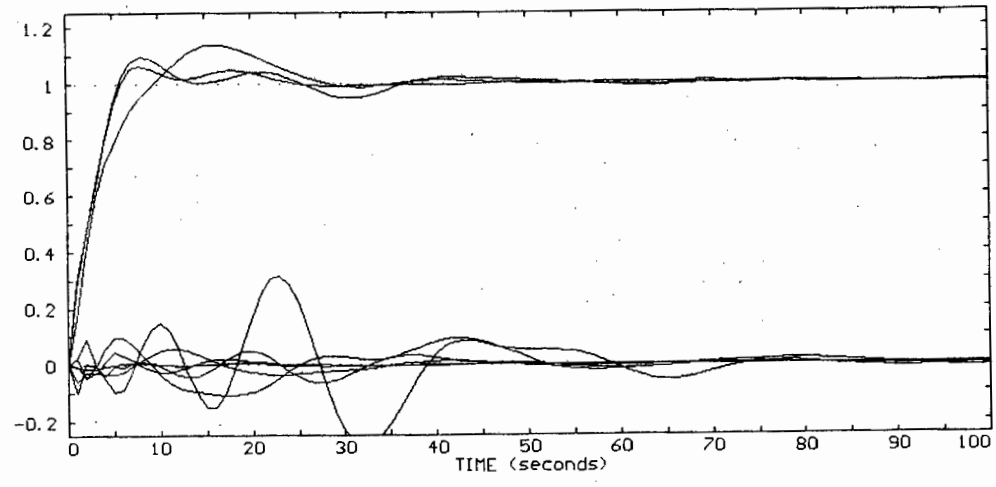


Figure 8.9. Response using estimated controller, m = 4.

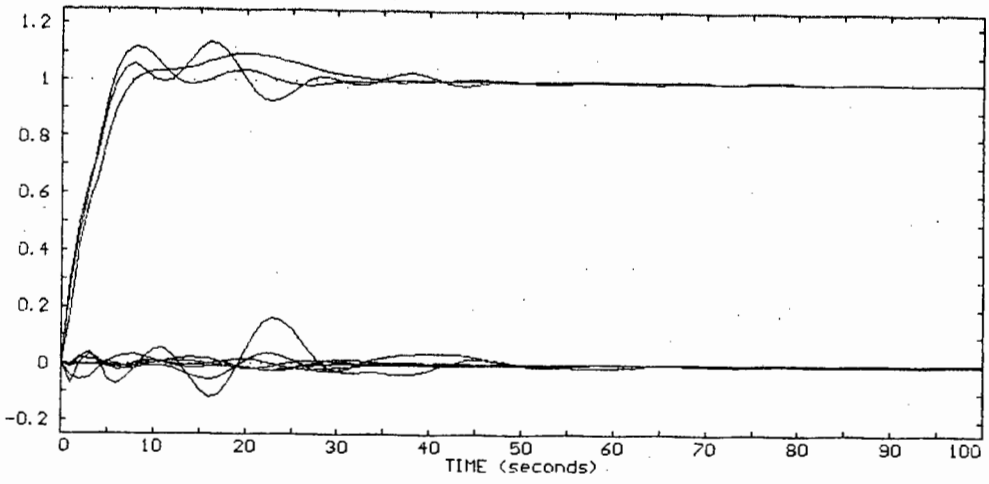


Figure 8.10. Response using estimated controller, m = 6.

8.5. SUMMARY.

Two alternatives for using the complex controllers synthesized by this design method have been discussed in this chapter. The first option is to implement the controllers in full, or at least approximately using high order finite impulse response filters. The second alternative is the estimation of reduced order controllers. A simple identification algorithm which estimates a low order controller from the frequency response of the full controller has been used to demonstrate the feasibility of this approach.

CHAPTER NINE. CONCLUSIONS.

A powerful CACSD package, based on the design method of Boyd *et al.* [B6], has been constructed. This method translates the control system design problem into, and solves it as, a linearly constrained quadratic programming problem. The efficiency of the design method, in terms of both memory requirements and execution speed, has been improved substantially.

A diagonal factorization technique was developed; when applied to the left factorization of the plant transfer function matrix, it allows the multivariable design problem to be reduced to a number of smaller sub-problems, which may be solved independently. Although this diagonal factorization is not always coprime, it is suitable for a wide range of plant transfer function matrices. A theorem to check that the factorization is coprime was developed, and is easy to apply. Formulae for (non-diagonal) coprime factorizations, where the nominal stabilizing controller is stable, have also been presented. A novel parameterization for the design transfer function **Q** has been introduced; by appropriate choice of the QSTEP parameter the designer benefits from the efficiency of a low order approximation and while enjoying almost the same precision as for a high order approximation. Finally, a very efficient representation for the linear constraints generated by the design method has been developed.

The extent of these improvements in efficiency is such that the solution of substantial multivariable control system design problems is practical using only a microcomputer. Similarly these techniques would enable large scale problems

to be tackled on a more powerful workstation.

The complex controllers synthesized by this design method may be implemented in full, or using high order finite impulse response filters. Controller reduction provides a second alternative. A simple identification algorithm to estimate a low order controller from the frequency response of the full controller has been included in the CACSD package, and used to demonstrate the feasibility of this approach.

An expert system interface to this CACSD package has been implemented to produce an intelligent, interactive design tool. The expert system guides and assists both novice and experienced designers in using the CACSD package. Based upon a database of common design features, the NEXT STEP and SUGGEST commands are able to guide and assist the user in formulating and refining the design specification. Similarly the COMPLETE command helps the user to check that the design is complete. Through analysis of the active set and Lagrange multipliers from the quadratic programming problem, the expert system is able to assist the user in identifying and dealing with conflicting performance constraints. The expert system has also been used to effectively extend the scope of the design method, as well as to integrate information from analyses of the design.

The expert system has been implemented on a personal computer, co-resident in memory with the CACSD package. This combination has produced a comprehensive control system design tool, which is nevertheless easy to use. Using a more powerful computer would allow the knowledge base to be extended further, covering even more design situations.

The effectiveness of the design system has been tested by students at the University of Cape Town. An early version, for single-variable design problems, was used by undergraduate students during their control systems design

project; the complete MV-CXS design tool was used by postgraduate students during their course on multivariable systems. In both cases, the expert system was used to introduce the students to a new approach to control system design, with minimal additional tuition. The results for the multivariable designs were particularly encouraging, with the expert system tool enabling the students to produce markedly superior results compared to various classical design methods.

Despite the power of the design system there remains a chasm between artificial and real intelligence. Since the designer has real intelligence, the aim of the expert system has been to assist and complement, rather than replace, the designer. This produces a team solution, which capitalizes on the inherent strengths of both the designer and computer.

REFERENCES.

[A1] Anderson, B. D. O, Y. Liu, "Controller Reduction : Concepts and Approaches", *IEEE Transactions on Automatic Control*, vol. 34, no. 8, pp. 802-812, August 1989.

[A2] Antsaklis, P. J, "Neural Networks in Control Systems", *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 3-5, April 1990.

[A3] Åström, K. J, "Computer Aided Tools for Control System Design", in M. Jamshidi, C. J. Herget, eds., *Computer-Aided Control Systems Engineering*, Amsterdam: North-Holland, pp. 3-40, 1985.

[A4] Åström, K. J, J. J. Anton, K.-E. Årzén, "Expert Control", *Automatica*, vol. 22, no. 3, pp. 277-286, 1986.

[B1] Becker, R. G, A. J. Heunis, D. Q. Mayne, "Computer-aided design of control systems via optimization", *Proceedings of the IEE*, vol. 126, no. 6, pp. 573-578, June 1979.

[B2] Bhattacharyya, S, L. Keel, J. Howse, "Stabilizability conditions using linear programming", *IEEE Transactions on Automatic Control*, vol. 33, no. 5, pp. 460-463, May 1988.

[B3] Birdwell, J. D, J. R. B. Cockett, R. Heller, R. W. Rochelle, A. J. Laub, M. Athans, L. Hatfield, "Expert Systems Techniques in a Computer-based Control System Analysis and Design Environment", *Proc. 3rd IFAC/IFIP Symposium*, Lyngby, Denmark, 1985.

[B4] Birdwell, J. D, "An Expert System can aid in the Evolution of a Design Methodology", *Proceedings of the American Control Conference*, 1987.

[B5] Birdwell, J. D, G. Chang, P. Hansen, L. Hong, J.-S. Lai, G. V. Murphy, "Teaching with the CASCADE Computer-Aided Control System Design Environment", *Proceedings of the 19th Southeastern Symposium on System Theory*, Clemson, SC, March 1987.

[B6] Boyd, S. P, V. Balakrishnan, C. H. Barrat, N. M. Kraishi, X. Li, D. G. Meyer, S. A. Norman, "A New CAD Method and Associated Architectures for Linear Controllers," *IEEE Transactions on Automatic Control*, vol. AC-33, no. 3, pp. 268-283, March 1988.

[B7] Boyd, S. P, C. H. Barrat, S. A. Norman, "Linear Controller Design : Limits of Performance via Convex Optimization", *Proceedings of the IEEE*, vol. 78, no. 3, pp. 529-574, March 1990.

[B8] Boyle, J.-M, G. K. H. Pang, A. G. J. MacFarlane, "The development and implementation of MAID: a knowledge based support system for use in control system design", *Transactions of the Institute of Measurement and Control*, vol. 11, no. 1, pp. 25-39, 1989.

[B9] Buchanan, B. G, E. H. Shortliffe, *Rule Based Expert Systems*, Addison-Wesley, 1984.

[B10] Businger, P. A, G. H. Golub, "Singular Value Decomposition of a Complex Matrix", *Communications of the ACM*, vol. 12, no. 10, October pp. 564-565, 1964.

[D1] Denham, M. J, "Design Issues for CACSD Systems", *Proceedings of the IEEE*, vol. 72, no. 12, pp. 1714-1723, December 1984.

[D2] Doyle, J. C, G. Stein, "Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis", *IEEE Transactions on Automatic Control*, vol. 26, no. 1, pp. 4-16, February 1981.

[D3] Dreyfus, H. L, S. E. Dreyfus, *Mind over Machine*, New York: The Free Press, 1986.

[D4] Dreyfus, H. L, S. E. Dreyfus, "Why Expert Systems Do Not Exhibit Expertise", *IEEE Expert*, vol. 1, no. 2, pp. 80-83, 1986.

[E1] Edmunds, J. M, "Cambridge linear analysis and design program", *Proceedings of the 1st IFAC Symposium on Computer-Aided Control System Design*, Zurich, pp. 253-258, 1979.

[F1] Fegley, K. A, "Designing Sampled-Data Control Systems by Linear Programming", *IEEE Transactions on Applications and Industry*, vol. 83, pp. 198-200, May 1964.

[F2] Francis, B. A, *A Course in H_∞ Control Theory*, Berlin: Springer-Verlag, 1987.

[G1] Genesereth, M. R, "The role of plans in intelligent teaching systems". In D. Sleeman and J. S. Brown (Eds.), *Intelligent Tutoring Systems*, London: Academic Press, 1982.

[G2] Gevarter, W. B., "The Nature and Evaluation of Commercial Expert System Building Tools", *Computer*, vol. 20, no. 5, pp. 24-41, May 1987.

[G3] Golub, G. E, C. F. van Loan, *Matrix Computations*, Oxford: North Oxford Academic, 1983.

[G4] Graham, N, *Artificial Intelligence*, Blue Ridge Summit, PA: Tab Books, 1979.

[G5] Gustafson, C. L, C. A. Desoer, "Controller design for linear multivariable feedback systems with stable plants, using optimization with inequality constraints", *International Journal of Control*, vol. 37, no. 5, pp. 881-907, 1983.

[G6] Gustafson, C. L, C. A. Desoer, "A CAD methodology for linear multivariable feedback systems based on algebraic theory", *International Journal of Control*, vol. 41, no. 3, pp. 653-675, 1985.

[H1] Haest, M, G. Bastin, M. Gevers, V. Wertz, "ESPION: an Expert System for System Identification", *Automatica*, vol. 26, no. 1, pp. 85-95, January 1990.

[H2] Hayes-Roth, F, D. A. Waterman, D. B. Lenat, *Building Expert Systems*. Reading, MA: Addison-Wesley, 1983.

[H3] Hulbert, D. G, M. Braae, "Multivariable Control of a Milling Circuit at East Driefontein Gold Mine", *National Institute for Metallurgy*, Report 2113, August 1981.

[J1] Jackson, P, P. Lefrere, "On the application of rule-based techniques to the design of advice-giving systems". In M. J. Coombs (Ed.), *Developments in expert systems*, London: Academic Press, 1984.

[J2] James, J. R, D. K. Frederick, P. P. Bonissone, J. H. Taylor, "A retrospective view of CACE-III: considerations in co-ordinating symbolic and numeric computation in a rule based expert system", *Proceedings of the Second Conference on A.I. Applications*, Miami Beach, Florida, 1986.

[J3] James, J. R, D. K. Frederick, J. H. Taylor, "Use of expert-systems programming techniques for the design of lead-lag compensators", *IEE Proceedings Part D*, vol. 134, no. 3, pp. 137-144, May 1987.

[J4] Jamshidi, M, C. J. Herget, eds., *Computer-Aided Control Systems Engineering*, Amsterdam: North-Holland, 1985.

[K1] Kailath, T, *Linear Systems*, Englewood Cliffs, N.J.: Prentice-Hall, 1980.

[L1] Larsson, J. E, P. Persson, "Knowledge Representation by Scripts in an Expert Interface", *Proceedings of the American Control Conference*, Seattle, USA, pp. 1159-1162, 1986.

[L2] Limebeer, D. J. N, J. M. Maciejowski, "Two tutorial examples of multivariable control system design", *Transactions of the Institute of Measurement and Control*, Vol. 7, no. 2, pp. 97-107, April 1985.

[L3] Little, J. N, A. Emami-Naeini, S. N. Bangert, "CTRL-C and Matrix Environments for the Computer-Aided Design of Control Systems", in M. Jamshidi, C. J. Herget, eds., *Computer-Aided Control Systems Engineering*, Amsterdam: North-Holland, 1985, pp. 111-124.

[M1] MacDuffee, C. C, *Theory of Matrices*, New York: Chelsea, 1946.

[M2] MacFarlane, A. G. J, G. Gruebel, J. Ackerman, "Future design Environments for Control Engineering", *Automatica*, vol. 25, pp. 165-176, 1989.

[M3] Maciejowski, J. M, *Multivariable Feedback Design*, Wokingham, England: Addison-Wesley, 1989.

[M4] Michie, D. (ed), *Introductory Readings in Expert Systems*, New York: Gordon and Breach Science Publishers, 1982.

[M5] Moore, K. L, S. P. Bhattacharyya, "A technique for choosing zero locations for minimal overshoot", *IEEE Transactions on Automatic Control*, vol. 35, no. 5, pp. 577-580, May 1990.

[N1] Nett, C. N, C. A. Jacobson, M. J. Balas, "A Connection Between State-Space and Doubly Coprime Fractional Representations," *IEEE Transactions on Automatic Control*, vol. AC-29, no. 9, pp. 831-832, September 1984.

[N2] Nolan, P. J, "An Intelligent Assistant for Control System Design", *Proceedings of the 1st International Conference on the Application of Artificial Intelligence in Engineering Problems*, University of Southampton, U.K., 1986.

[P1] Pang, G. K. H, A. G. J. MacFarlane, *An Expert Systems Approach to Computer-Aided Design of Multivariable Systems*, Berlin: Springer-Verlag, 1987.

[P2] Pang, G. K. H, "An Expert System for CAD of Multivariable Control Systems using a Systematic Design Approach", *Proceedings of the American Control Conference*, 1987.

[P3] Pang, G. K. H, M. Vidyasagar, A. J. Heunis, "Development of a New Generation of Interactive CACSD Environments", *IEEE Control Systems Magazine*, vol. 10, no. 5, pp. 40-44, August 1990.

[P4] Polak, E, D. Q. Mayne, D. M. Stimler, "Control System Design Via Semi-Infinite Optimization: A Review", *Proceedings of the IEEE*, vol. 72, no. 12, pp. 1777-1794, December 1984.

[P5] Press, W. H, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, *Numerical Recipes*, Cambridge: Cambridge University Press, 1986.

[R1] Rich, E, *Artificial Intelligence*, New York: McGraw-Hill, 1983.

[R2] Rosenbrock, H. H, *Computer-Aided Control System Design*, New York: Academic Press, 1974.

[R3] Rychener, M. D, *Expert Systems for Engineering Design*, New York: Academic Press, 1988.

[S1] Scales, L. E, *Introduction to Non-Linear Optimization*, London: MacMillan Publishers, 1985.

[T1] Taylor, J. H, D. K. Frederick, "An Expert System Architecture for Computer-Aided Control Engineering", *Proceedings of the IEEE*, vol. 72, no. 12, pp. 1795-1805, December 1984.

[T2] Tebbutt, C. D, "An Efficient Representation for Linear Constraints", *IEEE Transactions on Automatic Control*, vol. 35, no. 8, pp. 949-951, August 1990.

[T3] Tebbutt, C. D, "A Microprocessor Implementation of Multivariable Factorization Theory", *submitted to IEEE Transactions on Automatic Control*, February 1990.

[T4] Tebbutt, C. D, "Expert Systems Approach to Controller Design", *IEE Proceedings Part D*, to appear.

[T5] Tebbutt, C. D, "Use of an Expert System for Controller Design", *presented at the RUGSA Symposium on the Applications of Knowledge-based Systems in Engineering and Control*, Johannesburg, South Africa, October 1989. Reprinted in *Computech*, vol. 6, no. 7, pp. 30-33, July 1990.

[T6] Tebbutt, C. D, "An Expert System for Controller Design", *Transactions of the SAIEE*, vol. 81, no. 3, pp. 15-19, September 1990.

[T7] Tebbutt, C. D, "An Expert System for Multivariable Controller Design", *submitted to Automatica*, May 1990, revised November 1990.

[T8] Tebbutt, C. D, *The CXS Expert System Shell*, 1990.

[T9] Trankle, T. L, P. Sheu, U. H. Rabin, "Expert System Architecture for Control System Design", *Proceedings of the American Control Conference*, Seatle, USA, 1986.

[V1] Vidyasagar, M, *Control System Synthesis: A Factorization Approach*, Cambridge, MA: M.I.T. Press, 1985.

[W1] Winston, P. H, B. K. P. Horn, *Lisp (2nd edition)*, Reading, MA: Addison-Wesley, 1984.

[W2] Winter, H. (ed), *Artificial Intelligence and Man-Machine Systems*, Berlin: Springer-Verlag, 1986.

[Y1] Youla, D. C, H. A. Jabr, J. J. Bongiorno, "Modern Wiener-Hopf design of optimal controllers, part II: The multivariable case", *IEEE Transactions on Automatic Control*, vol. 21, no. 3, pp. 319-338, June 1976.

[Z1] Zakian, V, L. Al-Naib, "Design of dynamical and control systems by the method of inequalities", *Proceedings of the IEE*, vol. 120, no. 11, 1973.

[Z2] Zhao, Y, H. Kimura, "Two-degree-of-freedom dead-beat control system with robustness: multivariable case", *International Journal of Control*, vol. 49, no. 2, pp. 667-679, 1989.

APPENDIX A. MV-CXS SPECIFICATIONS.

A.1. REQUIREMENTS FOR THE COMPUTER.

The MV-CXS design system runs on an IBM-PC or compatible computer, under the MS-DOS operating system (version 2.0 or later). At least 640k bytes of RAM, and a 80x87 floating point coprocessor, are required. Hercules, CGA, EGA or VGA graphics facilities are necessary to view the graphics plots.

A.2. REQUIREMENTS FOR THE PLANT.

The plant must be described by a linear, time-invariant, z domain transfer function matrix. Each element in this matrix must be a rational function with real coefficients, and must be strictly proper; the order of the numerator and denominator polynomials must not exceed 50, and they must be coprime. The plant may have up to 5 inputs and 5 outputs.

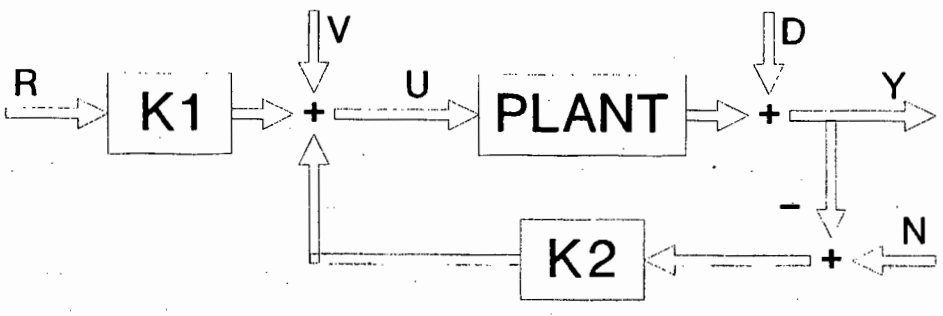


Figure A.1. Two parameter controller configuration.

If the plant is unstable, a stabilizing controller is required. This controller should be stable if possible.

MV-CXS produces a two parameter controller, with the configuration shown in figure A.1.

A.3. MV-CXS COMMAND LANGUAGE SPECIFICATION.

After the initial loading of the plant transfer function and previous design (if any), the MV-CXS design session is mainly command driven. The commands listed below may be used during the design phase.

A.3.1 Commands to select a response.

Many commands relate to a particular response. The response currently selected is shown by the command prompt. For example the prompt

```
NY[2,3](time) >
```

indicates that element [2,3] of the response to a unit step at the N input, as observed at output Y, is selected. The commands listed below are used to change the selected response.

RY, RU, NY, NU, or DY

These are used to select a closed loop response. Commands such as EDIT and PLOT, for example, refer to the currently selected response. Note that the first letter refers to the input node for the response, for example R, and the second to the output node, for example Y. The two parameter controller configuration is shown in figure A.1.

VY, or VU

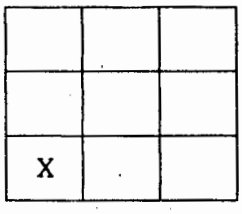
Similar to those above, except that only the frequency responses may be viewed. It is not possible to place constraints on these responses, although constraints on the singular values are allowed.

G, GK, K1, or K2

Also similar to those above, but for the open loop responses. Note that GK refers to G.K2. Step responses are not available for GK, K1 or K2.

i j

To select a specific element of a response, enter the output number (1...n), followed by the input number (1...n). For example, the element indicated with an X in the diagram below is selected using the command '3 1'.



TIME

Select the time domain (step) response. This command may be abbreviated to simply T.

FREQUENCY

Select the frequency response. This command may be abbreviated to FREQ or simply F.

A.3.2 Graphics commands.

The commands listed below are used to display many types of responses graphically. The vertical scale of the graphs may be changed by typing S once the graph has been displayed; for the Nyquist type plots, this automatically adjusts the horizontal scale also.

PLOT

Plot the currently selected response element. For example,

NY[2,3](time) >**PLOT**

will display the response at output Y2 following a unit step at input N3.

GROUP PLOT

Plot all elements of the currently selected response. The command may be abbreviated to GP.

SVD

Plot the maximum and minimum singular values of the currently selected frequency response.

NYQUIST

Display the Nyquist plot of the currently selected frequency response. The command may be abbreviated to NYQ.

NYQUIST ARRAY

Display the array of Nyquist plots for the currently selected frequency response. The command may be abbreviated to NA.

DNA

Plot the Direct Nyquist Array of the G.K2 frequency response, with Gershgorin circles.

INA

Plot the Inverse Nyquist Array of the G.K2 frequency response, with Gershgorin circles.

A.3.3 Commands to enter the specification.

The commands below usually refer to the currently selected response element, for example the DY[1,1] frequency response. Some of the commands may also relate to groups of elements; valid groups are :

ALL	all elements of the current response.
DIAG	the elements on the diagonal of the current response.
OFFDIAG	the elements not on the diagonal of the current response.
COLUMN	the elements in the current column of the current response..

EDIT [group]
Edit the performance constraints on the current response element, or group of elements. This command brings up a spreadsheet style editor. Use the cursor keys (with or without the CTRL key) to select the fields, and CTRL-Y to delete a field. When no value is given in the "value" field, the constraint is assumed void. Zero is assumed when the "from" field is blank, and the range maximum is assumed when the "to" field is blank.

EDIT SVD
As above, but for constraints on the singular values of the frequency response.

SET OPT weight [group]
Set the optimization weight for the current response element, or group of elements. In the time domain the effective cost function is

$$J = \sum_{k=0}^{N} (\text{weight} * \text{error}^2(kT))$$

where N is the number of samples. For the frequency responses the cost is approximately

$$J = \int_0^{\pi} (\text{weight} * \text{error}^2(e^{jw})) \, dw$$

In both cases, error is computed as

$$(I - h_c) \text{ or } (I - H_c), \quad c \in \{RY, NY\}$$

or

$$h_c \text{ or } H_c \text{ elsewhere.}$$

RESET OPT [group]
Cancel any optimization on the current response element, or group of elements.

OPT TYPE n
Sets the type of optimization used for subsequent optimization specifications, by modifying the weight specified in the SET OPT command. If a frequency response is currently selected, then this command applies to the optimization of frequency responses; similarly to set the type for the step responses, first select any time domain response. For time domain responses the possible types (effective weights) are :

n=1 : weight
n=2 : weight*(kT)
n=3 : weight*$(kT)^2$

where k is the sample number, and T the sample time. For the frequency domain responses they are :

n=1 : weight
n=2 : weight*w
n=3 : weight/w

where w is the argument of the frequency when mapped to the unit circle (0 to pi radians).

SHOW OPT
Display the elements of the current response which have optimization specified.

SHOW OPT ALL
Display the responses which have optimization specified.

SET ASYM [group]
Set the asymptotic properties for the current response element, or group of elements. For the RY response, the dc response will be forced to the identity matrix, and for DY to the zero matrix. Asymptotic properties may not be set on other responses.

RESET ASYM [group]
Cancel any asymptotic specification on the current response element, or group of elements.

SET ASYM TRACKING
Equivalent to SET ASYM ALL for the RY response.

RESET ASYM TRACKING
Equivalent to RESET ASYM ALL for the RY response.

SET ASYM REJECTION
Equivalent to SET ASYM ALL for the DY response.

RESET ASYM REJECTION
Equivalent to RESET ASYM ALL for the DY response.

SHOW ASYM
Display the elements of the current response which have asymptotic specifications.

SHOW ASYM ALL
Display those responses which have asymptotic specifications.

SAMPLES [n]
Set the value of the SAMPLES parameter to n. If n is omitted, display the current value of SAMPLES. Step responses are evaluated and displayed over the time period 0 to SAMPLES*T, where T is the sample time. The maximum value for SAMPLES is 100.

NFREQ [n]
Set the value of the NFREQ design parameter to n. If n is omitted, display the current value of NFREQ. NFREQ is the number of points used to evaluate frequency responses. The maximum value for NFREQ is 50.

NVARS [n]
Set the value of the NVARS design parameter to n. If n is omitted, display the current value of NVARS.

QSTEP [n]
Set the value of the QSTEP design parameter to n. If n is omitted, display the current value of QSTEP.

A.3.4 Commands to assist the designer.

The expert system is programmed with a large amount of information to assist the designer. Besides the commands listed below, the user may also type /EXPLAIN to obtain further help on the specific question being asked.

HELP
This command invokes a menu driven help system, covering the use of the design package as well as guidance for formulating the specification.

HELP topic
This gives help on specific topics. The topics available are SELECT, EDIT, OPT, ASYM, SOLVE, PLOT, SVD, NYQUIST, INA, DNA, SAVE, ESTIMATE, SAMPLES, NFREQ, QSTEP, or NVARS.

NEXT STEP
Execute a step by step design mode. This command, which may be abbreviated to NS, is used to initiate and continue the process.

SUGGEST
This command may be used to provide assistance with formulating the specification, and dealing with conflicting constraints.

SUGGEST EDIT
This command provides information on how the currently selected response may be constrained to improve the control system's performance.

SUGGEST OPT
This command provides assistance with the optimization facilities, and on how they may be used to improve the control system's performance.

COMPLETE
This command is used to help check that the design is complete.

A.3.5 Miscellaneous commands.

SOLVE
Find a controller, if possible, which meets the current specifications.

ESTIMATE n
Estimate a controller of order n. Subsequent to this command, the various plot commands will show the response using the reduced order controller. If n < 0, then revert to the full controller.

SHOW K1
SHOW K2
Display the transfer function of the estimated controller (K1 or K2 sections).

SAVE
Save the current design.

REPORT

Print a report on the design.

EXIT

Terminate the design session.

APPENDIX B. CACSD PACKAGE INTERFACE SPECIFICATION.

The interface between the expert system and the CACSD package is specified below. The CACSD package is memory resident, and is organized as a library of functions which the expert system can call upon.

The expert system uses the INTR statement to communicate with the CACSD package. This statement has the syntax

```
INTR int,P1,P2
```

where int is the interrupt number to used (96 in this application), and P1 and P2 are two integer parameters. In general, P1 specifies the function required, and P2 the sub-function. Additional data is transfered through an array of variables named T0, T1, ... T99.

Some functions refer to a specific response, for example RY[2,3](time); for these the response must first be set up using function 35. Functions with P1 ≥ 1000 use the currently selected response, and those with P1 ≥ 2000 use the current element of that response. If P1 ≥ 3000, then the appropriate element data structure is created if necessary.

Table B1 below lists the CACSD functions and their parameters. The parameters marked '=>' are sent from the expert system to the CACSD package, and those marked '<=' are returned by the package to the expert system.

P1	P2	Description
1	x	Edit matrix x (0 = plant transfer function) (1 = nominal controller txfer fn) => T0 = access code 0 = read only 1 = edit transfer functions 2 = (1) and change dimensions <= T0 = dimension
2	x	Save matrix x (x as above)
3	x	Load matrix x (x as above) <= T0 = dimension
4		Define CXS values => T0 = unknown T1 = RY T2 = RU T3 = DY T4 = NY T5 = NU T6 = time T7 = frequency T8 = max T9 = min T10 = active T11 = satis T12 = unsatis T13 = GK T14 = G T15 = K1 T16 = K2 T17 = VY T18 = VU
3005		Edit constraints <= T5 = changed (0/1) T6 = some constraints (0/1)
1006		Group plot
2006		Plot signal
7		Display free mem
8	Qn	Solve => T0 = column to solve for (1 - max) <= T0 = asymptotic cond conflict (0/1) T1 = cond conflict (0/1) T2 = solved (0/1) T3 = interrupted (0/1) T4 = asym ratio
2009		Get optimization values <= T5 = weight

3009		Set optimization values => T5 = weight
10	0 1	Direct Nyquist Array plot (of G.K2). Inverse Nyquist Array plot (of G.K2).
2011 3011		Get asymptotic value <= T5 = asymptotic value Set asymptotic value => T5 = asymptotic value
12		Save design in file project.DSN
13		Load design in file project.DSN
15		Set n_vars parameter => T0 = n_vars
1016		Display Nyquist array.
2016		Display single Nyquist plot.
17	i	Compute minimum distance from zeros of $\tilde{D}[i,i]$ to the point z = 1. <= T0 = distance
18		Estimate a controller => T0 = order <= T0 = sufficient memory (0/1)
19	0 1	Evaluate H0/1/2 using initial K0 Evaluate H0/1/2 using estimated K <= T0 = -2 insufficient memory -1 no stabilizing controller given 0 plant stable 1 right coprime fact found 2 K0 stable 3 no right coprime fact found, and K0 not stable
20		Evaluate max modulus of plant poles. <= T0 = max pole modulus
1021		Plot singular values (SVD) of response.
2022		Get condition status <= T5 = status (satis/active/unsatis/unknown)
23		Free memory allocated for controller estimation. Use full order controller.
2024		Examine response over interval => T5 = range low T6 = range high <= T7 = minimum over range T8 = maximum over range

1025 2025	0 1 0 1	Examine SVD minimum conditions set Examine SVD maximum conditions set Examine minimum conditions set Examine maximum conditions set => T5 = range low T6 = range high <= T7 = max over range T8 = min over range T9 = complete (0/1)
26		Get project name <= T0 = project name (far ptr)
27	ext	Get file name (see table B2 for .ext codes) <= T0 = filename = "project.ext" (far ptr)
28	ext	Test if file exists <= T0 = exists("project.ext") (0/1)
29		Set qstep parameter => T0 = qstep
30		Set t_sample parameter => T0 = t_sample
2031		Copy conditions on the current element => T2 = destination input T3 = destination output
32		Free memory allocated for H0/1/2 tables.
33		Check plant transfer function element => T0 = output number T1 = input number <= T2 = strictly proper (0/1) T3 = coprime (0/1)
34		Perform left MFD of plant <= T0 = (D == I) ie plant is stable (0/1) T1 = left MFD is coprime (0/1)
35		Select response as current => T0 = domain (Time/Frequency) T1 = response T2 = input number T3 = output number
36	0 1	Check that controller satisfies performance constraints on current response. Check that controller satisfies asymptotic constraints on current response. <= T0 = satisfied (0/1)
37		Check for (and then reset) memory allocation failures <= T0 = local allocation failure T1 = far allocation failure

1038	x	Edit SVD constraints (if x, edit min also) <= T5 = changed (0/1) T6 = some constraints (0/1)
1039		Test SVD constraints <= T5 = satisfied (0/1)
1040		Check if any SVD constraints <= T5 = some (0/1)
41		Set variable max_pole_mod => T0 = max_pole_mod
42	0 1	Save the Q matrix Load the Q matrix
43		Print report
44		Set parameter nfreq => T0 = nfreq
45		Set parameter samples => T0 = samples
46		Initialize Q1 and Q2 matrices.
47		Copy K2 to K0.
48	n	Set optimization type n for frequency domain responses.
49	n	Set optimization type n for time domain responses.

Table B1. CACSD functions

Code	Extension
0	.GZ
1	.K0Z
2	.PRM
3	.DSN
4	.DBF
5	.Q
6	.SPC
7	.SOL
8	.K1Z
9	.K2Z

Table B2. Filename extension codes.

APPENDIX C. MV-CXS DESIGN PROJECT HANDOUT.

USING THE CXS DESIGN PACKAGE.

An expert system design package, running under the CXS expert system, has been developed to assist the user with multivariable control system design problems. It is based upon a new design method, where specifications on the closed loop system are satisfied explicitly.

Starting the design system.

Make your own copy of the CXS disk, as your design parameters and specifications will be saved on it. Insert this disk into the A drive, and enter the following commands :

```
C:>A:
A:>HARDCOPY          (If you want to print graphics)
A:>GO
```

The system will now start loading. Do not remove the disk until the design session is over.

To print graphics screens, press the SHIFT and PrtSc keys simultaneously, and then press the 1 key. Remember that you must have run the HARDCOPY program first.

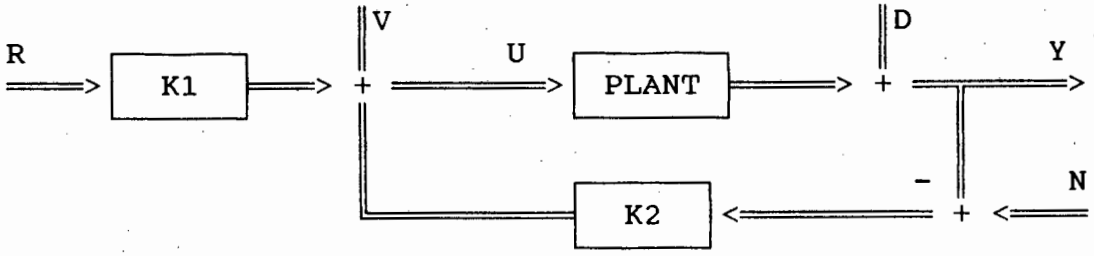
Using the expert system effectively.

The expert system is programmed with many features to assist you with your design. For example, if you do not understand a question, type **/EXPLAIN** (or **/E** for short) to get a more detailed explanation of it.

Most of the design session is driven by commands. Details of the commands available can be seen by typing the **HELP**

command. Information on a specific command, such as **EDIT**, can also be obtained by typing **HELP EDIT**.

The structure used for the control system is shown below. Nodes R, V, D and N are inputs, and U and Y are outputs. Note that N is equivalent to the command input for the standard controller configuration (ie K2 only).



The expert system refers to responses using a two letter name; for example DU indicates the response from input D to output U. In addition a particular element of the response matrix, for example [2,2], can also be selected. A typical prompt, showing the currently selected response and domain, is

RY[2,1](time) >

This indicates that commands such as **PLOT** will operate on the RY step response element [2,1] (i.e. the response to a unit step at input R1, as observed at output Y2). The current response may be changed as desired (type **HELP SELECT** for details). Note that it is also possible to select certain open loop frequency responses; these are G, GK, K1 and K2.

The overall design sequence is to enter the design specification using the **EDIT**, **ASYM**, and **OPT** commands. Then use the **SOLVE** command to find a controller (if possible) which meets these specifications. Check the responses with the **PLOT** command. If necessary, revise the specifications using the **EDIT** command again. There are two other commands to help you: **NEXT STEP**, or **NS** for short, guides you step by step in developing the specification, and **SUGGEST** can provide further suggestions on what to do.

Objectives for the design.

The design must satisfy the following specifications :

1 The step response of the diagonal elements of NY and RY (the command inputs) must rise to at least 95% within 30 seconds. The overshoot must not be more than 5%.

2 The absolute value of any of the control signals U must not exceed 3 units following a unit step input at R or N.

In addition to these, the interaction seen after step inputs at R or N should be minimized, as should the maximum amplification of disturbances at input D.

Report.

Once the design is complete, the **REPORT** command must be used to document the design specifications and performance. Be sure that the printer is ready!

IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. 35, NO. 8, AUGUST 1990

An Efficient Representation for Linear Constraints

COLIN D. TEBBUTT

Abstract—**An efficient representation for the linear constraints generated by a recent CACSD method has been developed. This reduces the storage requirements and improves the efficiency of the quadratic programming algorithm used by the design method.**

I. Introduction

The problem of designing linear controllers subject to convex performance constraints may be transformed into a linearly constrained quadratic programming problem [1]. In general, the linear constraints arise from engineering constraints on the time and frequency domain responses of the closed-loop system. The inherent structure of these linear constraints can be exploited using a special representation, leading to a reduction in the execution time and the storage requirements of the quadratic programming algorithm.

The particular quadratic programming algorithm used is a Gill-Murray active set method, as given by Scales [2]. This is a two-phase iterative algorithm; phase I searches for a feasible solution, and if successful, phase II then locates the constrained optimum of a quadratic cost function.

The algorithm performs two distinct operations on the individual constraints. The first determines if a constraint has been satisfied; this is computed for each previously unsatisfied constraint during every iteration of phase I. The second is the computation of the distance, in a specified search direction, to the constraint boundary; this must be computed for each nonactive constraint during at least half the total number of iterations. For problems with a large number of constraints and a small number of decision variables, which is typical for applications of the type described in [1], these two activities are the most time-consuming sections of the algorithm. The representation discussed below impacts on these two activities alone.

In the present implementation, the constraints are stored as a list of records. Each record is denoted by $[A, B1, B2, T]$ where $A \in R^n$, $B1 \in R$, and $B2 \in R$. T defines the constraint type.

II. Time Domain Constraints

Time domain constraints on a closed-loop step response $h(t)$ are evaluated at discrete sampling intervals. At a specific time t_1, the constraints may be posed in the form

$$k1 \le h(t_1) \le k2$$

where $k1 \in R$ and $k2 \in R$.

In some cases, only one bound is present or the bounds are equal (an equality constraint). The resulting linear constraints are in standard form, and neither requires nor benefits from any special representation. The remainder of the section will deal with the case where both bounds are present and distinct.

The step response, evaluated at time t_1, can be written in terms of the parameter vector $x \in R^n$ as

$$h(t_1) = a + b^T \cdot x$$

where $a \in R$ and $b \in R^n$. This results in the linear constraints

$$k1 - a \le b^T \cdot x \le k2 - a$$

which can be stored as the record

$$[b, k1 - a, k2 - a, \text{between}].$$

Combining both constraints into one record also gives a computational advantage since only one instead of two dot products need be computed to evaluate the pair. When computing the distance to the constraint boundary, there are additional advantages.

1) If the lower (respectively, upper) bound of the constraint is active, then the upper (respectively, lower) bound need not be checked as it cannot be violated.

2) The sign of the scalar product of the search direction with the constraint vector constant indicates directly which bound (upper or lower) needs to be checked; if the scalar product is zero, then neither bound is significant during that particular iteration.

III. Frequency Domain Constraints

Frequency domain constraints on a closed-loop transfer function $H(w)$ are approximated by evaluating H at discrete frequencies. At a specific frequency w_1, the constraint is in the form

$$|H(w_1)| \le k$$

where $k \in R$ and $k > 0$.

This complex modulus constraint is not a linear constraint; in the complex plane, the constraint boundary is a circle of radius k. However, the circle may be approximated by linear constraints as shown in [1]. The present implementation approximates the circle using 16 linear constraints, resulting in a maximum approximation error of less than 1%. For this choice, the boundary of the feasible region falls between an inner circle of radius $\sqrt{\alpha} \cdot k$ and an outer of radius $k/\sqrt{\alpha}$ where

$$\alpha = \cos(\pi/16) \approx 0.98.$$

Fig. 1 illustrates the geometry of the approximation; for the sake of clarity, the complex modulus constraint is approximated by eight linear constraints, and only one quadrant is shown.

The transfer function evaluated at a frequency w_1 can be written in terms of the parameter vector x as

$$H(w_1) = a + b^T \cdot x$$

where $a \in C$ and $b \in C^n$.

When split into separate real and imaginary components as

$$\text{Re}(H(w)) = c + d^T \cdot x$$

and

$$\text{Im}(H(w)) = e + f^T \cdot x$$

with $c \in R$, $d \in R^n$, $e \in R$, and $f \in R^n$, the complex modulus constraint may be written as

$$(c + d^T \cdot x)^2 + (e + f^T \cdot x)^2 \le k^2$$

and stored as the pair of records

$$[d, c, k, \text{complex-1}], \quad [f, e, k, \text{complex-2}].$$

This gives a great storage advantage over storing 16 such elements; if required, any of the 16 may be easily recovered as a linear combination of these two. As for the time domain case, there are computational advantages too; for evaluation purposes, only two dot products need to be computed, instead of 16. This representation does require a small overhead in tracking which of the 16 combinations are active.

The discussion below relates to a single interaction of the algorithm. The current point denotes the projection of the current parameter vector x onto the complex plane, and the search direction denotes the projection of the current search vector p. A distance β in this direction is computed in terms of the vector equation

$$x' = x + \beta \cdot p.$$

Using the geometry of the line in the search direction through the current point, the distance from the current point to the inner circle (β) and to

Manuscript received June 2, 1989; revised October 13, 1989. This work was supported in part by AECI Ltd. and the Foundation for Research Development.

The author is with the Department of Electrical and Electronic Engineering, University of Cape Town, Rondebosch 7700, South Africa.

IEEE Log Number 9035886.

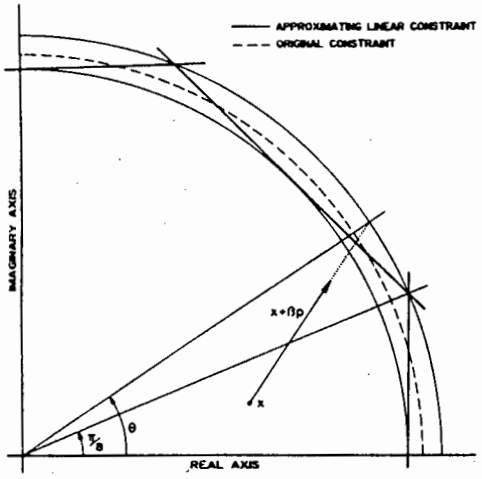


Fig. 1. Approximation of a complex modulus constraint using linear constraints.

the outer circle (β_o) are easily computed as

$$\beta_i = -g + \sqrt{(g^2 + h_i)}$$
$$\beta_o = -g + \sqrt{(g^2 + h_0)}$$

where

$$g = \frac{(c + d^T \cdot x)(d^T \cdot p) + (e + f^T \cdot x)(f^T \cdot p)}{(d^T \cdot p)^2 + (f^T \cdot p)^2}$$

$$h_i = \frac{\alpha \cdot k^2 - (c + d^T \cdot x)^2 - (e + f^T \cdot x)^2}{(d^T \cdot p)^2 + (f^T \cdot p)^2}$$

$$h_0 = \frac{k^2 / \alpha - (c + d^T \cdot x)^2 - (e + f^T \cdot x)^2}{(d^T \cdot p)^2 + (f^T \cdot p)^2}.$$

Further advantages of this representation are derived as follows.

1) If the condition

$$(c + d^T \cdot x)^2 + (e + f^T \cdot x)^2 < \alpha \cdot k^2$$

is satisfied (the current point is inside the inner circle), then all 16 approximating linear constraints are satisfied and inactive.

2) At most two of the 16 approximating linear constraints may be active at any time. In addition, if one of these constraints is active, then only the adjacent (in a circular sense) linear constraints need to be checked; the distance to the relevant constraint is β_o. All 16 constraints are also known to be satisfied if two adjacent constraints are active.

3) When β_i is real, the distance in the search direction to the nearest of the 16 approximating linear constraints is at least β_i. β_i is real if the line in the search direction through the current point intersects the inner circle. If β_i is larger than the current minimum distance to a constraint boundary, then no further checks are required on this complex constraint during the present iteration. Failing this, the particular linear constraint (one of the 16) which needs to be checked explicitly can be determined from the point of interception of the line in the search direction through the current point and the outer circle. The formula for the angle of this point in the complex plane, which determines the relevant constraint, is

$$\theta = \tan^{-1}\left[\frac{(e + f^T \cdot x) + \beta_o(f^T \cdot p)}{(c + d^T \cdot x) + \beta_o(d^T \cdot p)}\right].$$

4) When β_i is complex, it is possible to determine which of the 16 constraints is the nearest to the current point in terms of the search direction. This involves computing the tangents to the outer circle which pass through the current point. An alternative is to use these tangents as additional constraints; since they are tangent to the outer circle, they do not change the feasible set. Clearly, the distance to either tangent is zero. These features have not been implemented in the modified quadratic programming algorithm, as the computations are sufficiently complex to outweigh their advantages. However, should the number of approximating lines be significantly larger than 16, these computations will be beneficial.

IV. Examples

Two control system design problems are used to illustrate the relative efficiency of the representation described above.

Example 1: The first is the design of a two-parameter controller for an SISO plant, using a parameter vector of order 5. The plant transfer function, sampled at a rate of 1 Hz, is

$$G(z) = \frac{0.15z^{-5}}{1 - z^{-1}}.$$

The precompensator was designed using time domain constraints on the response of the plant output $y(t)$ and the control signal $u(t)$ to a unit step at the reference input; the constraints set were

$$0 \le y(t) \le 1.05, \qquad 0 \le t < 20\,\text{s},$$
$$0.9 \le y(t) \le 1.05, \qquad 20 \le t < 30\,\text{s},$$
$$0.95 \le y(t) \le 1.05, \qquad 30 \le t \le 50\,\text{s}, \text{ and}$$
$$-2.0 \le u(t) \le 2.0\ , \qquad 0 \le t \le 50\,\text{s}.$$

A quadratic cost function based on the plant output error was specified, and the constrained optimal solution was found after nine iterations. The standard quadratic programming algorithm required 194 linear constraints, and was solved in 26.9 s on an 8088-based personal computer. The modified program required 97 constraint records, and the solution was found in 15.8 s.

The feedback element was designed using frequency domain constraints on the closed-loop gain $H_{YN}(w)$ and the output disturbance response $H_{YD}(w)$ as shown below:

$$\left|H_{YN}(e^{j2\pi f})\right| \le 1.2, \qquad 0 \le f \le 0.5\,\text{Hz}$$

and

$$\left|H_{YD}(e^{j2\pi f})\right| \le 0.1, \qquad 0 \le f \le 0.004\,\text{Hz}.$$

These constraints were evaluated at 25 discrete frequencies. The optimal solution was found after 17 iterations, given a quadratic cost function based on $|H_{YN}(w)|$. The standard algorithm found the solution in 124.5 s, using 548 linear constraints. The modified program used 72 constraint records, and took 35.2 s.

Example 2: The second example is the design of a controller for an MIMO plant with transfer function

$$G(z) = \begin{bmatrix} \frac{0.1}{z - 0.9} & \frac{0.3}{z - 0.9} \\ \frac{0.2}{z - 0.9} & \frac{0.8}{z - 0.8} \end{bmatrix}.$$

Given a sampling rate of 1 Hz, the closed-loop system is required to meet the constraints

$$H_{ij}(1) = 0$$
$$\left|H_{ij}(e^{j2\pi f})\right| \le 0.25 \qquad 0 \le f \le 0.5\,\text{Hz}, i \ne j$$
$$\left|H_{ii}(e^{j2\pi f})\right| \le 1.5 \qquad 0 \le f \le 0.5\,\text{Hz}$$
$$\left|u_{ij}(t)\right| \le 5.0 \qquad 0 \le t \le 100\,\text{s}$$

for $i, j \in \{1, 2\}$, and minimize the cost function

$$J = \sum_{t=0}^{100} [0.1 * y_{11}^2(t) + y_{12}^2(t) + y_{21}^2(t) + 0.1 * y_{22}^2(t)]$$

where $H_{ij}(z)$ are the frequency responses to output disturbances, and

$y_{ij}(t)$ and $u_{ij}(t)$ are the time domain responses of the measured output and control signal, respectively, to unit step output disturbances.

The design may be split into two subproblems, which are then solved independently. For each of these, a parameter vector of order 10 was used, and the frequency domain constraints were evaluated at 100 discrete points. The subproblems were solved in 38 and 55 iterations, respectively.

For the standard algorithm, each subproblem translated into 3406 linear constraints; the optimum solutions were found in 20.1 and 32.2 s, respectively, using an 80386/387-based computer. The modified algorithm required only 504 constraint records for each subproblem, and produced the solutions in 5.3 and 6.3 s, respectively.

V. Conclusions

A quadratic programming algorithm has been modified to deal with a class of compound constraints directly. The representation discussed allows for both compact storage of the constraints and improved execution efficiency, while preserving the properties of the standard algorithm.

References

[1] S. P. Boyd, V. Balakrishnan, C. H. Barrat, N. M. Kraishi, X. Li, D. G. Meyer, and S. A. Norman, "A new CAD method and associated architectures for linear controllers," *IEEE Trans. Automat. Contr.*, vol. 33, pp. 268-283, Mar. 1988.

[2] L. E. Scales, *Introduction to Non-Linear Optimization*. New York: Macmillan, 1985.

APPENDIX E. A MICROCOMPUTER IMPLEMENTATION OF MULTIVARIABLE FACTORIZATION THEORY.

Abstract

Factorization theory forms the basis of a powerful new method for the design of linear control systems. A few techniques which simplify the implementation of this approach are examined; in particular, it is shown that under certain conditions, the multivariable design problem may be divided into a number of smaller independent sub-problems. A microcomputer design system employing these techniques has been used to solve substantial multivariable design problems.

I. INTRODUCTION

A powerful new CAD method for the design of linear control systems has been introduced recently [1]. It is based on translating the control system design problem into an approximately equivalent linearly constrained quadratic programming problem, finding the solution to this using a standard algorithm, and then translating the solution back to give the corresponding controller. The design may be specified directly in terms of constraints on the closed loop time and frequency domain responses, and a quadratic cost function to be minimized.

This design algorithm turns out to be computationally demanding in terms of both memory size and processing speed, especially when used for multivariable systems. However there are a number of techniques which, while retaining most of the strengths of the original algorithm, ease the

computational burden substantially, and make implementation on a low cost personal computer feasible.

Much of the effort of this paper is directed at reducing the dimension of the parameter vector used in the quadratic programming algorithm. This dimension impacts on the memory needed to store the linear constraints, and on the internal storage requirements of the quadratic programming algorithm, effectively limiting the size of problems which may be addressed. It will be shown that under certain conditions the multivariable design problem may be divided into a number of smaller independent sub-problems, which greatly reduces the parameter vector dimension. Further computational advantage can be obtained using the efficient representation for the linear constraints developed by Tebbutt [2]. The task of obtaining stable coprime factorizations of the plant transfer function matrix is also addressed. While some of these techniques are not applicable in every design situation, the range of designs which may be tackled remains large.

II. NOTATION

The notation used below follows that of Vidyasagar [3] closely. Let R[z] denote the set of polynomials in the indeterminate z, with real coefficients, and R(z) the field of fractions associated with R[z]. Define a subset of the complex plane C_- as a region of stability; the set $\{z: |z| < 1\}$ is often chosen, and will be assumed for the examples given later. Let C_{+e} denote the complement of this region, including the point at infinity. Let S denote the subset of R(z) comprising all rational functions analytic on C_{+e}, i.e. the set of proper stable transfer functions. S is then a commutative ring with identity, and is a domain. Let F be the field of fractions associated with S, which is also R(z).

Let M(S) denote the set of matrices with elements in S, and M(F) the set of matrices with elements in F. Let U denote

the set of units in **S**, and **U(S)** the set of unimodular matrices in **M(S)**.

This application assumes that the plant to be controlled is a linear time-invariant sampled-data process which is described by a proper z domain matrix transfer function $G(z) \in \mathbf{M(F)}$. For notational convenience the plant is assumed to be square, with n inputs and n outputs; the theory in fact readily extends to the case of non-square plants. Transfer function matrices are denoted by upper case letters, and for clarity their dependence on z is not always expressed explicitly. The individual elements of a matrix refered to by the indices [i,j], for example G[1,1].

The closed loop transfer functions are refered to as H_c, where c indicates the input and output nodes shown in figure 1. For example, H_{RY} is the closed loop transfer function from R to Y. The corresponding time domain response at the output node to a unit step at the input node is indicated by $h_c(kT)$, where T is the sample time.

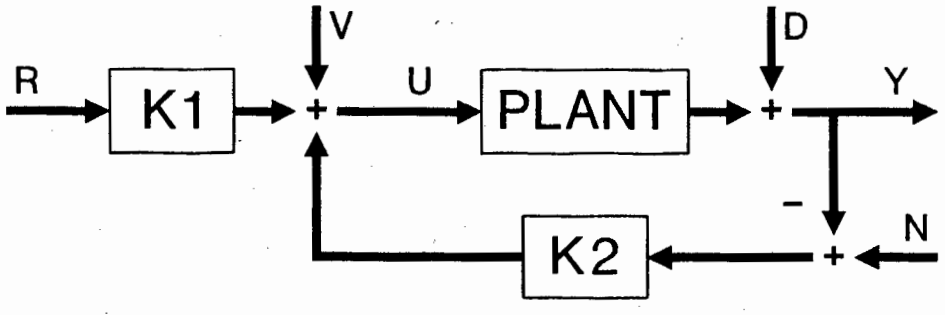


Fig 1. Two-parameter control system configuration.

III. SUMMARY OF FACTORIZATION THEORY

A brief summary of the factorization approach to the design of linear control systems is given below; for a thorough treatment of the subject see Vidyasagar [3]. This application is based upon the two-degree-of-freedom multivariable control system structure, shown in block

diagram form in figure 1.

Given left- and right-coprime factorizations $N,D \in \mathbf{M(S)}$ and $\tilde{N},\tilde{D} \in \mathbf{M(S)}$ of the plant

$$G = N.D^{-1} = \tilde{D}^{-1}.\tilde{N} \qquad (3.1)$$

there exist matrices $X,Y \in \mathbf{M(S)}$ which satisfy the Bezout Identity

$$X.N + Y.D = I. \qquad (3.2)$$

Then all internally stable closed loop transfer functions $H_C \in \mathbf{M(S)}$ may be parameterized in terms of some $Q \in \mathbf{M(S)}$ as

$$H_C = H0_C + H1_C.Q.H2_C. \qquad (3.3)$$

The transfer function matrices $H0_C$, $H1_C$ and $H2_C \in \mathbf{M(S)}$ are defined in table 1 for each of the closed loop transfer functions. Q is one of two independent parameter matrices Q1 and Q2; these are chosen by the designer to give the required closed loop performance.

Transfer functions
$H_{RY} = N.Q1$
$H_{RU} = D.Q1$
$H_{NY} = N.X + N.Q2.\tilde{D}$
$H_{NU} = D.X + D.Q2.\tilde{D}$
$H_{DY} = I-N.X - N.Q2.\tilde{D}$
$H_{DU} = -D.X - D.Q2.\tilde{D}$
$H_{VY} = N.Y - N.Q2.\tilde{N}$
$H_{VU} = D.Y - D.Q2.\tilde{N}$

Table 1. Closed loop transfer functions.

The final step of the design process is the computation of the controller from the formulae

$$K1 = (Y - Q2.\tilde{N})^{-1}(Q1)$$

and

$$K2 = (Y - Q2.\tilde{N})^{-1}(X + Q2.\tilde{D}). \qquad (3.4)$$

The matrix

$$K0 = Y^{-1}.X \qquad (3.5)$$

may be thought of as an initial stabilizing controller for the one-degree-of-freedom system shown in figure 2.

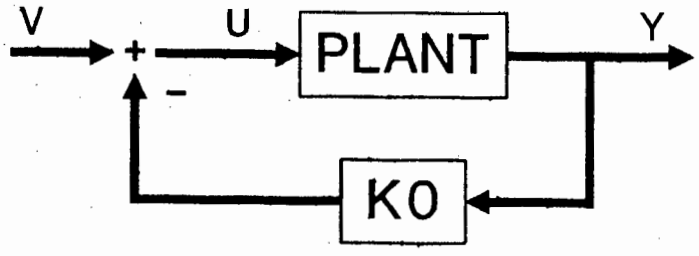


Fig 2. One-parameter control system configuration.

IV. COMPUTING THE COPRIME MATRIX FRACTIONS

For the general case, when a state-space representation of the plant is available together with stabilizing state feedback matrices, formulae for stable coprime fractions are available [4]. Zhao and Kimura [5] give formulae based on the Smith-McMillan form of the plant transfer function matrix. However neither of these sets of equations are necessarily the most convenient to use; in particular there are two special cases which lend themselves to simplified computation of coprime factorizations.

1. *Stable plant.* [6]
   In this case the factorization is trivial, and no stabilizing controller is required.

$$N = \tilde{N} = G$$

$$D = \tilde{D} = Y = I$$

$$X = 0 \qquad (4.1)$$

2. *Stable initial controller.*
   Many unstable plants can be stabilized using a stable controller $K0 \in \mathbf{M(S)}$; Vidyasagar [3] gives the conditions under which this is possible (corollary 5.3.2). In this case a right-coprime factorization $N,D \in \mathbf{M(S)}$ can be chosen as

$$N = G(I + K0.G)^{-1}$$

$$D = (I + K0.G)^{-1}$$

$$X = K0$$

$$Y = I \qquad (4.2)$$

Proof

$$N.D^{-1} = G(I + K0.G)^{-1}.(I + K0.G) = G$$

$$X.N + Y.D = K0.G(I + K0.G)^{-1} + (I + K0.G)^{-1}$$

$$= (K0.G + I)(I + K0.G)^{-1}$$

$$= I$$

As K0 is a stabilizing controller, N and D are stable, representing the closed loop transfer functions from V to Y and V to U in figure 2 respectively. Since the

Bezout Identity is also satisfied, it follows that N and D are right-coprime ([3] corollary 4.1.17).

∎

While the expressions for N and D may be quite complex, their algebraic form need not be computed explicitly. In the frequency domain these expressions are easily evaluated at discrete frequencies in terms of the transfer functions G and K0. In the time domain their impulse responses are required; these may be obtained by simulating the corresponding closed loop systems.

Similar expressions exist for stable left-coprime $\tilde{N}$ and D, and are derived in an analogous fashion; they are

$$\tilde{N} = (I + G.K0)^{-1}.G$$

$$\tilde{D} = (I + G.K0)^{-1}. \qquad (4.3)$$

V. DIAGONAL FACTORIZATION

It is interesting to note that if the matrix $H2_c$ is diagonal, then the individual elements of a specific closed loop transfer function can be written as

$$H_c[i,j] = H0_c[i,j] + \sum_{k=1}^{n} \left[H1_c[i,k].Q[k,j].H2_c[j,j] \right]. \qquad (5.1)$$

Thus column j of H_c depends only on column j of Q, and the design may be reduced from a single design problem of size αn^2 to n independent sub-problems each of size αn. Size here refers to the dimension of the search vector used by the quadratic programming algorithm. This reduction, when possible, greatly extends the scale of design problems which may be tackled given limited computer memory.

Consider a plant with 4 inputs and 4 outputs, and where each element of Q has 5 decision variables. Here the reduction

results in 4 sub-problems each of size 20, instead of a single problem of size 80. Each of the linear constraint vectors will require n times as much storage for the single problem, and there will usually be about n times as many of them, increasing the storage requirements by a factor n^2. For example, assume 1000 linear constraints are generated per sub-problem, and 8 bytes are needed per floating point number; then 1000.20.8 = 160 kilobytes are required to store the constraint vectors for each sub-problem, as opposed to 4.1000.80.8 = 2560 kilobytes for the single problem. Note that since the sub-problems are solved in independently, the constraints for each need not be stored simultaneously. Furthermore the storage requirement for the six matrices in the search algorithm is only 20^2.8.6 = 19200 bytes for each sub-problem, instead of 80^2.8.6 = 307200 bytes.

The time required to produce the final solution is usually also reduced for the partitioned problem as, although there are n problems to solve instead of just one, each one is very much simpler. An additional advantage of dividing the problem is that conflicts in the engineering specifications are generally easier to identify and resolve, as there are fewer specifications in each sub-problem. Unfortunately the partitioning scheme hinges on a diagonal $H2_c$. The circumstances under which this may be arranged are investigated next. From table 1,

$$H2_c = \begin{cases} I, & c \in \{ RY, RU \} \\ \tilde{D}, & c \in \{ DY, DU, NY, NU \} \\ \tilde{N}, & c \in \{ VY, VU \}. \end{cases}$$

The matrix I is diagonal by definition. Clearly both $\tilde{N}$ and $\tilde{D}$ cannot be diagonal simultaneously, except when the plant is diagonal; this case will not be considered further as it may be solved using standard single variable methods. Assuming that $\tilde{D}$ may be chosen to be diagonal, it is then necessary to forgo the opportunity of explicitly designing the closed

loop transfer functions H_{VY} and H_{VU}. This is considered a small sacrifice compared to the advantages of the resulting independence. By comparison many other multivariable design methods, such as the INA and characteristic loci methods, allow disturbances to be considered explicitly at either the input or the output of the plant, but not at both simultaneously.

The statements above should not be taken to imply that the designer has no control over the closed loop responses H_{VY} and H_{VU}, only that it will not be possible to tailor them *explicitly*. Since

$$H_{VY} = H_{DY}.G,$$

and $$H_{VU} = H_{DU}.G,$$

these responses may be designed, particularly in the frequency domain, by careful shaping of the H_{DY} and H_{DU}, bearing in mind the characteristics of the plant, which can be thought of as a pre-filter.

Definition.
A matrix factorization will be termed diagonal when the denominator matrix is diagonal.

The two theorems that follow are based on theorems found in Vidyasagar [3]. While he generally treats only the right-coprime case explicitly, the corresponding left-coprime forms used below follow readily. Theorem 1 is derived from problem 4.1.11 of [3].

Theorem 1
Let $G \in \mathbf{M(F)}$ have a left-coprime factorization $A,B \in \mathbf{M(S)}$. Then G + H has a left-coprime factorization A + BH,B for all $H \in \mathbf{M(S)}$.

Proof

As A and B are left-coprime, there exist $X,Y \in \mathbf{M(S)}$ such that

$$A.X + B.Y = I \tag{5.2}$$

([3] corollary 4.1.17). Define $Y' \in \mathbf{M(S)}$ as

$$Y' = Y - H.X. \tag{5.3}$$

Then

$$(A + B.H)X + B.Y' = A.X + B.Y = I. \tag{5.4}$$

Thus $(A + BH),B \in \mathbf{M(S)}$ are left-coprime ([3] corollary 4.1.17). Furthermore

$$B^{-1}(A + B.H) = G + H. \tag{5.5}$$

∎

Diagonal factorization algorithm.

Split the plant into stable and unstable components

$$G = G_s + G_u$$

such that $G_s \in \mathbf{M(S)}$, and $G_u \in \mathbf{M(F)}$ contains only the unstable poles of G. Then, if G_u has a diagonal left-coprime factorization, so does G.

A left factorization of G_u

$$G_u = \tilde{D}^{-1}.\tilde{N}_u,$$

with $\tilde{N}_u,\tilde{D} \in \mathbf{M(S)}$ and $\tilde{D}$ diagonal, may be produced as follows :

let $$G_u[i,j] = \frac{a_{ij}}{b_{ij}}, \qquad i,j \in \{1, 2, \ldots n\} \tag{5.6}$$

and $$d_i = \text{LCM } b_{ik}, \qquad k \in \{1, 2, \ldots n\} \tag{5.7}$$

where a_{ij}, b_{ij}, $d_i \in \mathbf{R}[z]$. Choose some polynomial $c_i \in \mathbf{R}[z]$ with the same order as d_i, and all of its zeros in C_-; then define the elements of the denominator matrix as

$$\tilde{D}[i,i] = \frac{d_i}{c_i} \tag{5.8}$$

Finally the numerator matrix is given by

$$\tilde{N}_u = \tilde{D}.G_u \tag{5.9}$$

Next it is necessary to determine if this diagonal factorization is left-coprime. Theorem 2 provides a simple test to establish if a matrix pair is coprime or not.

Theorem 2
Let $A,B \in \mathbf{M(S)}$ each have n rows, and let the sum of the number of columns of each be at least n. Then A and B are left-coprime if rank([A B]) = n at all points in C_{+e}.

Proof
There exists $U \in \mathbf{U(S)}$ such that

$$[A\ B]U = [R\ 0], \tag{5.10}$$

where $R \in \mathbf{M(S)}$ is a greatest common left divisor of A and B ([3] corollary B.2.15). Further there exist $X,Y \in \mathbf{M(S)}$ such that

$$A.X + B.Y = R. \tag{5.11}$$

([3] theorem 4.1.7).

Since U is unimodular, det(U) ϵ **U** ([3] fact B.1.26), and thus det(U) is nonzero at all points in C_{+e}. Thus U is nonsingular in C_{+e}, and

$$\text{rank}([A\ B]) = \text{rank}(R). \tag{5.12}$$

If rank([A B]) = n in C_{+e}, then det(R) is nonzero in C_{+e}. Note that det(R) ϵ **S** by definition of the determinant, and therefore det(R) has neither poles nor zeros in C_{+e}. Consequently det(R) ϵ **U**, R ϵ **U(S)** ([3] fact B.1.26), and R^{-1} ϵ **M(S)**.

Multiplying (5.11) on the right by R^{-1} gives

$$A.X.R^{-1} + B.Y.R^{-1} = R.R^{-1} = I. \tag{5.13}$$

As $X.R^{-1}$,$Y.R^{-1}$ ϵ **M(S)**, it follows that A and B are left-coprime ([3] corollary 4.1.17).

∎

Remarks

Although not required for the purposes of this paper, an "only if" clause for this theorem can also be proved. A similar theorem, using **R**[z] in place of **S**, is found in Kailath [7].

The application of this theorem to the left factorization described above is eased by the diagonal structure of $\tilde{D}$. [$\tilde{N}_u$ $\tilde{D}$] clearly has full rank in C_{+e}, except (possibly) at the unstable poles of G. At each of these poles z_u, full rank is possible if the rows of $\tilde{N}_u(z_u)$ corresponding to those of $\tilde{D}(z_u)$ which are now zero, are linearly independent. Stable matrices, and matrices where the unstable poles of different rows are distinct, are amongst those which have a diagonal left-coprime factorization.

As an example, consider the unstable plant

$$G(z) = \begin{bmatrix} \dfrac{-10.517}{z - 1.1052} & \dfrac{-10.517}{z - 1.1052} \\ \dfrac{-10.517}{z - 1.1052} & \dfrac{-11.070}{z - 1.2214} \end{bmatrix} z^{-2}$$

examined in [5]. Applying the technique above gives the diagonal left factorization

$$\tilde{D}(z) = \begin{bmatrix} \dfrac{z - 1.1052}{z - 0.9} & 0 \\ 0 & \dfrac{(z - 1.1052)(z - 1.2214)}{(z - 0.9)^2} \end{bmatrix}$$

$$\tilde{N}(z) = \begin{bmatrix} \dfrac{-10.517}{z - 0.9} & \dfrac{-10.517}{z - 0.9} \\ \dfrac{-10.517(z-1.2214)}{(z - 0.9)^2} & \dfrac{-11.070(z-1.1052)}{(z - 0.9)^2} \end{bmatrix} z^{-2}$$

Evaluating $[\tilde{N}\ \tilde{D}]$ at the unstable poles $z = 1.1052$ and $z = 1.2214$ gives

$$\begin{bmatrix} -41.96 & -41.96 & 0 & 0 \\ 23.76 & 0 & 0 & 0 \end{bmatrix}$$

and

$$\begin{bmatrix} -21.93 & -21.93 & 0.36 & 0 \\ 0 & -8.34 & 0 & 0 \end{bmatrix}$$

respectively. Both of these matrices have full row rank, and thus $\tilde{N}$ and $\tilde{D}$ are left-coprime.

VI. THE QSTEP PARAMETER

A result from factorization theory is that all stable closed loop transfer functions may generated by equation (3.3) for some stable transfer function matrix $Q \in \mathbf{M(S)}$; when the domain of Q is restricted, this fact no longer holds. Unfortunately some restriction is inevitable when Q is represented on a finite computer.

To produce an acceptable engineering solution it is generally not essential that all possible transfer functions be generated; a representative range suffices. Consider a typical computer representation of real numbers, where both the range of numbers, as well as the precision of the representation, is limited; yet for most problems this set of values available is adequate. Similarly it is required that the set of possible values for Q spans an adequate subset of all stable transfer function matrices, and with adequate precision.

The design method of Boyd [1] is based upon a finite impulse response (FIR) representation of Q; in essence the method requires a matrix of proper stable polynomial ratios, where the denominator polynomials are fixed, and the coefficients of the numerator polynomials are determined by the search algorithm. To what extent does this representation of Q approximate the set of all stable transfer functions? For low order filters, it would seem rather poorly.

A clue to the physical meaning of Q is obtained from the case where the plant is stable. Choosing the factorization of (4.1) gives

$$H_c = Q, \quad c \in \{RU, NU\}.$$

Here Q is required to represent the closed loop transfer functions to the plant input. In the time domain it is clearly seen that the order of the FIR filter marks a time window over which control actions, following a disturbance

impulse, may be taken. A high order filter is therefore necessary if a 'slow' control is desired. Unfortunately the dimension of the search vector in the quadratic programming algorithm is directly proportional to the order of the FIR filter, which makes the use of high order filters prohibitive on a small computer. In order to gain the computational advantages of using a low dimension search vector, and yet retain some of the benefits of a high order FIR filter, the following parameterization is proposed :

$$Q[i,j](z) = q_{0,ij} + \sum_{k=1}^{p-1} \left[q_{k,ij} \sum_{r=1}^{QSTEP} z^{-(r + (k-1)*QSTEP)} \right] \quad (6.1)$$

where $q_{k,ij}$, $k \in \{0, 1, \ldots p-1\}$, are the coefficients (decision variables) of the new filter Q[i,j]. QSTEP is a positive integer which effectively stretches the FIR filter; the standard FIR form results when QSTEP is unity. This parameter gives the designer a further degree of freedom; generally QSTEP is chosen in the light of the required speed of response (relative to the sampling rate).

The merit of the QSTEP parameter is most clearly illustrated by a single-input single-output example. Consider the plant, sampled at a rate of 1 Hz, with transfer function

$$G = \frac{0.1}{z - 1} z^{-6}$$

and initial stabilizing controller

$$K0 = 0.1$$

Let the design problem be the minimization of the cost function J based on the response to a unit step disturbance at the plant output,

$$J = \sum_{k=0}^{50} \left[h_{DY}(kT)^2 \right]$$

subject to the constraint on the control signal

$$| h_{DU}(kT) | \leq 1.0, \qquad k = 0, 1, \ldots 50$$

Table 2 lists the minimum value found for the cost function J for different combinations of the number of decision variables p and the value of QSTEP. Using 5 decision variables, the minimum cost is obtained with QSTEP set at 4, and this cost is only slightly higher than that of the best long FIR filter. Figure 3 shows the step responses for the cases (a: p=5, QSTEP=1), (b: p=5, QSTEP=4) and (c: p=25, QSTEP=1). Note that linear interpolation is used between the output values at the sampling instants in this and the other graphs.

p	QSTEP	Cost
5	1	10.498
5	2	10.023
5	3	9.865
5	4	9.853
5	5	9.894
9	1	9.957
13	1	9.850
17	1	9.850
25	1	9.850

Table 2. Design cost for various values of p and QSTEP.

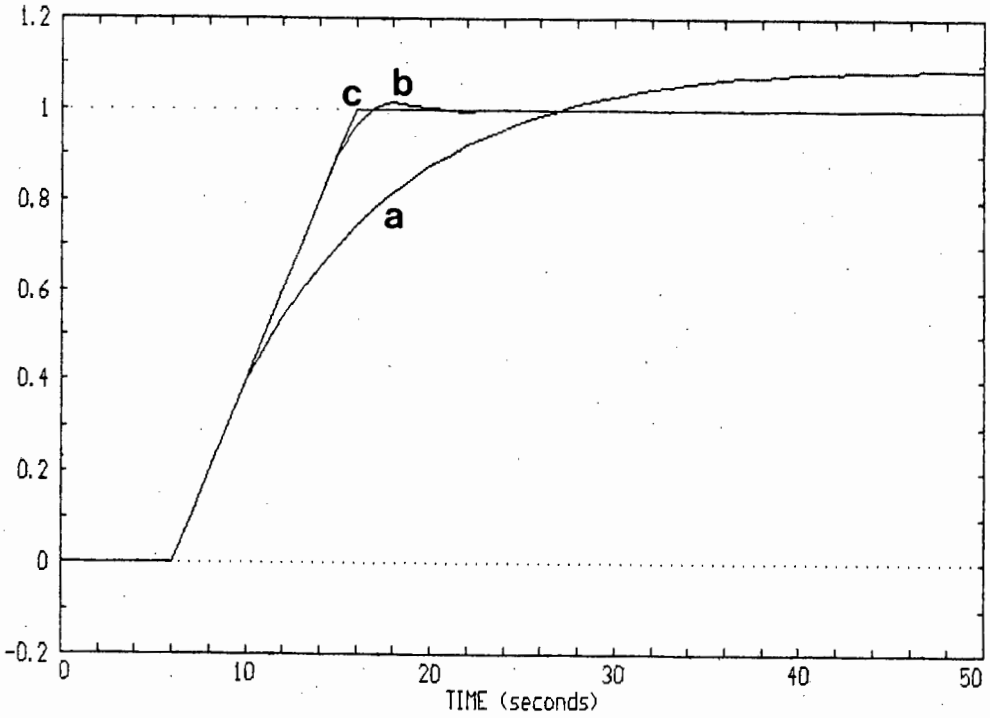


Fig 3. Step responses for various values of p and QSTEP.

VII. MULTIVARIABLE DESIGN EXAMPLE

An interactive design system similar to that described by Boyd [1], but incorporating the features described above, has been implemented on an 8088/87 based personal computer. The system has been programmed for plant matrices with up to 5 inputs and outputs, and a maximum search vector dimension of 25. This permits up to 5 decision variables per element of Q for plants of maximum dimensions, and up to 25 for SISO plants. Frequency response conditions are evaluated at 50 points logarithmically spaced on the unit circle; each resulting complex modulus constraint is approximated using eight linear constraints.

The design system has been used to design a controller for the flotation plant simulator. Discretizing the plant at a sampling rate of 0.2 Hz (T = 5 seconds) gives

$$G = \frac{1}{100}\begin{bmatrix} \frac{8.622z-8.428}{z^2-1.708z+0.7240} & \frac{0.6650z^{-4}}{z-0.9708} & \frac{-5.336z-3.284}{z^2-1.172z+0.2290} & \frac{-0.3460}{z-0.9614} \\ \frac{-6.235}{z-0.8618} & \frac{8.007}{z-0.9533} & \frac{-2.360z^{-4}}{z-0.9316} & \frac{-0.1403}{z-0.9628} \\ \frac{3.636z-3.612}{z^2-1.905z+0.9071} & \frac{4.655z^{-6}}{z-0.9834} & \frac{30.39z-30.25}{z^2-1.739z+0.7475} & \frac{-2.780}{z-0.8963} \\ \frac{(8.298z-8.186)z^{-5}}{z^2-1.884z+0.8869} & \frac{9.509z^{-14}}{z-0.9843} & \frac{(57.36z-56.10)z^{-2}}{z^2-1.743z+0.7597} & \frac{0.6820z-0.6756}{z^2-1.810z+0.8187} \end{bmatrix}$$

Since this matrix is stable, the diagonal coprime factorizations (4.1) are trivial. The design thus may be split into eight independent sub-problems, relating to the four columns of each of Q1 and Q2. The cost functions $J_{Qi,j}$ for these sub-problems, chosen to give good setpoint tracking and disturbance rejection, were

$$J_{Q1,j} = \sum_{k=0}^{50} \sum_{i=1}^{4} \left[a_{i,j}(I[i,j] - h_{RY}[i,j](kT))^2 \right]$$

$$J_{Q2,j} = \sum_{k=0}^{50} \sum_{i=1}^{4} \left[a_{i,j}(h_{DY}[i,j](kT))^2 \right]$$

where

$$a_{i,j} = \begin{cases} 0.1 & i = j \\ 1 & i \neq j. \end{cases}$$

Asymptotic tracking and disturbance rejection requirements were specified using the constraints

$$H_{RY}(1) = I$$

and

$$H_{DZ}(1) = 0.$$

Further constraints were placed on the control signals in the time domain, to limit the control action following a

unit step input. These were

$$| h_{RU}[i,j](kT) | \leq b_{i,j} \qquad k = 0, 1, \ldots 50$$

and

$$| h_{NU}[i,j](kT) | \leq b_{i,j} \qquad k = 0, 1, \ldots 50$$

where

$$b_{i,j} = \begin{cases} 4 & (i,j) = (4,3) \\ 2 & \text{elsewhere.} \end{cases}$$

Finally, to ensure low frequency disturbance rejection and robust stability to high frequency modelling errors, the constraints

$$| H_{DY}[i,j](e^{j2\pi fT}) | \leq 0.1 \qquad 0 \leq f \leq 0.0005\text{Hz}$$

and

$$| H_{NY}[i,j](e^{j2\pi fT}) | \leq 0.1 \qquad 0.05\text{Hz} \leq f \leq 0.1\text{Hz}$$

were included. The design system found a controller satisfying the constraints, using 5 decision variables per element of Q; setting QSTEP at 10 gave the lowest cost function values. Table 3 lists the results for each of the eight sub-problems. The column labeled 'Constrs' indicates the number of linear constraints used to approximate the problem, 'Its' the number of iterations of the quadratic programming algorithm, and 'Time' the time in seconds taken to solve the quadratic programming problem using a 4.77 MHz 8088/87 based computer. 'Cost' lists the final values of the cost functions. Figure 4 shows the closed loop step responses of h_{RY}, and figure 5 those of h_{NY}. The minimum and maximum singular values of H_{DY} and H_{NY} are plotted in figures 6 and 7 respectively. The total time to solve this problem, including the translation of the engineering specifications into linear constraints, was approximately 23 minutes; Using 6 decision variables per element of Q, with QSTEP set at 9, reduced the overall cost from 3.943 to 3.323 (a 15% improvement), but required an additional 18 minutes for the complete solution. A 25 MHz 80386/387 computer performed these designs approximately twelve times faster.

Problem	Constrs	Its	Time	Cost
Q1 / 1	412	12	15	0.541
Q1 / 2	412	11	11	0.515
Q1 / 3	412	11	10	0.420
Q1 / 4	412	12	16	0.297
Q2 / 1	956	111	467	0.675
Q2 / 2	956	136	353	0.558
Q2 / 3	956	67	171	0.565
Q2 / 4	956	61	160	0.372

Table 3. Results from the multivariable design example.

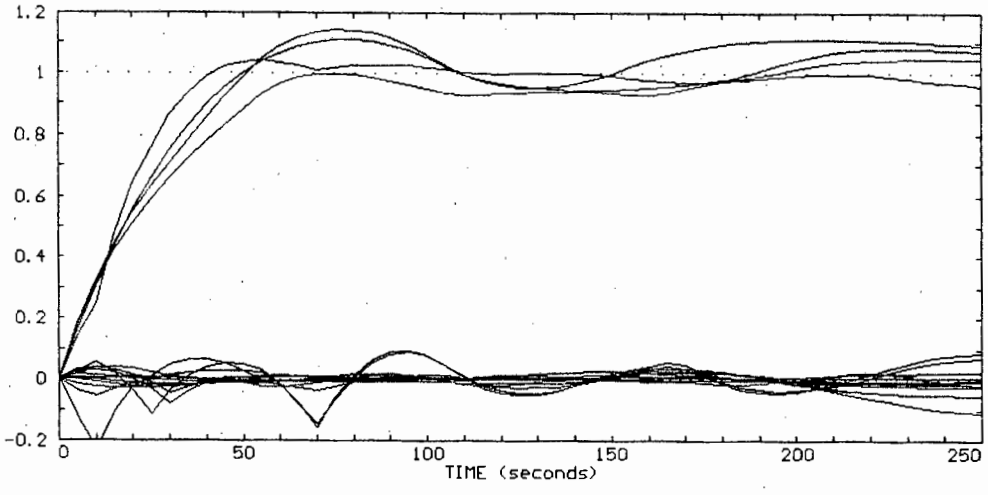


Fig 4. Step responses of h_{RY}.

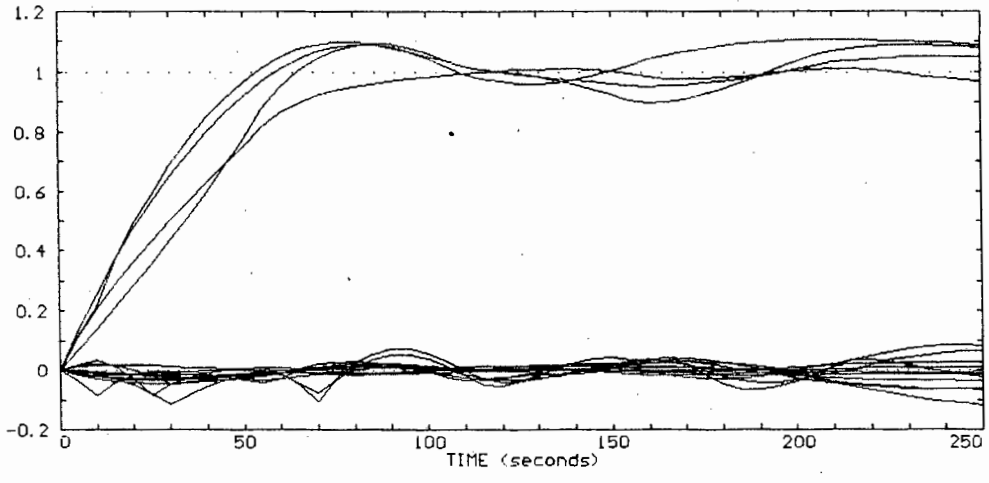


Fig 5. Step responses of h_{NY}.

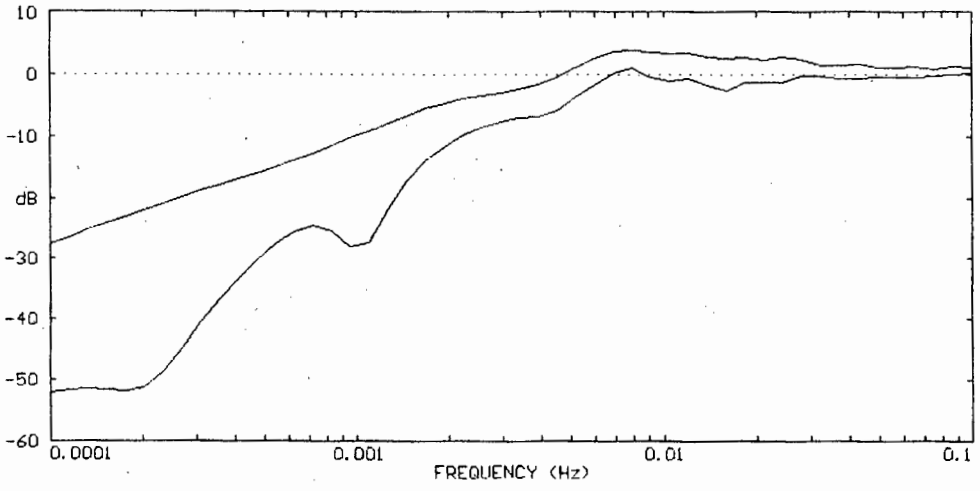


Fig 6. Max/min singular value plot of H_{DY}.

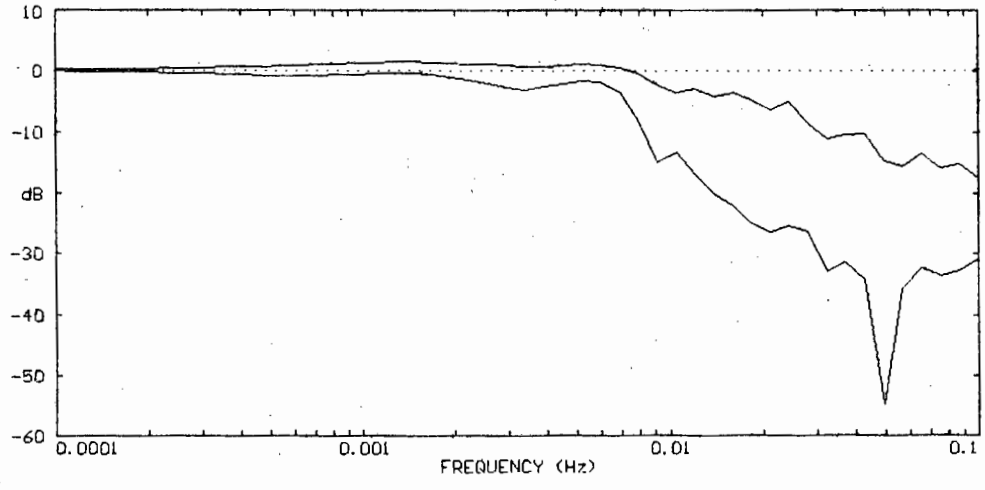


Fig 7. Max/min singular value plot of H_{NY}.

VIII. CONCLUSION

A few modifications to the CAD method of Boyd [1] have been used to enable medium sized multivariable control system design problems to be solved using a microcomputer. The same techniques could similarly enable large scale problems to be tackled on a more powerful workstation. Although the diagonal factorization technique cannot be applied to all plant transfer function matrices, many are amenable to it.

One of the principal drawbacks of this approach is the form and complexity of the resulting controller; Boyd [1] describes a special purpose architecture for addressing this

problem. Another deficiency is that only constraints which are convex with respect to the closed loop system can be treated directly by this method. Some non-convex constraints, for example gain and phase margins, may be satisfied indirectly by judicious choice of the closed loop responses. Despite these drawbacks, the CAD method remains powerful and yet easy to use.

REFERENCES

[1] S. P. Boyd, V. Balakrishnan, C. H. Barrat, N. M. Kraishi, X. Li, D. G. Meyer, S. A. Norman, "A New CAD Method and Associated Architectures for Linear Controllers," *IEEE Trans. Automat. Contr.*, vol. AC-33, no. 3, pp. 268-283, March 1988.

[2] C. D. Tebbutt, "An Efficient Representation for Linear Constraints", *IEEE Trans. Automat. Contr.*, vol. AC-35, no. 8, pp. 949-951, August 1990.

[3] M. Vidyasagar, *Control System Synthesis: A Factorization Approach*, Cambridge, MA: M.I.T. Press, 1985.

[4] C. N. Nett, C. A. Jacobson, M. J. Balas, "A Connection Between State-Space and Doubly Coprime Fractional Representations," *IEEE Trans. Automat. Contr.*, vol. AC-29, no. 9, pp. 831-832, September 1984.

[5] Y. Zhao, H. Kimura, "Two-degree-of-freedom dead-beat control system with robustness: multivariable case", *Int. J. Control*, vol. 49, no. 2, pp. 667-679, 1989.

[6] C. L. Gustafson, C. A. Desoer, "Controller design for linear multivariable feedback systems with stable plants, using optimization with inequality constraints", *Int. J. Control*, vol. 37, no. 5, pp. 881-907, 1983.

[7] T. Kailath, *Linear Systems*, Englewood Cliffs, N.J.: Prentice-Hall, 1980.

[8] C. C. MacDuffee, *Theory of Matrices*, New York: Chelsea, 1946.

An expert systems approach to controller design

C.D. Tebbutt

Corrected proof copy.

Indexing terms: Algorithms, Computer-aided design, Control systems, Mathematical techniques

Abstract: An expert system has been used to build an intelligent interface to a recently developed CACSD technique for the design of linear controllers. This combination allows the design to be specified in terms of time and frequency domain constraints on the closed-loop transfer functions, and provides the designer with assistance in formulating a reasonable specification and in dealing with constraint conflicts.

1 Introduction

The 1980s have seen much interest in the application of expert system techniques to the problem of designing control systems [1–4]. In general, applications have consisted of an underlying numerical design technique, for example LQG, with an expert system program forming the interface between it and the designer.

With any man/machine interface there is an enginering tradeoff between designing the interface for 'expert' users and designing it for novice or 'less-than-expert' users. Expert users tend to dislike being constantly reminded of obvious little details; however, they are not necessarily frequent users of the system and so may need some help with the design language syntax. A novice designer, on the other hand, usually appreciates as much assistance as possible. This assistance should go much deeper than simply explaining the design language; it may include suggestions for setting a realistic specification, an explanation of, and help with, each step in the particular design process, and checking that the final design is adequate. A novice should be expected to come away from such a design session having learned something. Expert systems have shown much promise in dealing with the problem of presenting a large amount of information when required, without obstructing an experienced user.

Some applications have taken a classical, or at least a strongly human-oriented, design technique and have used the expert system to simulate human thought processes. James *et al.* [1] is a good example of this type. Birdwell [2] and Birdwell *et al.* [3] have shown how an expert system may be used to accelerate the design cycle time for a relatively complex design method. Expert systems may also be used to plug the holes in a conventional design algorithm; the designer may be warned of design aspects not explicitly covered by the particular algorithm, and given advice on how to account for them.

This application is based on a numerical design technique that is fundamentally computer-oriented, where the control system design problem is transformed into a linearly constrained quadratic programming problem, which may then be solved using a standard algorithm. An expert system is used to coordinate the transformation process and then to interpret the numerical results for the user. Representing the design in terms of constraints also proves useful in that it allows the expert system to analyse the design specifications readily. It can determine which design factors have not yet been considered, and infer reasons for conflicts in the engineering specifications.

In the present implementation the design technique is applicable to SISO plants which can be described adequately by a linear, time invariant, z-domain transfer function. In this paper the constraint transformation process is described briefly, followed by details of the expert system. Possible modifications and extensions to the system are discussed later.

2 Transforming the design into a quadratic programming problem

The performance specifications for a control system are often set as constraints in the time and/or frequency domains. For example, a design may require a rise time of at most 0.2 s with no more than 10% overshoot, and at least 20 dB rejection of disturbances at the plant output for frequencies up to 3 Hz. It turns out that these and many other design constraints may easily be transformed into a list of linear constraints on a parameter vector, suitable for solution by a standard quadratic programming algorithm; a quadratic cost function to be minimised may also be specified. The strength of the quadratic programming algorithm is that it always finds the global optimum solution subject to the constraints within a finite number of iterations, if the problem is feasible; if there is no feasible solution, this will also be determined within a finite number of iterations.

Vidyasagar [5] provides a good description of the details of the factorisation for the general MIMO case, and Boyd *et al.* [6] discuss the subsequent transformation into linear constraints. The important steps in this process for an SISO plant are summarised below.

The system shown in Fig. 1 is partitioned into the transfer function matrices P_{zw}, P_{zu}, P_{yw}, and P_{yu}, with

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} P_{zw} & P_{zu} \\ P_{yw} & P_{yu} \end{bmatrix} \cdot \begin{bmatrix} w \\ u \end{bmatrix} \tag{1}$$

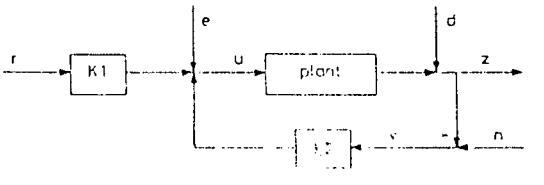


Fig. 1 *Control system structure*

Paper 7543D (C8), received 17th May 1989

The author is with the Department of Electrical and Electronic Engineering, University of Cape Town, Rondebosch, 7700, South Africa

and

$$u = [K_1 \quad K_2] \cdot y \tag{2}$$

where

$w = [r \quad n]^T$ are exogenous inputs
$y = [r \quad v]^T$ are measured outputs
u is the control input
z is the regulated output

For simplicity, exogenous inputs d and e shown in Fig. 1 are not included in the derivation that follows. Transfer functions such as H_{zd} are, however, easily be derived from H_{zw}. Note that r is used both as an exogenous input and a measured output.

Left $(\tilde{D}^{-1}, \tilde{N})$ and right (N, D^{-1}) stable coprime factorisations may then be found for P_{yu} as

$$P_{yu} = \begin{bmatrix} 0 \\ -G \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & G_D/\alpha_p \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ -G_N/\alpha_p \end{bmatrix} = \tilde{D}^{-1} \cdot \tilde{N} \tag{3}$$

$$= \begin{bmatrix} 0 \\ -G_N/\delta_p \end{bmatrix} \cdot [G_D/\alpha_p]^{-1} = N \cdot D^{-1} \tag{4}$$

where

$G = G_N/G_D$ is the plant transfer function
α_p is a stable polynomial of order equal to that of G_D.

A stable polynomial is one whose zeros lie inside the unit circle in the complex plane. Note that for a SISO plant the factorisation amounts to no more than choosing a stable α_p, assuming that G_N and G_D are coprime.

The next step is to find a nominal controller K_{nom} which stabilises the plant, with a stable left factorisation

$$\begin{aligned} K_{nom} &= [K_{nom1} \quad K_{nom2}] \\ &= [Y/\alpha_c]^{-1}[X_1/\alpha_c \quad X_2/\alpha_c] \\ &= Y^{-1} \cdot X \end{aligned} \tag{5}$$

where α_c is a stable polynomial of order equal to that of Y, and which also satisfies the equation

$$-X \cdot N + Y \cdot D = 1$$

or

$$-[X_1/\alpha_c \quad X_2/\alpha_c] \cdot \begin{bmatrix} 0 \\ -G_N/\alpha_p \end{bmatrix} + [Y/\alpha_c] \cdot [G_D/\alpha_p] = 1$$

or

$$X_2 \cdot G_N + Y \cdot G_D = \alpha_p \alpha_c \tag{6}$$

This subproblem may be inverted; by specifying a stable α_p and α_c, values may be obtained for X_2 and Y using a standard pole-placement algorithm to solve eqn. 6. An arbitrary polynomial polynomial of order no more than that of Y, for example Y itself, may then be specified for X_1 This technique guarantees that K_{nom} is a stabilising controller.

The closed-loop transfer function can then be written as

$$H_{zw} = T_1 + T_2 \cdot Q \cdot T_3 \tag{7}$$

$$Q = [Q_1 \quad Q_2] \tag{8}$$

where Q_1 and Q_2 are arbitrary stable transfer functions, and

$$T_1 = P_{zw} + P_{zu} \cdot D \cdot X \cdot P_{yw} = G_N[X_1 \quad X_2]/\alpha_p\alpha_c \tag{9}$$

$$T_2 = P_{zu} \cdot D = G_N/\alpha_p \tag{10}$$

$$T_3 = \tilde{D} \cdot P_{yw} \begin{bmatrix} 1 & 0 \\ 0 & G_D/\alpha_p \end{bmatrix} \tag{11}$$

The elements of vector Q are defined as

$$Q_1 = \beta_1/\alpha_q \tag{12}$$

$$Q_2 = \beta_2/\alpha_q \tag{13}$$

where α_q is a stable polynomial of order $n-1$, for example z^{n-1}. It is shown later that n is the dimension of the decision vector. β_1, β_2 are polynomials of the form

$$\beta_i = \beta_{i,0} + \beta_{i,1} z + \cdots + \beta_{i,n-1} z^{n-1} \tag{14}$$

Substituting for T_1, T_2, T_3 and Q in eqn. 7 gives

$$H_{zw} = \frac{G_N}{\alpha_p} \left[\frac{X_1}{\alpha_c} + \frac{\beta_1}{\alpha_q} \quad \frac{X_2}{\alpha_c} + \frac{G_D\beta_2}{\alpha_p\alpha_q} \right] \tag{15}$$

The controller is then computed from the equation

$$\begin{aligned} K &= [K_1 \quad K_2] \\ &= [\alpha_p\alpha_q Y - \alpha_C G_N \beta_2]^{-1} \\ &\quad \times [\alpha_p\alpha_q X_1 + \alpha_c\alpha_p\beta_1 \quad \alpha_p\alpha_q X_2 + \alpha_c G_D\beta_2] \end{aligned}$$

The important factorisation result is that all stable closed-loop systems for the plant may be generated using eqn. 7, with some stable Q. In this application Q is restricted to a finite-order polynomial ratio, with a fixed denominator; if this order is large a wide range of stable transfer functions may be generated. The design is carried out directly on the transfer functions H_{zr}, H_{ur}, H_{zn}, H_{un}, H_{zd}, H_{ud}, and H_{ze}, which are derived directly from H_{zw}. Each of these can be written as the sum of two terms, the first independent of β, and the second with β appearing directly as a factor. For example,

$$\begin{aligned} H_{zr} &= \frac{G_N X_1}{\alpha_p\alpha_c} + \frac{G_N}{\alpha_p\alpha_q}\beta_1 \\ &= H_a + H_b\beta_1 \end{aligned} \tag{17}$$

This can be expanded as

$$\begin{aligned} H_{zr} = H_a &+ H_b \cdot \beta_{1,0} + H_b z \cdot \beta_{1,1} \\ &+ \cdots + H_b z^{n-1} \cdot \beta_{1,n-1} \end{aligned} \tag{18}$$

giving a linear combination of $n+1$ transfer functions, with n degrees of freedom (decision variables). Similar expressions are easily derived for H_{zn}, H_{wd}, etc.

In the time domain, the inverse transform h_{zr} of H_{zr} may similarly be evaluated at a particular time t, giving

$$h_{zr}(t) = k' + k_0 \cdot \beta_{1,0} + k_1 \cdot \beta_{1,1} + \cdots + k_{n-1} \cdot \beta_{1,n-1} \tag{19}$$

where

k' and k_i, $i = 0, 1, \ldots, n-1$, are real constants

Thus a condition such as $p1 \leqslant h_{zr}(t) \leqslant p2$, $t_1 \leqslant t \leqslant t_2$, may be tested at each sample time in $[t_1, t_2]$. Evaluating at each of these times then gives a set of conditions in the form

$$\begin{aligned} p1 \leqslant k' + k_0 \cdot \beta_{1,0} + k_1 \cdot \beta_{1,1} \\ + \cdots + k_{n-1} \cdot \beta_{1,n-1} \leqslant p2 \end{aligned} \tag{20}$$

or

$$p1' \leqslant k_0 \cdot \beta_{1,0} + k_1 + \cdots + k_{n-1} \cdot \beta_{1,n-1} \leqslant p2' \tag{21}$$

which gives two linear inequality constraints.

A condition such as $|H_{zr}(w)| \leqslant p3$, $w \in W$, where W is some frequency range, may be approximated by testing at a finite number of frequency points on W. When

evaluated at a specific frequency w,

$$H_{zr}(w) = c' + c_0 \cdot \beta_{1,0} + c_1 \cdot \beta_{1,1} + \cdots + c_{n-1} \cdot \beta_{1,n-1} \tag{22}$$

where

c' and c_i, $i = 0, 1, \ldots, n-1$, are complex constants

which results in a set of conditions in the form

$$|c' + c_0 \cdot \beta_{1,0} + c_1 \cdot \beta_{1,1} + \cdots + c_{n-1} \cdot \beta_{1,n-1}| \leqslant p3 \tag{23}$$

Clearly these are not linear constraints; drawn on the complex plane the constraint boundary is a circle. However, it is possible to approximate the circular area using a number of linear constraints [6]. In the present implementation this circle is approximated using 16 linear equations, with a maximum approximation error of about 1%.

Equality constraints may also be set on time and frequency responses. These are particularly useful for the specification of asymptotic properties. For example, asymptotic tracking is ensured if the constraint $H_{zr}(0) = 1$ is satisfied.

For optimisation purposes, the cost function to be minimised must be specified in terms of a quadratic equation in the form

$$J(\beta) = \beta^T \cdot A \cdot \beta + B^T \cdot \beta + C \tag{24}$$

where

A is a real-valued square matrix
B is a real-valued vector
C is a real scalar
β is the decision vector

In the time domain, a cost function specified as

$$J = \sum (h(t) - c)^2 \tag{25}$$

is easily transformed into the format of eqn. (24) as

$$\begin{aligned} J &= \sum (k' + k_0 \cdot \beta_0 + k_1 \cdot \beta_1 + \cdots + k_{n-1} \cdot \beta_{n-1} - c)^2 \\ &= \sum (K^T \cdot \beta + c')^2 \\ &= \sum (\beta^T \cdot k \cdot k^T \cdot \beta + 2c' \cdot k^T \cdot \beta + c'^2) \end{aligned} \tag{26}$$

In the frequency domain, a cost function in the form

$$J = \int |H(w)|^2 \, dw \tag{27}$$

may be approximated by evaluating the integral at discrete frequency points. This then gives the sum

$$\begin{aligned} J &\approx \sum (|c' + c_0 \cdot \beta_0 + c_1 \cdot \beta_1 + \cdots + c_{n-1} \cdot \beta_{n-1}|)^2 \\ &= \sum (|c' + c^T \cdot \beta|)^2 \\ &= \sum \{(\mathrm{Re}\,(c' + c^T \cdot \beta))^2 + (\mathrm{Im}\,(c' + c^T \cdot \beta))^2\} \\ &= \sum \{\beta^T \cdot \mathrm{Re}\,(c) \cdot \mathrm{Re}\,(c)^T \cdot \beta + 2\,\mathrm{Re}\,(c') \cdot \mathrm{Re}\,(c)^T \cdot \beta \\ &\quad + (\mathrm{Re}\,(c'))^2\} + \{\beta^T \cdot \mathrm{Im}\,(c) \cdot \mathrm{Im}\,(c)^T \cdot \beta \\ &\quad + 2\,\mathrm{Im}\,(c') \cdot \mathrm{Im}\,(c)^T \cdot \beta + (\mathrm{Im}\,(c'))^2\} \end{aligned} \tag{28}$$

Both of these cost functions may easily be scaled by a time/frequency dependent weighting function. Note that these forms also guarantee that the A matrix will be at least positive semidefinite; it will in fact be positive definite for all but extreme pathological cases where the particular response is independent of one or more elements of β. Hence, there is always a finite value for β giving the minimum cost, and this is almost always unique. These properties also hold when the domain of β is restricted by a set of convex constraints, and both the time and frequency domain constraints described above are convex.

A very interesting feature of this Q parameterisation is that the transfer functions H_{zr} and H_{ur} are designed independently of the H_{zn}, H_{un}, H_{zd}, H_{ud}, and H_{ze} transfer functions [5]. It turns out that the transfer functions H_{zr} and H_{ur} depend on Q_1 alone, and the others depend only on Q_2. In effect, it means that the prefilter $K1$ is designed independently of the loop compensator $K2$. This in turn gives a great computation advantage when compared to the prospect of having some transfer functions dependent on both Q_1 and Q_2.

Having transformed the engineering specifications into a list of linear constraints, the optimum feasible solution may be sought using a quadratic programming algorithm. The algorithm used in the present implementation is a Gill-Murray active set method, as described in Scales [7].

3 Why use an expert system?

Although the technique of transforming the engineering specification into a quadratic programming problem is extremely powerful, it is only one step in the overall design process. There are many issues which need to be addressed in order to apply it successfully, and it is here where the expert system approach proves very useful.

The structure of the intelligent design system as implemented is shown in Fig. 2. The numeric modules include

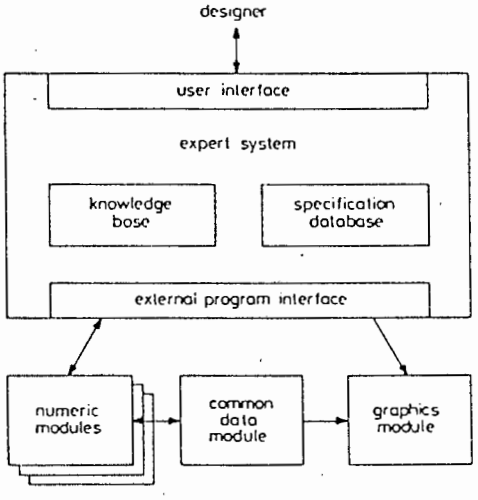


Fig. 2 *Structure of the design system*

the quadratic programming algorithm, the code to transform the engineering specifications into linear constraints and a quadratic objective function, and the software for computing an initial stabilising controller. The graphics module contains the code to display step and frequency responses, including a Nyquist plot. A further module stores data common to the numeric and graphics modules, such as the plant transfer function. These modules are not suitable for direct inclusion in the expert system; they have been implemented as external programs which are invoked as required by the expert system.

Of the remaining tasks, the most important are the user interface, the coordination of the numeric and graphic modules, recommending design features for the

user's consideration, and explaining any constraint conflicts. These functions could conceivably also be programmed in a standard language; however, the advantages offered by the expert systems approach make it an excellent choice for this application.

Expert system techniques deal effectively with the problem of complexity management. In this application, as with many artificial intelligence problems, there is a complex decision-making structure and a large amount of knowledge. Expert systems offer an effective means of representing decision-making knowledge in terms of rules and facts. It is easy to extend the knowledge base as experience with the system grows, and this may usually be done without disturbing the structure of the knowledge already present. Expert system shells usually offer good online debugging aids; for example, one may trace the reasoning process to determine how a value was inferred for a particular variable. Debugging facilities are a vital component of any complex software project.

4 Knowledge and information representation

The heart of an artificial intelligence program is its knowledge representation structure. The CXS expert system shell used is a rule-based expert system; almost all of the knowledge is programmed into a set of production rules. Since the function of these rules is information analysis, it is important for this information to be stored in an easily accessible form. CXS offers two data types: the symbol-value type, and the database type.

Simple facts are represented using the symbol-value data type. For example, the symbol INP_DIST_REJ_REQD may have a value of YES, NO or UNKNOWN representing whether rejection of disturbances at the plant input is required or not. Numeric values may also be assigned to these symbols.

The database type allows a record structure to be specified and many such records to be stored. The expert system has commands to search through the database, and update, insert or delete records. This data structure is ideally suited for the representation of the engineering constraints. For these, a database is defined with the structure

SIGNAL	the transfer function, e.g. H_{zn}
DOMAIN	the time or frequency domain
VALUE	the constraint value
TYPE	the type of constraint, e.g. MIN or MAX
LOW	the lower end of the domain range
HIGH	the upper end of the domain range

Thus an engineering constraint such as a maximum overshoot of 5% during the time interval from 0 to 10 s is represented by the record [H_{zr}, Time, 1.05, Max, 0.0, 10.0].

Not all of the knowledge is stored in the expert system; the plant transfer function, for example, is more conveniently stored in the numeric software modules. Since there is an efficient interface to these modules, the expert system has ready access to this knowledge. Often such knowledge transfer takes place at an abstract level; for example, the expert system does not want to know the actual plant transfer function, but needs to know if it is coprime or not. CXS provides two mechanisms for communication with external programs. The first is used to execute disk-resident programs; data transfer may take place through files. The second is a message passing scheme intended for memory-resident programs, and the messages contain a command code and a data block. The second type has been used extensively in the present implementation.

5 Advising the designer

Perhaps the major contribution of the expert system in assisting the novice designer is that of providing advice on formulating a good specification for the design.

The expert system knowledge base is programmed with a list of design features which are usually considered by expert designers. The list contains features such as a robust stability requirement, and limits on control signals. Each feature has been programmed as a rule, with the following generic form:

```
IF this advice has not previously been given
AND the feature has not yet been accounted for
AND the feature is probably required
THEN
   Display the advice.
   Note that this advice has now been given.
END
```

A simple search through the specification database is all that is usually required to check that a particular feature has been accounted for. For example, when checking that robust stability of the closed-loop system has been specified, the expert system considers the frequency domain constraints that have been set on the H_{zn} signal. If this signal has not been limited over the entire frequency range, then this feature has probably not yet been considered, and merits the designer's attention.

The information necessary for the expert system to determine which features are required in the design must usually be extracted directly from the designer. Often it suffices to ask a question of the form 'Is feature x required?', although even in a case as simple as this the knowledge base should consider the possibility that the user is unable to answer yes or no. There are some instances when the expert system may infer whether or not a feature is required; for example, if the plant contains an integrator, or if asymptotic rejection of disturbances at the plant input has already been specified, it is not necessary to specify asymptotic rejection of disturbances at the plant output.

Each design feature in the list is identified as being of primary or secondary importance. When the designer attempts to solve for a controller, the primary features in the list are used to check that the design is complete in a minimal sense. The designer may also explicitly ask the system for suggestions on what to do next. In this case the complete list of features is used to recommend the most appropriate next step.

A typical rule from the knowledge base is given below. This rule detects that the designer has not yet specified optimisation on any of the transfer functions which affect the selection of $Q1$; clearly some objective function should be given if all the degrees of freedom in the design technique are to be used.

```
IF suggestn10 is unknown
AND NOT FIND graphdb[*,*,*,1,"q1"]
THEN
 suggestn10 is "done"
   suggestion is "done"
   DISPLAY 8
```

No optimization has been specified for either of the RZ or RU signals. You may wish to optimise the step response of the RZ signal, or trade it off against that

of the RZ signal. Select the desired signal and domain, and then use the command 'SET OPT n' to specify the optimisation. If you type 'HELP OPT' you will be given more information on the optimisation process.
END

6 Explaining constraint conflicts

The expert system has been programmed to deal with conflicts in the engineering constraints. Specifications which on their own are unreasonable in the sense that no stabilising controller could satisfy them are also considered here. A number of superficial checks on the engineering constraints are performed by the expert system. For example, a rise time less than the dead time in the plant will not be achievable by any causal controller, and is detected directly when specified.

More important, however, are the constraint conflicts which are identified by the quadratic programming algorithm. In [6] the quadratic programming algorithm was used to give a solution/no-solution answer to a single list of linear constraints. For the present implementation the linear constraints are grouped according to the engineering constraints they represent, and the standard quadratic programming algorithm has been modified to deal with groups of constraints. A solution is sought in stages, with a new group being added at each stage until it is determined that no feasible solution exists, or that all the groups have been included. Although this strategy is slightly less efficient when there is a feasible solution, it provides useful information when none exists. In this latter case the expert system is able to analyse the status (satisfied, active, or not satisfied) of each group attempted in order to gain an understanding of the conflicts in the engineering constraints. To suggest a suitable remedy, the expert system also uses knowledge of the order in which the groups are added, and in some cases knowledge of the plant characteristics and the engineering specifications. This analysis is coded in the knowledge base as a set of rules, each with the form

```
IF group n is not satisfied
AND groups 1, 2, ..., n − 1 are active, satisfied, ...
AND plant has characteristics a and b
AND specification is c or d
THEN
   suggest remedy x for the problem
END
```

A typical rule from the section of the knowledge base which analyses constraint conflicts is shown below. The value for the expert system symbol nzf_stat, indicating the status of the frequency domain constraints on the H_{zn} transfer function, is obtained from one of the numerical software modules.

```
IF nzf_stat is "unsatisfied"
AND cz_asym is number
OR nz_asym is number
OR dz_asym is number
AND low_freq_phase < −145
OR low_freq_phase > 0
THEN
   explain_q2 is "done"
   DISPLAY 8
```

The Nyquist stability requirement (frequency domain constraints on the NZ transfer function) cannot be met. It is possible that this is caused by a combination of poor low frequency plant response and the asymptotic values specified. You should consider removing the asymptotic specifications; otherwise you must relax the frequency domain specifications on the NZ signal.
END

7 Other tasks managed by the expert system

Taylor and Frederick [8] list ten distinct control engineering activities, ranging from the modeling of the plant to be controlled through to the implementation of the final design. Any of these could be approached from an expert system perspective, and with its inherent flexibility an expert system is well suited for integrating many of them into a comprehensive design tool. It has already been shown how the expert system is used to assist in the formulation of the design, and in checking that it is reasonable. The expert system has also been used to cover other design activities.

In the total design process, an important task is that of finding a representative and usable plant model; in this case the model is specified in terms of a linear, z-domain, transfer function. Although the expert system presently offers no assistance in obtaining the model, it does perform a range of checks on the model. For example, the model must be coprime (controllable and observable); the system can cancel stable pole-zero pairs if required.

In the present implementation the initial stabilizing controller is found using a pole-placement algorithm. Since the closed-loop poles are specified explicitly, it is a trivial matter to ensure that they all lie within a restricted region of the unit circle; the effects on the final design of specifying a restricted region are far from trivial by contrast. The expert system allows the designer to define the stability region by specifying a maximum modulus and minimum damping factor for the closed-loop poles. If the values given for these are quite limiting, and a subsequent failure to find a solution could possibly be attributed to this choice, the designer is suitably warned.

Expert systems offer a versatile and yet consistent user interface. An explanation facility is usually built into this interface which, despite its simplicity, is very useful for guiding inexperienced users. The expert system does not assume that the user will be able to answer all the questions posed; often the knowledge base may be programmed to induce the answer from other facts, or at least guess a suitable default value.

8 Sample design session

An extract from a design session is given below. The units used for frequency domain specifications are hertz and decibels; for time domain specifications they are seconds and units (where a step input of 1 unit is used). Commands which are entered by the user appear in bold type, and explanatory comments added are in italics.

What is the sampling time ? **0.1**
The plant is acceptable.
The Nyquist frequency is 5 Hz.
You must now enter your design, using the SET commands. When it has been entered, the SOLVE command will make the system search for a controller that satisfies the specifications, if one exists. For more details on these and the other commands, type / EXPLAIN or HELP. The prompt displays which response will be affected by the command, eg

'NZ,TIME>' indicates that the command relates to the time domain of the signal from input N to output Z.
NZ,FREQUENCY > **solve**
[*The SOLVE command initially checks that the design meets some minimal requirements.*]
Is the plant transfer function an EXACT model of the real plant ? **no**
Do you have a model for the plant uncertainty ? **/explain**
Assume we can write the model uncertainty in the form

$$G^{\circ} = (1 + Gd)G$$

where G° is the true plant transfer function, G is the transfer function of the model used for the design. Then Gd represents the transfer function of the error between the two. If you have a model for Gd, answer YES to this question.
Do you have a model for the plant uncertainty ? **no**
Do you wish to specify an M-circle condition ? **yes**
What is the M-circle value ? **1.5**
An equivalent way of stating an M-circle requirement of 1.5 is to use the command 'SET MAX 3.521827', where the limit has been computed as 20*log10(mcircle value). Enter this command if required.
NZ,FREQUENCY > **set max 3.52**
[*Constrain the maximum value of the NZ frequency response to 3.52 dB.*]
NZ,FREQUENCY > **solve**
Is asymptotic tracking of the reference signal required ? **yes**
The command 'SET ASYMPTOTIC 1' for the RZ signal has been entered for you. This forces the DC gain from R to Z to be unity, and thus achieve asymptotic tracking.
Is rejection of low frequency disturbances at the plant input important ? **yes**
Do you require asymptotic rejection of disturbances at the plant input ? **yes**
The command 'SET ASYMPTOTIC 0' for the EZ signal has been entered for you. This forces the DC gain from E to Z to be zero, and thus achieve asymptotic disturbance rejection. In effect this means that the controller will contain an integrator.
No specifications for the RZ transfer function have been given, and thus there is no tracking specification. I would suggest that you specify optimisation for the RZ step response (time domain), or at least set minimum rise time and maximum overshoot constraints.
RZ,TIME > **set opt**
What type of optimisation do you require ? **/explain**
There are 4 types of optimisation available for time domain signals
0 : No optimisation
1 : Integral of error^2
2 : Integral of t.error^2
3 : Integral of t^2 .error^2
The default is type 1
What type of optimisation do you require ? **1**
What is the optimisation weight ? **1**
RZ,TIME > **solve**
[*At this point the numerical software is called to find a controller, and it is successful.*]
RZ,TIME > **suggest**
Is rejection of low frequency disturbances at the plant input important ? **no**
A low frequency limit should be set on the EZ signal to ensure disturbance rejection. A starting specification could be 'SET MAX −20 TO 0.025'.
EZ,FREQUENCY > **set max −20 to .05**
[*Constrain the maximum value of the EZ frequency response to −20 dB over the range 0 to 0.05 Hz.*]
EZ,FREQUENCY > **suggest**
Do you require the control signal to be limited ? **yes**
Neither time nor frequency domain limits have been set for the NU signal. You should check the graphs for these using the PLOT command, and if necessary set constraints on at least one of them. Generally a high frequency limit, or bounds on the initial part of the time response, will be required, as real control actuators (eg valves, motors etc) have a limited output range.
NU,TIME > **plot**
NU,TIME > **frequency**
NU,FREQUENCY > **plot**
NU,FREQUENCY > **set max 12 from 1**
[*Constrain the maximum value of the NU frequency response to 12 dB over the range 1–5 Hz.*]
NU,FREQUENCY > **suggest**
No optimisation has been specified for any of the signals NZ, DZ, EZ, or NU. You may wish to optimise the response of the NZ signal, or some of the disturbance responses. Select the desired signal and domain, and then use the command 'SET OPT' to specify the optimisation. If you type 'HELP OPT' you will be given more information on the optimisation commands.
NU,FREQUENCY > **ez**
EZ,FREQUENCY > **time**
EZ,TIME > **set opt**
What type of optimisation do you require ? **1**
What is the optimisation weight ? **1**
EZ,TIME > **solve**
[*This time the search for suitable a controller is not successful.*]
The input disturbance rejection specification cannot be met, as it conflicts with the constraints on the NU signals, and the frequency domain constraints on NZ. You will need to relax at least one of these sets of constraints.

.

.

[*The designer continues to modify the specification until a satisfactory and achievable design is produced.*]

9 Modifications and extensions

To modify this design technique for use in the *s*- instead of *z*-domain is not difficult. The stability region will be defined differently, although the concepts behind its definition remain the same. For frequency domain conditions the set of frequencies used for evaluation will lie on the imaginary axis of the complex plane instead of on the unit circle. For time domain constraints integration of the differential equations is required instead of summation of the difference equations; this will increase the computational load marginally.

The design system may also be extended to include a wider range of convex engineering constraints. Boyd *et al.* [6] provide details of some of these. Unfortunately not all types of engineering constraints are convex constraints on the closed-loop transfer functions. It may, however, be possible to program the expert system to deal with some of these constraints, for example gain and phase margins

The expert system could possibly determine convex constraints to approximate these and to revise the approximations as necessary each time a new controller is computed.

The changes to the numerical software required to design controllers for MIMO plants are not expected to be conceptually difficult, although the computational load will rise significantly as vector and matrix dimensions increase. The step where left and right coprime factorisations of P_{yu} are found will be more complex, as will finding a nominal stabilising controller. The modification of the expert system layer is where the real challenge will lie, and this is the subject of ongoing research.

10 Conclusions

A criticism of expert systems, and artificial intelligence in general, voiced by Dreyfus and Dreyfus [9] is that, while they have been successfully applied in a microworld situation, they are not adequate to address real-world problems. This criticism should not be dismissed lightly, and control system design is most certainly a complex real-world task requiring a great deal of intelligence. However, by restricting the problem to that of designing a controller for a plant model, as opposed to real problem of designing it for the physical plant, a microworld situation is effectively created.

Denham [10] notes that engineering design is a combination of creativity and the selection of the best design alternatives. The study of artificial intelligence has made little progress towards creative programs; certainly the expert system makes no attempt to supplant the designer in this role. The best that it can aim for is to provide a flexible design environment which will stimulate instead of stifle natural creativity. It has been shown how the expert system can offer assistance in the selection of design alternatives.

The linear constraint design technique effectively allows the user to concentrate on designing the specification. By augmenting this technique with an expert system it has in addition been possible to address a wider range of design activities, and provide an intelligent user interface.

11 Acknowledgment

Financial support for this research from AECI Ltd. and the Foundation for Research Development is gratefully acknowledged.

12 References

1 JAMES, J.R., FREDERICK, D.K., and TAYLOR, J.H.: 'Use of expert-system programming techniques for the design of lead-lag compensators', *IEE Proc. D, Control Theory & Appl.*, 1987, **134**, (3), pp. 137–144

2 BIRDWELL, J.D.: 'An expert system can aid in the evolution of a design methodology', *Proceedings of the American Control Conference*, 1987, pp. 541–546

3 BIRDWELL, J.D., CHANG, G., HANSON, P., HONG, L., LAI, J.-S., and MURPHY, G.V.: 'Teaching with the CASCADE computer-aided control system design environment'. 19th Southeastern Symposium on Systems Theory, Clemson, SC, 1987

4 PANG, G.K.H., and MACFARLANE, A.G.J.: 'An expert systems approach to computer-aided design of multivariable systems' (Springer-Verlag, 1987)

5 VIDYASAGAR, M.: 'Control system synthesis: a factorization approach' (M.I.T. Press, 1985)

6 BOYD, S.P., BALAKRISHNAN, V., BARRAT, C.H., KHRAISHI, N.M., LI, X., MEYER, G., and NORMAN, S.A.: 'A new CAD method and associated architectures for linear controllers', *EEE Trans.*, 1988, AC-33, (3), pp. 268–283

7 SCALES, L.E.: 'Introduction to non-linear optimization' (Macmillan Publishers Ltd., 1985)

8 TAYLOR, J.H., and FREDERICK, D.K.: 'An expert system architecture for computer-aided control engineering', *Proc. IEEE*, 1984, **72**, (12), pp. 1795–1805

9 DREYFUS, H.L., and DREYFUS, S.E.: 'Mind over machine' (The Free Press, 1986)

10 DENHAM, M.J.: 'Design issues for CACSD systems', *Proc. IEEE*, 1984, **72**, (12), pp. 1714–1723

APPENDIX G. USE OF AN EXPERT SYSTEM FOR CONTROLLER DESIGN.

Presented at the 1989 RUGSA Symposium on the Applications of Knowledge-based Systems in Engineering and Control.

ABSTRACT

The design of control systems is a complex task requiring a combination of knowledge and engineering judgement. In particular the designer needs a good understanding of the design technique being used, and an ability to recognize and evaluate engineering tradeoffs. Traditional CAD tools have not addressed these issues directly.

An intelligent CAD tool has been built which combines an expert system with a recently developed numerical design technique. The design is specified directly in terms of time and frequency domain constraints on the closed loop transfer functions, and a cost function to be minimized. The expert system assists the designer in formulating a comprehensive and achievable specification, and in dealing with constraint conflicts. It is also used to coordinate the mechanics of the design technique.

This design tool is currently being used by a group of students as part of their control system design project. The paper will document the experience gained from this exercise in terms of the student's reactions to the tool, and will include a performance comparison of the controllers designed using it against those designed by classical techniques. It will also discuss the issues involved in implementing such a tool, the strengths and weaknesses of this approach, and the prospects for artificial intelligence methods in computer aided design and education for control engineering.

INTRODUCTION

The human interface to a design system is usually aimed at either "expert" or novice users. Expert users tend to dislike having familiar concepts explained; however they are not necessarily frequent users and so may require some help with the design language. A novice designer, on the other hand, usually appreciates as much assistance as possible. This assistance should go much deeper than simply explaining the design language; it may include suggestions for setting a realistic specification, an explanation of, and help with, each step in the particular design process, and checking that the final design is adequate. A novice should be expected to come away from such a design session having learned something. Expert systems have shown much promise in dealing with the problem of presenting a large amount of information when required, without obstructing an experienced user.

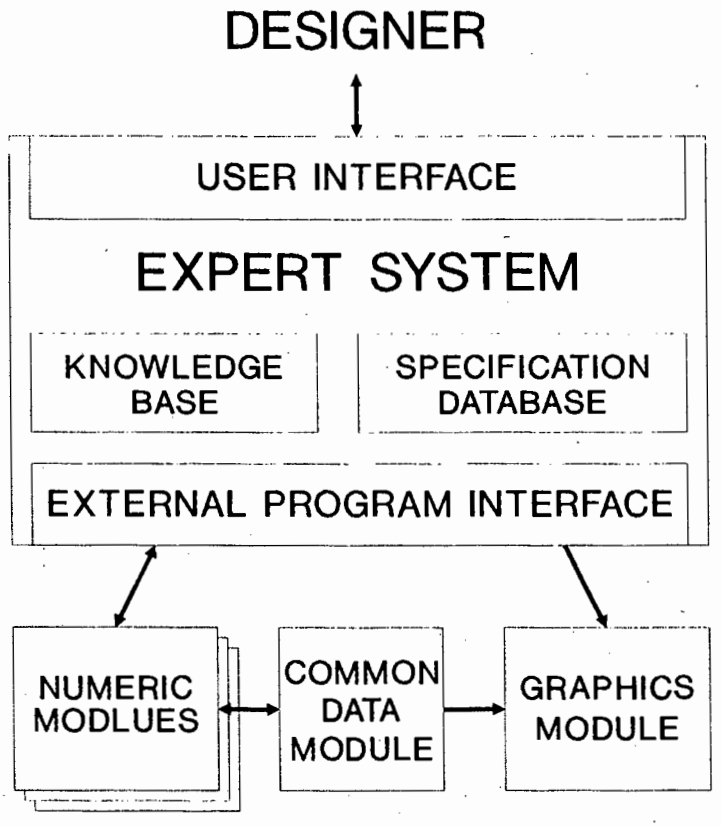


Figure 1. Structure of the CXS design package.

The 1980's have seen many applications of expert system techniques to the problem of designing control systems [1-5]. This application is based on a numerical design technique where the control system design problem is transformed into a linearly constrained quadratic programming problem, which is then solved using a standard algorithm [6]. An expert system is used to coordinate the transformation, and interpret the numerical results for the user. The overall structure of the design system is shown in figure 1. The design is specified directly in terms of time and frequency domain constraints on the closed loop transfer

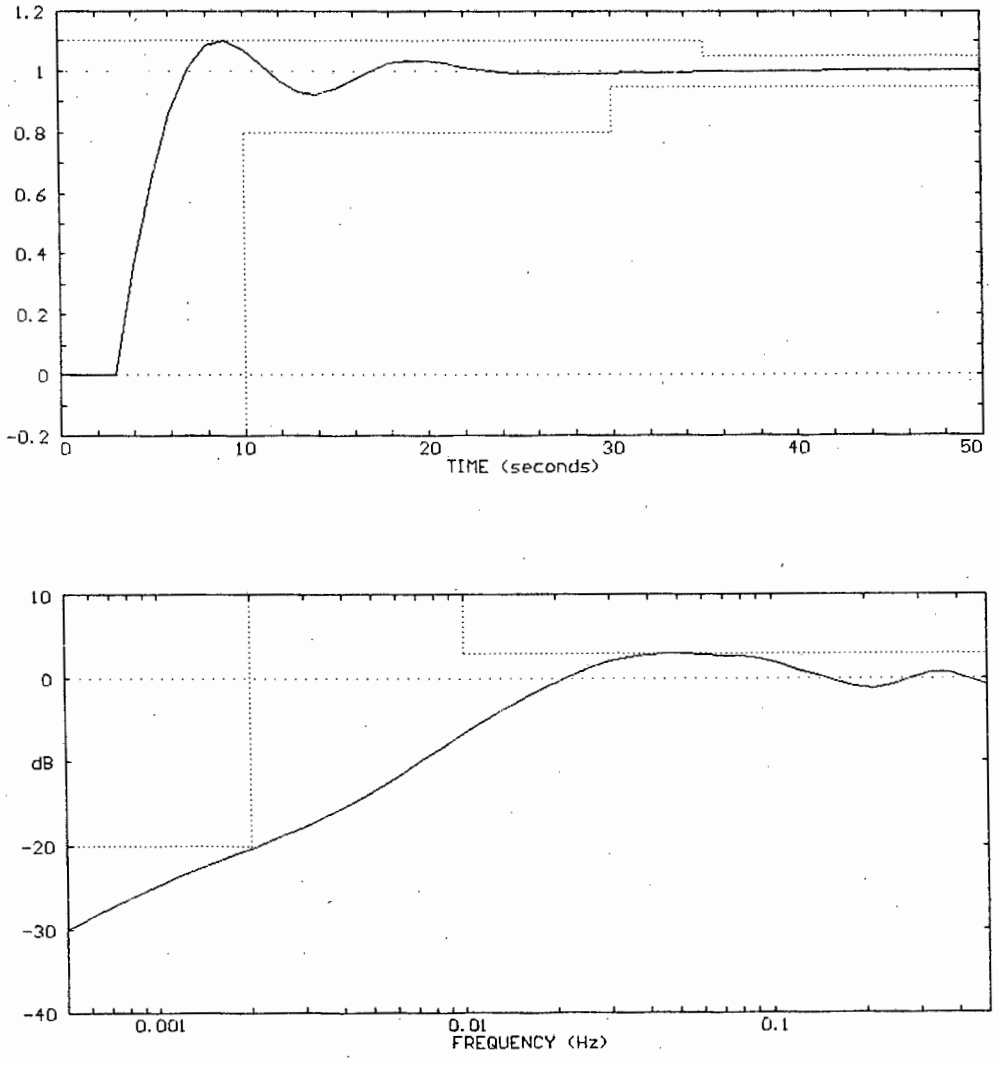


Figure 2. Typical time and frequency domain constraints.

functions, and (optionally) a cost function to be minimized. For example, a design may require a rise time of at most 20 seconds with no more than 5% overshoot, and at least 20dB rejection of disturbances at the plant output for frequencies up to 0.005 Hz. Typical constraints are illustrated in figure 2.

EXPERT SYSTEM TASKS

Although the numerical design technique described is extremely powerful, it forms only one part of the total design process. There are many other issues to be considered, and the expert system proves very useful in addressing these. Apart from coordinating the functions of the numeric and graphic modules, the most important tasks performed by the expert system are :

1. Providing a versatile and yet consistent user interface. Incorporated into this interface is an explanation facility which is very useful in guiding inexperienced users.

2. Checking that the plant model is usable. In particular the plant transfer function must be causal and coprime (controllable and observable); the system will optionally cancel stable pole-zero pairs.

3. Checking and explaining conflicts in the engineering specification. This includes preliminary checks on the specifications for conflicts such as a specified rise time less than the plant dead time, or conflicts between the asymptotic properties and general frequency domain specifications. More important is the explanation of conflicts identified by the quadratic programming algorithm. In this case the source of the conflict is explained to the user, together with suggestions for dealing with it.

4. Providing advice for the designer. Representing the design in terms of constraints also proves useful in that it allows the expert system to analyse the design specifications readily. It can determine which design features have not yet been considered, and remind the user of their importance where necessary.

THE FOURTH YEAR CONTROL DESIGN PROJECT

The project for the fourth year control class was the design of a level controller for the sump of a milling process. This was modelled by a transfer function of the form

$$G(s) = \frac{A}{s} e^{-Bs}$$

where the parameters A and B were assigned different values for each student; typical values were 0.2 and 4.0 respectively. The parameters were also modified slightly each time the simulation was run. Disturbances were added at the input of the plant, and comprised a dc offset, a relatively large disturbance at 0.016Hz plus a number of harmonics, and high frequency noise. Since each plant had a different gain, the level of the disturbances as seen at the plant output was different for each student.

The project was divided into a number of phases. One of the early phases included modelling the plant from input-output data. For the design stages, however, the students were given the nominal s domain transfer function.

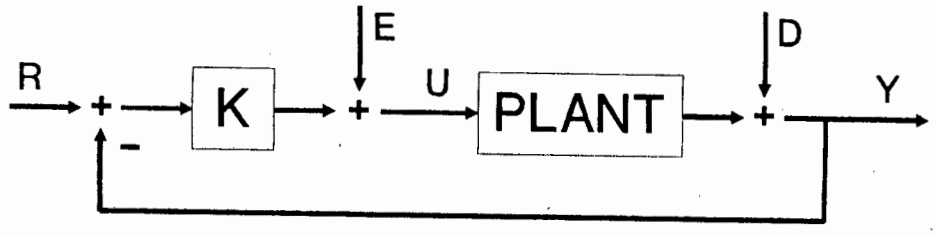
The students performed two designs for this problem. The first was done using the classical methods as taught in class, and generally resulted in a PI controller. For the second design the CXS expert system was used, with very little additional tuition. It is however probable that significant "peer tuition" took place during both designs. The instructions given to the students for the CXS section are reproduced in appendix A; this includes a sample design session using the expert system. Appendix B tabulates the

controller performance data for the two design approaches. The comparison below is based on the designs of the students who completed both design sections; students who made gross errors in computing the z domain transfer function for the plant were excluded.

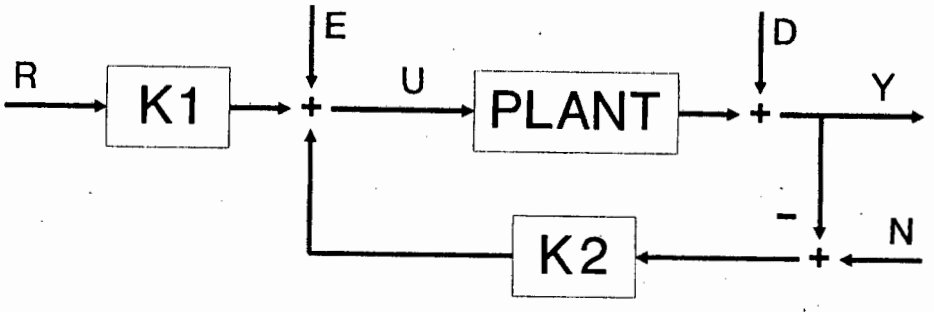
The performance of a controller for this problem can be judged in terms of its ability to track the setpoint and reject disturbances. Additional features which should also be considered are the robustness of the controller, and the demands placed on the actuator. The engineering tradeoff is between a slow, robust, low performance controller, and a sensitive high performance controller.

To compare the response to setpoint changes, the time to settle to within 5% following a step input was taken as the measure of performance. The columns labelled "RY" in appendix B contain these settling times. In 89% of the designs the CXS controllers exhibited better responses to setpoint changes; the average settling time was 16.2 seconds compared to 63.4 seconds for the classical controller. This was to be expected from the advantage of using a two-parameter control structure over the PI configuration (figure 3), which makes the comparison unfair in a sense. However it should be remembered that the students would not have been able to design a suitable pre-filter without the expert system.

More important are the disturbance rejection properties. The parameter chosen for the comparison here was the frequency below which disturbances at the plant output were rejected by at least 3 dB; the data is listed in the columns labeled "DY3db" of appendix B. The CXS controllers performed better in 70% of the designs in this regard; the average frequency was 0.0146 Hz for the classical design, against 0.0187 Hz for the CXS controller, an improvement of 28%. In general the gain in low frequency disturbance rejection was obtained at the expense of increased control effort.



Standard controller configuration



Two-parameter controller configuration

Figure 3. Two-parameter controller configuration vs PI controller's.

Unfortunately almost half of the CXS designs suffered from excessive controller gain and/or amplification of high frequency disturbances. Appendix B lists the maximum amplification of disturbances under the columns labeled "DYmax"; this parameter is also a good measure of the controller robustness. The column "NUmax" gives the peak value of the control signal U following a unit step at input N or D; this gives a measure of the controller gain. The failures were due in part to the expert system not adequately warning of the deficiencies in the design, and in part to the students ignoring the advice that was given. In many cases there was a deliberate decision to tradeoff almost everything for low frequency disturbance rejection.

ANALYSIS OF THE EXPERT SYSTEMS APPROACH

The results of this experiment were not always as encouraging as had been hoped for, particularly in the sense of the failures mentioned above. Some of the reasons for the failures were :

1. The students had difficulty in relating to the two-parameter controller structure. In particular it was difficult for them to understand the independence of the setpoint tracking and the disturbance rejection or stability properties of this form. The expert system could have spent more effort in explaining this, or could have allowed the classical one-parameter structure to be used.

2. The students were not always able to interpret the closed loop responses in terms of physical properties of the system. While they were generally comfortable using the Nyquist plot, they also had difficulty in relating this to the various closed loop responses. This often resulted in the students placing unreasonable or meaningless constraints on certain responses. While it is possible to have the expert system tell the designer that a particular specification is questionable, the system has been designed assuming that the designer has a valid reason for each specification. It is the designer who needs to make the engineering tradeoffs; the best that the expert system can do is to provide advice on what the tradeoffs are, and what the consequences of a certain decision might be.

3. The students had almost no conceptual understanding of the optimization of responses, and certainly no ideas on trading one response against another using weighted optimization. The expert system should have provided a "beginners" guide to optimization, asking the student which features they considered important, and then selecting appropriate optimization weights.

4. Many students were overwhelmed by the number of degrees of freedom associated with this design approach. For example, disturbances may be dealt with at the input or output (or both) of the plant; restricting this to one would have saved a lot of time in many cases. Similarly the ability the deal with both frequency and time domain

responses confused students who did not understand the relationships between them. In essence the problem for many students is that the ability to produce a reasonable design by trial and error decreases as the number of degrees of freedom increases.

5. Few students seemed to make proper use of the help and explanation facilities offered. This can probably be attributed to their lack of experience in using similar software.

There were also many encouraging aspects to the experiment. Apart from its use for designing a controller for a specific task, the expert systems approach was also found to be useful for introducing new concepts to students. Some of these were :

1. Most of the students gained a first hand appreciation of the tradeoffs involved in control system design. This generally occurred as a result of the poor performance of early designs. A powerful feature of the design method is that it readily allows one to squeeze a response and view the effects of this on the others. The expert system is also very useful in identifying explicitly which of the constraints conflict.

2. In spite of no tuition on the subject, many students learned something of the concept of optimizing responses. The setpoint tracking response is one that vividly illustrates the value of this technique; figure 4 shows an optimized command tracking response subject to constraints on the control signal.

3. Some students gained an understanding of the advantages of the two-parameter controller structure. This, combined with use of the optimization feature, were the main reasons for the relatively good tracking performance of the CXS controllers.

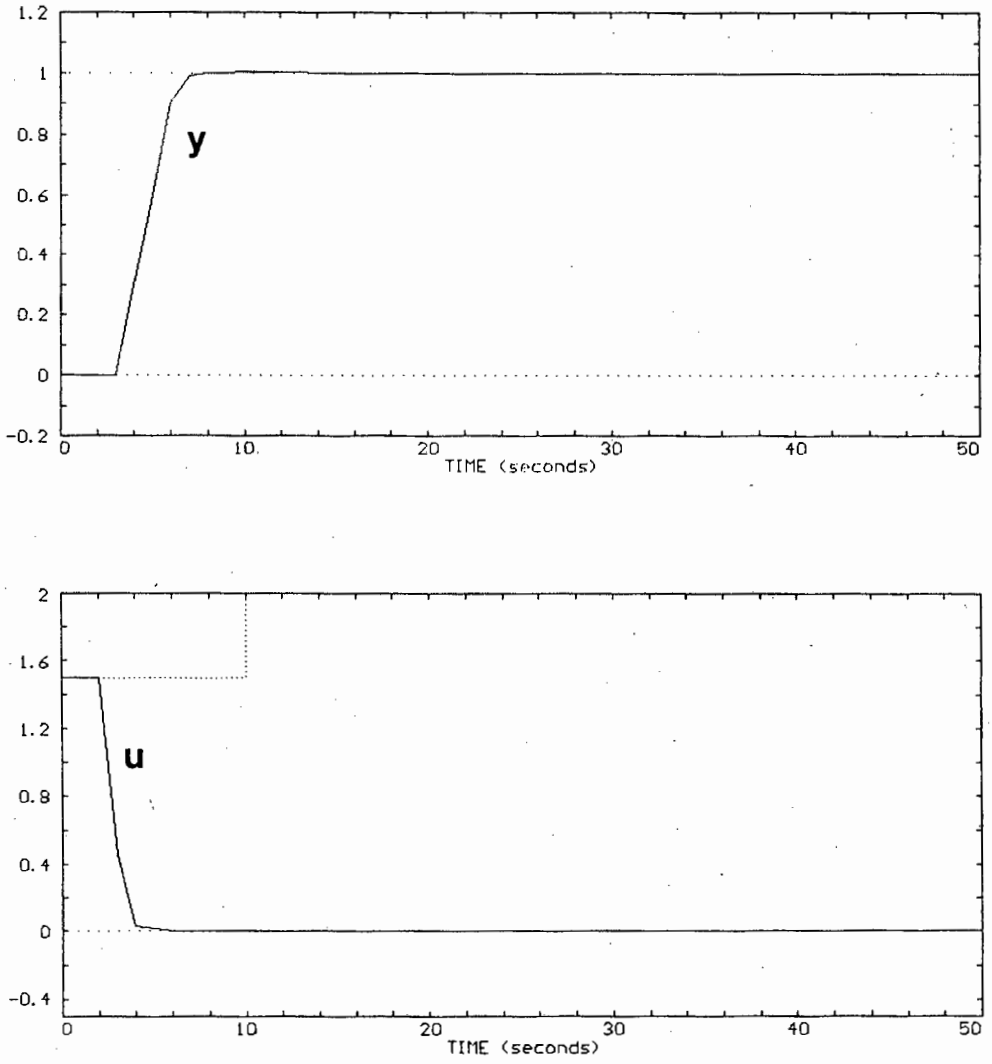


Figure 4. An optimized step response with constrained control signal.

CONCLUSION

An expert system has been used to introduce a new approach to the design of control systems to students, with little additional tuition. In addition to assisting them with their specific design problem, it has been useful in introducing the concepts of tradeoff and optimization in the design process.

There is no such thing as an instant human expert; in practice every expert goes through a learning phase.

Similarly it can be expected that an expert system will go through a similar development cycle. The study has indicated that the expert system still has room to grow. One of the attractive features of the expert system methodology is that it permits extensions to its knowledge base with relative ease.

ACKNOWLEDGEMENTS

Financial support for this research from AECI Ltd. and the Foundation for Research Development is gratefully acknowledged.

REFERENCES

[1] J. R. James, D. K. Frederick, J. H. Taylor, "Use of expert-system programming techniques for the design of lead-lag compensators", *IEE Proceedings*, 1987, vol. 134, Pt. D, no. 3, pp. 137-144.

[2] J. D. Birdwell, "An expert system can aid in the evolution of a design methodology", *Proceedings of the American Control Conference*, 1987, pp. 541-546.

[3] J. D. Birdwell, G. Chang, P. Hanson, L. Hong, J.-S. Lai, G. V. Murphy, "Teaching with the CASCADE Computer-Aided Control System Design Environment", *19th Southeastern Symposium on Systems Theory, Clemson, SC*, 1987.

[4] G. K. H. Pang, A. G. J. MacFarlane, *An Expert Systems Approach to Computer-Aided Design of Multivariable Systems*, Springer-Verlag, 1987.

[5] J. H. Taylor, D. K. Frederick, "An Expert System Architecture for Computer-Aided Control Engineering", *Proceedings of the IEEE*, 1984, vol. 72, no. 12, pp. 1795-1805.

[6] S. P. Boyd, V. Balakrishnan, C. H. Barrat, N. M. Khraishi, X. Li, D. G. Meyer, S. A. Norman, "A New CAD Method and Associated Architectures for Linear Controllers," *IEEE Transactions on Automatic Control*, 1988, vol. AC-33, no. 3, pp. 268-283.

APPENDIX A - CXS design project handout.

Designing control systems using CXS

CXS is an expert system for designing control systems. It allows the user to concentrate on formulating the design specifications, and handles the details of finding a controller which meets them.

Specifications may be posed in terms of constraints on time and frequency domain responses. Specifying a maximum overshoot of 5% is an example of a time domain constraint; in the frequency domain the designer may wish to limit the repsonse to disturbance signals over a certain range of frequencies.

Time and frequency responses may also be optimized, subject to any constraints set. In the frequency domain, an attempt is made to minimize the integral of the gain of the frequency response, while in the time domain it is the integral of the square of the step response error which is minimized. Note that for the time domain responses RY and NY, an offset of 1 is subtracted to form the error value. Each signal is given a weight for optimization - zero means no optimization; this allows you to trade off one response against the other.

One feature of this design method not immediately obvious is that the design method consists of two independent sections; one deals with the transfer functions from R to Y or U, and the other section with the remaining transfer functions. In

physical terms it amounts to designing the precompensator and feedback controller components separately. This means that constraints and optimization weights for these two groups may be set, and solved, independently.

To start the session, boot up from the boot disk. Then insert the work disk. DESIGN.BAT is a batch file to run the CXS design package. Once you have used it to design [and save!] a controller, use the SIM.BAT batch file to run the simulator to evaluate your controller.

When using CXS, the system will ask many questions. In response you may type one of the following :

<RETURN>	This is usually an illegal response, and so it is a good way to get CXS to tell you what the legal answers are.
/explain	This will usually explain the question for you. Type /e for short if you like.
?	If you really don't know the answer to a question, type this and CXS will take things from there. But there are occasions when you will be expected to know the answer, and CXS will keep pestering you until it gets one. In general, the more questions you answer correctly the better the design will be. Use the /explain facility to help you.

For non-numeric answers to CXS questions, the following are considered equivalent.

YES = Y
NO = N
TIME = T
FREQUENCY = FREQ = F

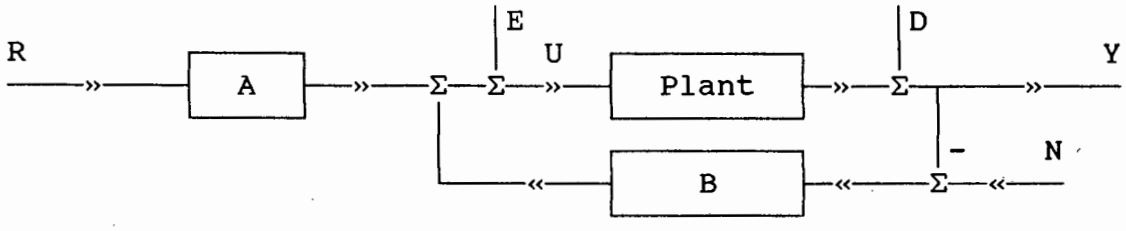
Also note that no distinction is made between upper and lower case letters for any user input.

No doubt you will uncover some bugs in the system. Please describe these in your report in detail. This should include a description of what went wrong, and a list of all the

specifications set at the time. If you find inaccurate or misleading messages, please include a screen dump of these, pointing out the problem.

A sample session with CXS is given below. The user responses are given in **bold** type, and some additional comments have been added in *italics*. Throughout the design session, transfer functions are identified by a two letter combination (source/destination). Thus DY refers to the transfer function from input D to output Y.

Welcome to the expert system based design tool for control system design for Linear, z domain, SISO plants. Please note that all transfer functions are specified as polynomials in z (and not 1/z). The form of the controller is a pre-compensator A and a feedback element B as shown below. The polynomials A and B have common denominators, making for simplified implementation.



NB : The following units are assumed throughout

Frequency domain specs : Hz and dB (20log(gain magnitude))

Time domain specs : Seconds and units (step input of 1 unit used)

What is the sampling time ? 1
The plant is acceptable.
Loading databases.
The Nyquist frequency is 0.5 Hz.
You must now enter your design, using the SET commands. When it has been entered, the SOLVE command will make the system search for a controller that satisfies the specifications, if one exists. For more details on these and the other commands, type /EXPLAIN or HELP. The prompt displays which response will be affected by the command, eg 'NY,TIME>' indicates that the command relates to the time domain of the signal from input N to output Y.

NY,FREQUENCY>**/EXPLAIN**
The commands are [] = optional
[SELECT] g Select a transfer function, where g = RY/RU/DY/DU/NY/NU/EY
d [DOMAIN] select the domain, d = TIME/FREQ

```
 SET LIMIT              Set a constraint on the current
                          transfer fn, or use
 SET lim n1 [FROM n2] [TO n3]   where lim = MIN/MAX
 SET OPT [none]         Set the optimization value
 SET ASYM n             Set the asymptotic value
 SHOW [ALL]             Show [all] the constraints set
 SHOW OPT               Show the optimization values set
 SHOW CONTROLLER        Show the controller transfer function
 SUGGEST                Suggest ways to improve the design
 SOLVE                  Try to find a controller which meets
                          the constraints
 PLOT [ALL]             Plot [all] the transfer function[s].
 PLOT [n1 TO n2]        Plot the transfer function, using
                          range n1 to n2
 PLOT NYQUIST           Do a Nyquist plot
 SAVE                   Save the controller to disk
 QUIT                   Finished / given up!
 HELP [command]         Display more detailed help on these
                          commands
```

*[An M-circle condition is specified as a constraint on the frequency response of the NY signal, where the constraint is computed as 20*log(M). The following command will enter this condition over the entire frequency range.]*

```
NY,FREQUENCY>SET MAX 5
```

[The next concern of our design is rejection of disturbances. This can be specified by constraints on the frequency response of the DY signal. The next set of commands specifies at least 6 dB rejection for frequencies up to 0.016Hz (0.1rad/s).]

```
NY,FREQUENCY>DY
DY,FREQUENCY>PLOT
DY,FREQUENCY>SET MAX -6 TO .016
DY,FREQUENCY>PLOT
DY,FREQUENCY>SHOW ALL
Graph NY [FREQUENCY] : MAX value is 5 from 0 to 0.5
Graph DY [FREQUENCY] : MAX value is -6 from 0 to 0.016
```

[After checking which conditions have been set so far, try to solve for a controller which satisfies them.]

```
DY,FREQUENCY>SOLVE
Is asymptotic tracking of the reference signal required ?
YES
The command 'SET ASYM 1' for the RY signal has been entered
for you. This forces the DC gain from R to Y to be unity,
and thus achieve asymptotic tracking.
Is rejection of low freq disturbances at the plant input
important ? YES
Do you require asymptotic rejection of disturbances at the
plant input ? YES
The command 'SET ASYM 0' for the EY signal has been entered
for you. This forces the DC gain from E to Y to be zero, and
thus achieve asymptotic disturbance rejection. In effect
this means that the controller will contain an integrator.
```

No specifications for the RY transfer function have been given, and thus there is no tracking specification. I would suggest that you specify optimization for the RY step response (time domain), or at least set minimum rise time and maximum overshoot constraints.

[*The expert system has pointed out that no specifications on the transfer function from R to Y have been given. The commands to specify a rise time (to 90%) of 15 seconds, with at most 10% overshoot, are entered below.*]

RY,TIME>**PLOT**
RY,TIME>**SET MAX 1.1**
RY,TIME>**SET MIN 0.9 FROM 15**
RY,TIME>**PLOT**
RY,TIME>**SOLVE**

[*The solve step can take quite some time on slower computers, so be patient! In this instance the solve was successful.*]

Do you wish to see the relevant graphs ? **YES**

[*After viewing the graphs, the expert system is asked to suggest what to do next.*]

RY,TIME>**SUGGEST**
A low frequency limit should be set on the EY signal to ensure disturbance rejection. A starting specification could be 'SET MAX -20 TO 0.0025'.

[*From the graphs shown we were quite happy with the low frequency response of the EY signal, and so this advice is ignored. Ask for another suggestion.*]

EY,FREQUENCY>**SUGGEST**
Do you require the control signal to be limited ? **/EXPLAIN**
In practice one cannot expect the actuator to produce an arbitrarily large input signal for the plant; usually the range of input signals has fixed limits. A time domain limit may be set on the RU and NU signals, or the frequency responses may be limited.
Do you require the control signal to be limited ? **YES**
Neither time nor frequency domain limits have been set for the NU signal. You should check the graphs for these using the PLOT command, and if necessary set constraints on at least one of them. Generally a high frequency limit, or bounds on the initial part of the time response, will be required, as real control actuators (eq valves, motors etc) have a limited output range.
NU,TIME>**PLOT**
NU,TIME>**PLOT -2 TO 3**
NU,TIME>**SET MAX 3 TO 10**

[*Now the control signal has been limited to 3 units over the period 0 to 10 seconds. See if there is a controller which can achieve this.*]

NU,TIME>**SOLVE**
The FREQUENCY domain constraints on DY could not be satisfied, as they conflict with the following specs :
The TIME domain constraints on NU.
Do you wish to see the relevant graphs ? **YES**

[*From the graphs, it seems that the best compromise to resolve the conflict would be to relax the specification on DY slightly. First check which constraints have already been set.*]

DY,FREQUENCY>**SHOW**
Graph DY [FREQUENCY] : MAX value is NONE from 0.016 to 0.5
Graph DY [FREQUENCY] : MAX value is -6 from 0 to 0.016
DY,FREQUENCY>**SET MAX -3 FROM 0.01 TO 0.016**
DY,FREQUENCY>**PLOT**
DY,FREQUENCY>**SOLVE**
Do you wish to see the relevant graphs ? **YES**

[*This compromise was successful. Continue to refine the design.*]

DY,FREQUENCY>**SUGGEST**
No optimization has been specified for any of the signals NY, DY, EY, or NU. You may wish to optimize the response of the NY signal, or some of the disturbance responses. Select the desired signal and domain, and then use the command 'SET OPT' to specify the optimization. If you type 'HELP OPT' you will be given more information on the optimization commands.
DY,FREQUENCY>**NY**
NY,FREQUENCY>**TIME**
NY,TIME>**PLOT**
NY,TIME>**SET OPT**
What type of optimization do you require ? **/EXPLAIN**
There are 4 type of optimization available for time domain signals
   0 : No optimization
   1 : Integral of error^2
   2 : Integral of t.error^2
   3 : Integral of t^2.error^2
The default is type 1
What type of optimization do you require ? **1**
What is the optimization weight ? **1**
NY,TIME>**SUGGEST**
Neither time nor frequency domain limits have been set for the RU signal. You should check the graphs for these using the PLOT command, and if necessary set constraints on at least one of them. Generally a high frequency limit, or bounds on the initial part of the time response, will be required, as real control actuators (eq valves, motors etc) have a limited output range.
RU,TIME>**PLOT**
RU,TIME>**PLOT -1 TO 3**
RU,TIME>**SET MAX 2 TO 10**
RU,TIME>**SUGGEST**
No optimization has been specified for any of the signals RY or RU. You may wish to optimize the step response of the RY signal, or trade it off against that of the RU signal. Select the desired signal and domain, and then use the

command 'SET OPT' to specify the optimization. If you type 'HELP OPT' you will be given more information on the optimization commands.

RU,TIME>**RY**
RY,TIME>**SET OPT**
What type of optimization do you require ? **1**
What is the optimization weight ? **1**
RY,TIME>**SOLVE**
Do you wish to see the relevant graphs ? **YES**

[*At this stage we are happy with the design. The next two steps are important: always save the controller you have designed, and be sure to print the report for your final controller, as this will be used in the evaluation of your design.*]

NY,FREQUENCY>**SAVE**
NY,FREQUENCY>**QUIT**
Do you want the report printed out ? **YES**

Objectives for the design session, and topics for your report.

Your final must include the printed report from your final controller design session, and the graph of the closed loop step response from the simulator. In addition, the following topics must be addressed :

1. Use CXS to design a controller for the plant. Remember to print the report for your final controller. How easy was it to use this design method compared to using classical methods? Did you find it any quicker? How did the performance of your controller compare to the "classical" one (in terms of rejection of disturbances and response to a step input)? Which specifications did you find to have the most effect on the performance of the design?

2. This design consists of precompensator and feedback control elements. Discuss the advantages/disadvantages of this approach against designs using only the feedback element (compare step responses of RY and NY). Why are they different?

3. Investigate the trade-off between the controller gain (limits on the time and/or frequency responses of the NU and RU signals), and the disturbance rejection and properties. Is it necessary to limit the controller gain? Why?

4. Investigate the effects of constraints on the frequency response of the NY signal. Are these limits necessary for a good design? Why? What else is affected by them?

5. Discuss how (if at all) CXS has helped you for items 1 to 4. Are there any design aspects which you could not have catered for without it? Have you learned anything about control theory from using it? What features are missing from this design approach (things you wanted to do but could not / did not know how to)?

APPENDIX B - Table of controller performance parameters

	Classical controller				CXS controller			
Student	RY	DY3db	DYmax	NUmax	RY	DY3db	DYmax	NUmax
1	73	0.014	2.966	0.475	8	0.014	3.304	1.069
2	147	0.006	1.594	0.222	8	0.022	3.941	3.000
3	30	0.014	2.867	0.410	8	0.017	6.781	1.500
4	154	0.008	1.457	0.408	7	0.017	3.139	2.000
5	41	0.022	7.975	0.768	51	0.011	2.063	0.630
6	68	0.017	2.002	0.525	27	0.028	4.374	1.897
7	102	0.011	2.066	0.735	8	0.022	7.636	8.211
8	93	0.011	1.917	0.565	20	0.022	6.132	3.045
9	18	0.014	1.780	0.385	6	0.028	4.146	1.500
10	81	0.011	2.970	0.465	24	0.014	2.816	3.000
11	50	0.017	3.152	0.780	16	0.008	2.195	0.700
12	66	0.014	3.742	0.684	9	0.011	2.327	1.147
13	20	0.017	2.563	0.808	6	0.028	2.999	2.143
14	66	0.014	2.599	0.636	7	0.022	5.831	2.336
15	42	0.011	2.323	0.512	27	0.014	3.467	1.500
16	28	0.014	3.944	0.912	41	0.017	5.210	3.478
17	72	0.014	3.756	0.925	51	0.017	4.264	2.670
18	14	0.017	2.280	1.008	46	0.011	2.476	1.500
19	62	0.014	4.199	1.232	14	0.017	4.873	3.000
20	67	0.014	2.474	0.746	8	0.014	3.104	2.234
21	81	0.011	1.682	0.888	11	0.022	3.667	4.443
22	134	0.008	1.456	0.306	6	0.017	2.360	1.210
23	61	0.014	1.907	0.766	20	0.011	1.732	1.000
24	22	0.017	2.765	0.637	6	0.017	3.145	1.100
25	69	0.014	2.514	0.772	10	0.017	7.797	3.970
26	88	0.011	1.609	0.424	12	0.017	2.235	1.372
27	26	0.014	2.636	0.404	7	0.011	2.508	0.838
28	51	0.017	2.970	0.628	7	0.017	2.028	2.221
29	78	0.014	2.391	0.830	36	0.017	3.392	2.000
30	63	0.014	3.490	0.538	7	0.017	3.068	1.000
31	51	0.022	2.320	1.590	6	0.028	3.948	4.160
32	51	0.014	2.431	0.512	7	0.022	4.530	1.800
33	78	0.014	1.772	0.416	5	0.017	2.099	1.190
34	67	0.014	2.676	0.530	6	0.017	3.211	1.000
35	59	0.017	2.519	0.566	27	0.028	4.667	2.000
36	24	0.022	2.854	1.020	27	0.022	3.866	2.000
37	50	0.017	3.796	0.940	7	0.022	5.402	4.357

Legend

RY : The time to settle to within 5% (in seconds) following a step change at input R.

DY3db : The frequency (in Hz) below which disturbances at input D are attenuated by least 3 dB's.

DYmax : The maximum amplification (in dB's) of disturbances at input D.

NUmax : The maximum change in the control signal following a unit step at input N or D.

September 1990 THE TRANSACTIONS OF THE SA INSTITUTE OF ELECTRICAL ENGINEERS 15

An Expert System for Controller Design

C.D. Tebbutt

ABSTRACT

An intelligent tool for the design of linear control systems has been built by combining an expert system with a recently developed numerical design technique. The expert system assists the designer in formulating a comprehensive and achievable specification, and in dealing with any constraint conflicts which may arise. The use of the design system by undergraduate students is also examined.

KEYWORDS

Expert Systems, Control System Design, Computer-Aided Design.

INTRODUCTION

The design of control systems is a complex task requiring a combination of knowledge, engineering judgement, and designer creativity. The designer needs to select an appropriate design technique for the particular problem, and have a good understanding of the mechanics, strengths and weaknesses of this technique. In addition it is necessary to be able to recognize and evaluate the engineering tradeoffs. Traditional computer aided control system design (CACSD) tools have not addressed these issues directly.

The 1980's have seen many applications of expert system techniques to the problem of designing control systems [1-5]. The expert systems of James [1], and Taylor and Frederick [5], allow little user interaction with the design process. In contrast, Pang and MacFarlane [4] stress that design should be an interactive process, with the engineering judgement left to the designer. This second philosophy was adopted; the expert system implemented goes further than the user interface described by Birdwell [2,3], helping the user to formulate a comprehensive design specification, and deal with any conflicts and tradeoffs in that specification.

This paper describes the construction of the intelligent design system, and discusses its use by a group of fourth year students during their control system design project.

Submitted : January 1990
Accepted : August 1990

CONSTRUCTION OF THE DESIGN SYSTEM

The overall structure of the design system is shown in figure 1. An important feature of the structure is the division of tasks between the expert system and the modules of the external package. These modules, written in a conventional programming language (C), are used as a collection of subroutines which the expert system may call upon. Broadly speaking, the design specification is formulated within the expert system, and then passed down to the numerical modules for synthesis of the controller. The external modules also provide analysis and graphics capabilities.

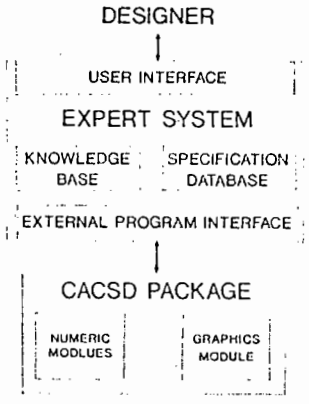


Figure 1. Structure of the Design System.

The design method.

The underlying design technique [6] used in this application transforms the control system design problem into a linearly constrained quadratic programming problem, which is then solved using a standard algorithm. The important steps in the design method are shown in figure 2, and are straightforward for single-input, single-output designs. Consider the two-paramater controller structure shown in figure 3, and let $S(z)$ be the set of proper stable z-domain transfer functions. Factor the plant transfer function $G(z)$ as

$$G(z) = N(z) / D(z) \qquad (1)$$

where $N(z)$, $D(z) \in S(z)$. Next, find $X(z)$, $Y(z) \in S(z)$, which satisfy

$$X(z)N(z) + Y(z)D(z) = 1. \qquad (2)$$

This This step is equivalent to finding a controller which stabilizes the plant. Factorization theory [7] then gives formulae

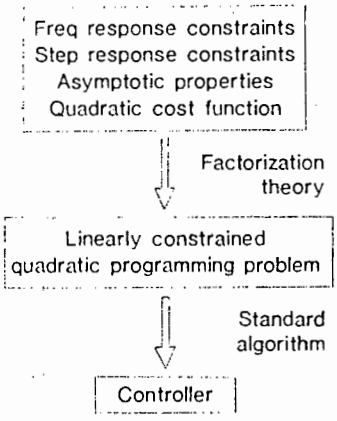


Figure 2. The Design Algorithm.

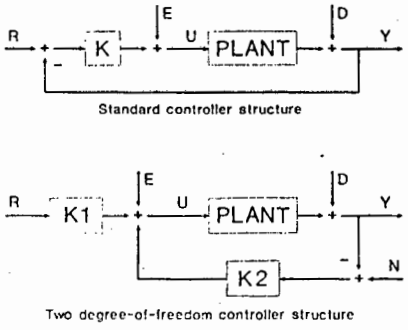


Figure 3. Alternative Controller Configurations.

for *all* stable closed loop transfer functions in terms of arbitrary transfer functions Q1(z), Q2(z) ϵ S(z); it turns out that the closed loop responses from input R depend only on Q1(z), and all others only on Q2(z). For example, the closed loop transfer function from input R to output Y is given by

$$H_{RY}(z) = N(z)Q1(z), \qquad (3)$$

and that from input D to output U by

$$H_{DU}(z) = -D(z)X(z) - D(z)Q2(z)D(z), \qquad (4)$$

Thus the design is reduced to finding transfer functions Q1(z) and Q2(z) which give the desired closed loop responses.

Design specifications can be set directly as constraints on the closed loop step or frequency responses, and (optionally) a quadratic cost function to be minimized. For example, a design may require a rise time of at most 20 seconds with no more than 5% overshoot, and at least 20dB rejection of disturbances at the plant output for frequencies up to 0.005 Hz. Asymptotic specifications are treated as equality constraints on the 0 Hz frequency response. Some typical performance constraints are illustrated in figure 4.

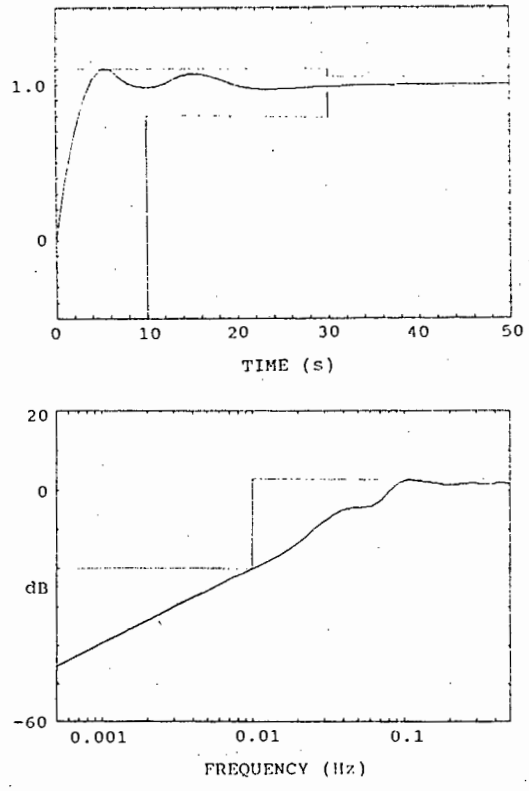


Figure 4. Typical Time and Frequency Domain Constraints.

Boyd [6] uses the set of finite impulse response (FIR) filters to approximate S(z), and gives details on how the design specifications may be translated into linear constraints on a parameter vector made up of the coefficients of the FIR filters. The resulting linearly constrained quadratic programming problem is then solved using a standard algorithm, giving Q1(z) and Q2(z). Finally, the controller transfer functions are obtained from the formulae

$$K1(z) = \frac{Q1(z)}{Y(z) - Q2(z)N(z)} \qquad (5)$$

and

$$K2(z) = \frac{X(z) + Q2(z)D(z)}{Y(z) - Q2(z)N(z)} \qquad (6)$$

Where no solution to the design problem exists, the quadratic programming algorithm terminates within a finite number of steps; here the algorithm provides additional data which the expert system uses to identify the conflicting specifications.

The expert system.

The expert system runs under CXS, a backward-chaining rule-based expert system shell

developed for the IBM/PC by the author [8]. At present there are 210 rules and 34 questions in the knowledge base; much of the 63k bytes of this program is text displayed to the user.

Knowledge is stored in rules and facts; in addition the Prolog-style database facilities in CXS are extremely useful for storing and analyzing the design specification. The records in the database of performance constraints have the following fields :

SIGNAL	the transfer function, eg H_{RY}
DOMAIN	the time or frequency domain
VALUE	the constraint value
TYPE	the type of constriant, eg MIN or MAX
LOW	the lower end of the domain range
HIGH	the upper end of the domain range

Thus an engineering constraint such as a maximum overshoot of 5% during the time interval from 0 to 10 seconds, is represented by the record ["RY", "time", 1.05, "max", 0.0, 10.0]. The rule shown below is typical of those used to analyze the specification database; this one fires when no low frequency disturbance rejection constraint has been specified.

```
IF suggstn3 is unknown
AND FIND specdb["dy","freq",=TO,"max",0,*]
AND TO is "none"
OR  TO > 0
AND outp_dist_rej_reqd is "yes"
THEN
! suggstn3 is "done"            ;don't repeat suggestion 3
  suggestion is "done"
  T1 = Nyq_freq/200
  DISPLAY
A low frequency limit should be set on the DY signal to
ensure disturbance rejection. A starting specification could
be 'SET MAX -20 TO \%T1%'.
END
```

An efficient interface to external software is essential for a high-performance design system. CXS allows linking to external memory resident programs through a message passing scheme based on DOS interrupts. The external programs have access to an array of expert system variables, which may contain symbolic or numeric values. This interface is used by the expert system to execute functions such as loading transfer functions, plotting step or frequency responses, translating the specification into a quadratic programming problem, or examining the status of a performance constraint. A typical rule using this interface is

```
IF LINE cmdline["plot"]    ;If the command line is PLOT
AND cmdline = 1            ;and only 1 symbol on the line
THEN
  T0 = current.signal      ;assign current response and
  T1 = current.domain      ;  domain to array variables
  INTR 96,5,1              ;call interrupt 96 with function
                           ;  code 5 and subfunction 1
  command is "plot"
END
```

TASKS PERFORMED BY THE EXPERT SYSTEM

Although the numerical design technique described above is extremely powerful, it forms only one part of the total design process. There are many other issues to be considered, and the expert system proves very useful in addressing these. Apart from coordinating the functions of the numeric and graphic modules, the most important tasks performed by the expert system are :

1. Providing a versatile and yet consistent user interface. Incorporated into this interface is an explanation facility which is very useful in guiding inexperienced users.

2. Checking that the plant model is usable. In particular the plant transfer function must be causal and coprime (controllable and observable); the system will optionally cancel any stable pole-zero pairs in the plant transfer function.

3. Checking for and explaining any conflicts in the engineering specifications. This includes preliminary and largely superficial checks on the specifications for conflicts such as a specified rise time less than the plant dead time, or conflicts between the asymptotic properties and other low frequency specifications.

   The explanation of conflicts identified by the quadratic programming algorithm is more important. Here the user is shown the source of the conflict, and given suggestions for dealing with it. To provide this information, the linear constraints are presented to the quadratic programming algorithm in groups, where each group relates to a particular engineering specification. When there is no solution, the status of each of these groups (satisfied, active or unsatisfied) is analysed to determine the cause of the conflict. This analysis is performed by rules with the generic form

```
IF group n is not satisfied
AND groups 1 ... n-1 are active/satisfied
THEN
  describe the problem
  suggest a remedy
END
```

4. Providing advice for the designer. Representing the design in terms of constraints also proves useful in that it allows the expert system to analyse the design specifications readily. It can determine which design features have not yet been considered, and remind the user of their importance where necessary. The expert system contains a list of features which an expert designer would consider; these are represented as a collection of rules with the generic form given below, and are used to determine what the best advice at the time is.

```
IF this advice has not been given yet
AND this design feature is not yet
    accounted for in the present design
    specifications
AND this design feature is probably
    required
THEN
  display the advice
END
```

USE OF THE EXPERT SYSTEM

The fourth year control systems was the design of a level controller for a simulator of the sump of a milling process. This process was modelled by a transfer function of the form

$$G(s) = \frac{A}{s} e^{-Bs}$$

The students were each given values for the parameters A and B, typically 0.2 and 4.0 respectively. Disturbances were added at the input of the plant, and comprised a dc offset, a relatively large disturbance at 0.016Hz plus a number of harmonics, and high frequency noise.

The students performed two designs for this problem. The first was done using the classical methods taught class; mostly an open loop Nyquist approach was used, resulting in a PI controller. For the second design the expert system was used, with minimal additional tuition. The instructions given to the students for the expert system design (referred to as CXS), and the controller performance data for the two designs, are reproduced elsewhere [9].

The performance of a controller for this problem can be judged in terms of its ability to track the setpoint and reject disturbances. Additional considerations are the robustness of the controller, and the demands it places on the actuator. The engineering tradeoff is between a slow, robust, low performance controller, and a sensitive high performance controller.

To compare the response to a step setpoint change, the time to settle to within 5% was taken as the measure of performance. In 89% of the designs the CXS controllers exhibited better responses; the average settling time was 16.2 seconds compared to 63.4 seconds for the classical controller. This was to be expected from the advantage of using a two-parameter control structure over the standard configuration (see figure 3), which makes the comparison unfair. However it should be remembered that the students would not have been able to design a suitable pre-filter without the expert system.

More important are the disturbance rejection properties. The parameter chosen for the comparison was the frequency below which disturbances at the plant output were rejected by at least 3 dB. The CXS controllers performed better in 70% of the designs in this regard; the average frequency was 0.0146 Hz for the classical design, against 0.0187 Hz for the CXS controller, an improvement of 28%.

Unfortunately almost half of the CXS designs suffered from excessive controller gain or amplification of high frequency disturbances. The failures were due in part to the expert system not adequately warning of the design deficiencies, and partly to the students ignoring the advice that was given. In many cases there was a deliberate decision to tradeoff almost everything for low frequency disturbance rejection.

The results of this experiment were not always as encouraging as had been hoped for, particularly in the sense of the failures mentioned above. Nevertheless there were many positive aspects. Some of the important points highlighted were :

1. Many students had difficulty with the two-parameter controller structure. In particular the independence of the setpoint tracking and the disturbance rejection or stability properties of this form was seldom grasped. The expert system could have spent more effort in explaining this, or could have allowed the standard structure as an option. Nevertheless the two-parameter structure, together with the optimization facilities, account for the relatively good tracking performance of the CXS controllers.

2. While the students were generally comfortable using the Nyquist plot, they were not always able to interpret the closed loop responses. This sometimes resulted unreasonable or meaningless performance constraints. While it is possible for the expert system to indicate that a particular specification is questionable, the system has been designed assuming that the designer has a purpose for each specification. It is the designer who needs to make the engineering tradeoffs; the best that the expert system can do is provide advice on what the tradeoffs are, and what the consequences of a certain decision might be.

3. Most of the students gained an appreciation of the tradeoffs involved in control system design, often as a result of the poor performance of early designs. It should be remembered that the students had received almost no tuition on the concept of cost of control. A powerful feature of this design method is that it readily allows one to squeeze the specifications on one response and view the effects on the others, identifying explicitly any constraint conflicts.

4. Initially the students had almost no conceptual understanding of the optimization of responses. The expert system should have provided a "beginners" guide to optimization, asking the student which features they considered important, and then selecting appropriate optimization weights. The students were greatly impressed by optimization feature, as they could not do anything similar using classical techniques.

CONCLUSION

An expert system has been used to build an intelligent CACSD tool, and this in turn to introduce a new approach to the design of control systems to students. In addition to assisting them with their specific design problem, it has been useful in introducing the concepts of tradeoff and optimization in the design process.

There is no such thing as an instant human expert; in practice every expert goes through a learning phase. It should be expected that an expert system will also mature through a similar development cycle. One of the attractive features of the expert system methodology is that it permits extensions to its knowledge base with relative ease.

ACKNOWLEDGEMENTS

Financial support for this research from AECI Ltd. and the Foundation for Research Development is gratefully acknowledged.

REFERENCES

[1] James, J R, D K Frederick, J H Taylor, "Use of expert-system programming techniques for the design of lead-lag compensators", *IEE Proceedings*, 1987, vol. 134, Pt. D, no. 3, pp. 137-144.

[2] Birdwell, J D, "An expert system can aid in the evolution of a design methodology", *Proceedings of the American Control Conference*, 1987, pp. 541-546.

[3] Birdwell, J D, C Chang, P Hanson, L Hong, J-S Lai, C V Murphy, "Teaching with the CASCADE Computer-Aided Control System Design Environment", *19th Southeastern Symposium on Systems Theory, Clemson, SC*, 1987.

[4] Pang, G K H, A G J MacFarlane, *An Expert Systems Approach to Computer-Aided Design of Multivariable Systems*, Berlin: Springer-Verlag, 1987.

[5] J H Taylor, D K Frederick, "An Expert System Architecture for Computer-Aided Control Engineering", *Proceedings of the IEEE*, 1984, vol. 72, no. 12, pp. 1795-1805.

[6] S P Boyd, V Balakrishnan, C H Barrat, N M Khraishi, X Li, D G Meyer, S A Norman, "A New CAD Method and Associated Architectures for Linear Controllers," *IEEE Transactions on Automatic Control*, 1988, vol. AC-33, no. 3, pp. 268-283.

[7] Vidyasagar, M, *Control System Synthesis: A Factorization Approach*, M.I.T. Press, 1985.

[8] Tebbutt, C D, *The CXS Expert System Shell*, 1990.

[9] Tebbutt, C D, "Use of an Expert System for Controller Design", presented at the RUGSA Symposium on the Applications of Knowledge-based Systems in Engineering and Control, Johannesburg, October 1989.

AUTHOR'S BIOGRAPHY

C D Tebbutt was born in Port Elizabeth, South Africa. He completed his B.Sc. degree at the University of Cape Town in 1983, and M.S.E. at the University of Michigan in 1985. He is presently studying towards a PhD in Control System Design at the University of Cape Town.

AUTHOR

Colin Tebbutt, Pr. Eng., B.Sc., M.S.E.
Department of Electrical and Electronic Engineering
University of Cape Town
University Private Bag
Rondebosch
7700
South Africa

APPENDIX I. AN EXPERT SYSTEM FOR MULTIVARIABLE CONTROLLER DESIGN.

Abstract

An expert system for the design of linear multivariable control systems has been implemented on a personal computer. It assists the user in formulating a comprehensive and achievable design specification, and in dealing with conflicting design constraints. The expert system also effectively extends the scope of the underlying CACSD package.

1. INTRODUCTION

Over the past decade there has been much interest in the application of artificial intelligence to control system design. While existing computer aided control system design (CACSD) tools are almost exclusively analysis packages (Pang *et al.*, 1990), the use of expert system techniques have offered the hope of producing full-fledged design packages, which are able to provide meaningful guidance for the user during the design process.

Taylor and Frederick (1984) present an overview of the application of expert systems to control engineering, in particular outlining the wide range of design activities which should be addressed. They propose an architecture where the expert system coordinates and integrates many analysis and design procedures. Their design system included an expert system implementation of an algorithm for single variable lead-lag compensator design (James *et al.*, 1986, 1987), which made use of subroutines in the Cambridge Linear

Analysis and Design Program CLADP (Edmunds, 1979).

Trankle, Sheu and Rabin (1986) describe a two-level expert planning system, based upon the CTRL-C package (Little, 1985). A high level planner produces a skeleton plan, and a list of performance specifications for the low level planner to satisfy. The low level planner in turn creates a list of commands for the CTRL-C package. Side effects, resulting from the tradeoffs inherent in controller design, complicate the planning. The user has minimal involvement in the design process with this system.

Nolan (1986) developed an expert system which deduces the feasibility of various single variable feedback configurations, as well as an appropriate synthesis method to be used, based upon the plant type number and order. The system uses algebraic manipulation of the plant transfer function and controller structure, and once a synthesis method is selected, guides the designer in using an appropriate conventional CACSD package.

Birdwell *et al.* (1985) discuss an expert system interface to a multivariable LQG/LTR design package. The expert system, named CASCADE, was found useful in automating some of the design details, thus allowing the user to concentrate on the design process and significantly reducing the time required to complete a trial design (Birdwell, 1987).

The MAID expert system described by Pang and MacFarlane (1987), and by Boyle *et al.* (1989), uses three design techniques to address the design problem. These are their 'simple design technique', a 'reverse frame alignment technique', and an observer-based controller design technique, in order of increasing complexity. Data for design analysis is obtained from the characteristic gains and phases and principal gains of the system. The design is attempted using the simpler methods first, and if not satisfactory, the more complex methods are tried. The

authors emphasize interactive design, and recommend that engineering judgement decisions be left to the designer. This package has recently been extended to include a stable factorization design approach (Pang *et al.*, 1990), similar to that used here.

While each of these design systems attempt to find a controller which satisfies some design specification, with varying degrees of user involvement, the design tool discussed below takes this approach one step further by also assisting the designer in developing the specification. As stressed by MacFarlane *et al.* (1989), control system design is an exploratory and experimental process during which the specification is systematically refined. Therefore it is appropriate that an intelligent design tool should address the evolution of the specification during this process.

To achieve this goal, an intelligent, interactive design tool named MV-CXS has been built by augmenting a powerful CACSD package with an expert system. The expert system assists the user in formulating a comprehensive and achievable design specification, and in dealing with conflicting design constraints; it also effectively extends the scope of the CACSD package. The CACSD package is concerned with finding a controller that meets these specifications.

2. NOTATION

During its interaction with the designer, the CACSD package and expert system refer to the various closed loop responses of Fig. 1 using a two letter notation; for example, DY indicates the closed loop response to stimuli at input D, as observed at output Y. Individual elements in a response are identified using the notation [i,j]; thus RU[2,3] denotes the response at output U2 to stimuli at input R3. In addition, the plant response is denoted by G, those of the controller by K1 and K2, and the open loop response (i.e.

G.K2) by GK. The same notation is used below.

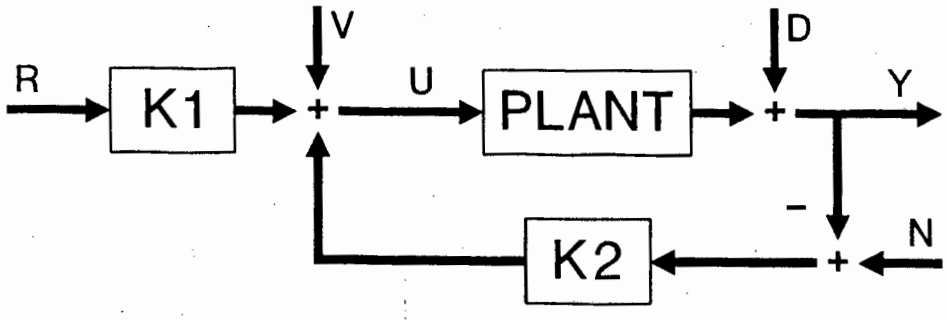


Fig 1. Control system configuration.

3. THE CACSD PACKAGE

Recently a powerful method for designing linear control systems has been introduced (Boyd *et al.*, 1988). Based upon factorization theory, it tackles control system design problems by translating them into approximately equivalent quadratic programming problems which can then be solved using an established algorithm. The important steps in this process are summarized below, and are illustrated in Fig 2.

An important result from factorization theory for linear time-invariant systems is that all internally stable closed loop transfer functions may be parameterized in the form

$$H = T1 + T2.Q.T3 \qquad (3.1)$$

where **T1**, **T2**, **T3** and **Q** are stable transfer function matrices. **T1**, **T2** and **T3** are defined in terms of stable coprime factorizations of the plant and a nominal stabilizing controller, and **Q** is chosen by the designer (or the design algorithm) to give the desired closed loop performance. The design is performed in two independent sections; the closed loop responses from input R depend only on one parameter matrix (named **Q1**), and the remainder only on matrix **Q2**. Vidyasagar (1985) provides a thorough treatment of this theory.

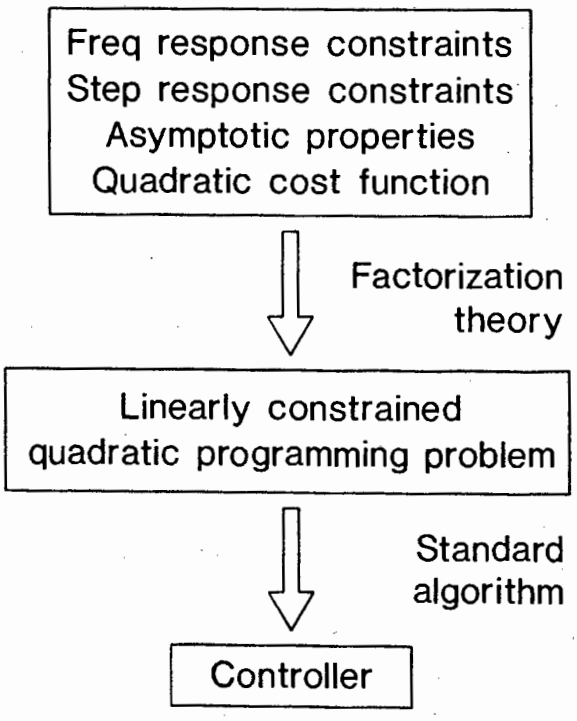


Fig 2. Overview of the design method.

For the sampled data case, Boyd *et al.* (1988) have shown that if **Q** is represented by a finite impulse response (FIR) filter, then constraints on the closed loop step and frequency responses may be translated into linear constraints on a parameter vector, where the elements of this vector are the coefficients of the FIR filter. The linear constraints each have the form

$$\mathbf{a}^T\mathbf{x} + b \geq 0 \qquad (3.2)$$

where **x** is the parameter vector. An efficient representation for these linear constraints has been developed by Tebbutt (1990a). Typical performance constraints are illustrated in Fig 3. Similarly, a quadratic cost function based on the closed loop responses can be translated into a quadratic function of the parameter vector, in the form

$$J(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c \qquad (3.3)$$

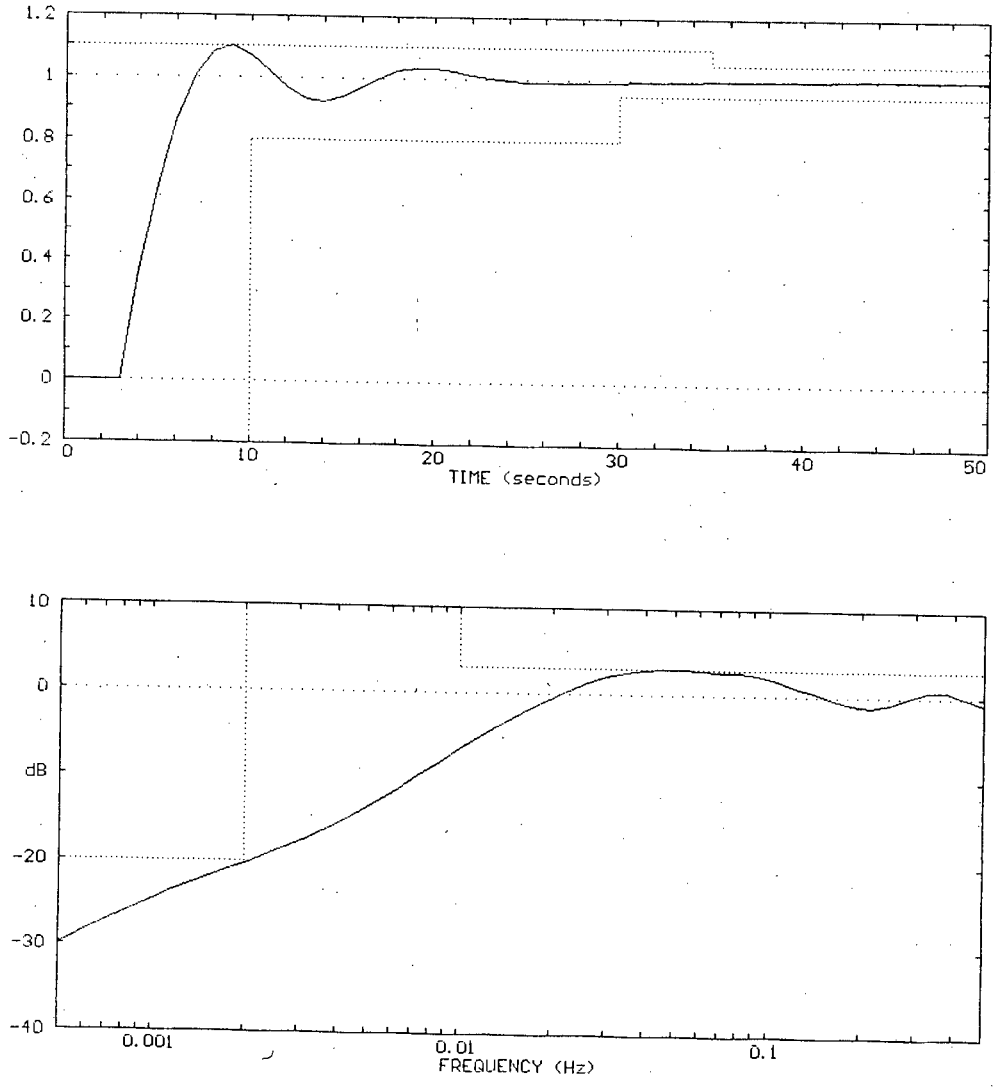


Fig 3. Typical performance constraints.

The resulting quadratic programming problem may then be solved using an active-set algorithm (for example Scales, 1985) to find the optimum **Q**, and the controller computed from the equation

$$\mathbf{K} = (\mathbf{T4} + \mathbf{Q}.\mathbf{T5})^{-1}(\mathbf{T6} + \mathbf{Q}.\mathbf{T7}). \qquad (3.4)$$

T4, **T5**, **T6** and **T7** are stable coprime matrix fractions of the plant and nominal controller transfer functions.

A powerful feature of the quadratic programming algorithm is that it always finds the global optimum solution where one

exists; the absence of any feasible solution is also determined within a finite number of steps. In the latter case, the Lagrange multipliers computed by the algorithm provide useful information; this is discussed in more detail later.

A CACSD package based on this method for multivariable systems has been implemented by Tebbutt (1990b). The package currently caters for linear sampled data plants with up to 5 inputs and 5 outputs; the design is specified in terms of performance constraints on the closed loop time and frequency domain responses, and may also include a quadratic cost function to be minimized. In addition to translating the control system design problem into an equivalent quadratic programming problem and solving this, the CACSD package forms a library of special functions. This enables the expert system to coordinate tasks such as loading and saving transfer function matrices, computing the matrix fractions, analyzing the closed loop performance, and plotting the step and frequency responses.

4. EXPERT SYSTEM TASKS

The CACSD package rEquires that the design be specified in terms of performance constraints on the closed loop response of the system. The design cycle comprises setting performance constraints and a cost function for the control system, and then using the CACSD package to find a controller which satisfies these. This is generally an iterative process whereby the specification is engineered in stages, with specifications being added, tightened, or relaxed at each step, until optimal performance is achieved. The main function of the expert system is to assist the user in formulating and refining this specification; the assistance takes place at many levels, ranging from simply explaining the commands available and how they should be used, to providing advice on how the specification could be improved.

Commands available to the user may be divided into two main groups. The first group comprises those used for editing the specification, solving for a controller, and examining the performance of the controller, for example the EDIT, SOLVE and PLOT commands. The second comprises those used to assist the designer, and includes the HELP, SUGGEST and NEXT STEP commands. This group, and the most important tasks performed by the expert system, are examined in detail below.

4.1 The user interface and philosophy.

The expert system attends to almost all of the interaction with designer. Unfortunately novice and experienced designers have different requirements of a user interface. Experienced designers need ready access to comprehensive and detailed information on the state of the design, and complete freedom to choose from all possible design alternatives; the expert system should also contribute towards improving the productivity of the expert. Novice designers are often overwhelmed by too much information, or too many degrees of freedom in the design method; they need assistance in interpreting the information, and guidance in choosing amongst the design options. Infrequent users of a design system, novice or otherwise, may require help with the design language. Taylor and Frederick (1984) give further examples of areas where 'less-than-expert' users experience difficulty with traditional CACSD software. The expert system user interface aims to deal with these issues.

The interface is largely command-driven for ease of use by proficient users; the expert system gives the experienced designer almost direct access to the underlying CACSD package, with only brief comments on the design progress. To assist novice and infrequent users there are also commands such as HELP, SUGGEST, and NEXT STEP. The concept of an expert system intercepting these types of commands has been termed 'command spy' by Larsson and Persson (1986). This structure allows users to request advice and assistance when desired. As the user's knowledge and experience with the

system grows, they will tend to use the advice facilities less frequently.

Pang and MacFarlane (1987) propose that matters of engineering judgement be left to the designer. This application follows the same philosophy, with the responsibility for all the final design decisions resting on the designer. The expert system has little knowledge of the physical system to be controlled; for example it does not know whether a high frequency gain of +10 dB on the NU response is excessive or not. While it can and does generate warnings and reminders, this type of decision is ultimately left to the designer.

On the whole, the expert system does not question the specifications placed by the designer; it is assumed that the designer has some constructive purpose for each specification. However there are certain situations where the designer is warned of 'suspicious' specifications. Similarly, the expert system does little in checking that the advice given to the user is applied; it is assumed that the designer may have a valid reason for ignoring it.

The expert system has no automatic learning capabilities. Nevertheless, after many design sessions and through the skill of the knowledge engineer, the expert system has undergone a substantial indirect 'learning' experience, resulting in the current version of the design tool being significantly more capable than earlier versions (Tebbutt, 1990c and 1990d). One of the strengths of the expert systems approach is the relative ease with which this learning, or extension of the knowledge base, can be effected (Taylor and Frederick, 1984).

4.2 Explaining the design language and methodology.

The design language, terminology and the overall design philosophy for the CACSD design method may be unfamiliar to novice or infrequent users; MacFarlane *et al.* (1989) point

out that expert systems may contribute towards making design packages more easily accessible in these situations. The expert system contains a large amount of information on the use of the design tool, covering the design language and notation, the syntax and use of the commands, and the general philosophy for using this design method. While much of this is a simple menu-driven help system as shown below, or is programmed into the expert system explanation facility, a small measure of reasoning is used in generating the context-sensitive help. One of the objectives of the interface is to tutor the user in applying the design tool.

```
DY[1,1](frequency) >HELP
Help is available in the following areas :

1.      The design language and philosophy.
2.      A particular command.
3.      The step-by-step design guide.
4.      Refining your design.
5.      Dealing with conflicting constraints.
6.      Checking that the design is complete.

Which of these do you need help with ? 1

Choose one of the following :

1.      The design method overview.
2.      Terminology used in this program.
3.      How to plan your design.
4.      Selecting a response to work on.
5.      Plotting a step or frequency response.
6.      Setting constraints on a response.
7.      Setting up an optimization function.
8.      Finding a suitable controller.
9.      Selecting the design parameters.

....
```

Genesereth (1982) shows how an intelligent system may reconstruct the user's plan of action in order to provide advice on the correct usage of MACSYMA, a symbolic mathematics program. Jackson and Lefrere (1984) develop this concept further. Larsson and Persson (1986) store typical command sequences in scripts, and match these against the history of commands issued by the user in an attempt to understand the user's intentions. Similar types of analysis could also be used to assist the control system designer.

For example, when an invalid command is issued, the expert system could attempt to identify the designer's intentions from the history of design steps, and then explain the correct usage of the commands. At present it is felt that these approaches are not justified, as the commands are seldom chained into distinct sequences but rather used independently. A simple notification of the error, combined with suggestions on how to obtain further help, suffices.

Nevertheless the expert system does monitor how it is being used by the designer. This information is useful, for example, when deciding whether or not certain features of the design system deserve explanation.

4.3 Presentation of the design data.

One of the most important functions of any design system is that of providing the designer with information on the status of the design; this information should be both comprehensive and easily comprehended.

The CACSD package on its own does not provide comprehensive information on the design in the sense that only information relating to the specifications is made available, and the range of specifications permitted is limited. The expert system, often using supporting numerical software, attempts to effectively increase the range of specifications available, and also performs further analyses on the design, supplementing the information available. This aspect of the expert system is dealt with in detail later.

The expert system also attempts to provide the design information in an easily comprehended form. An example of this is the analysis of the Lagrange multipliers from the quadratic programming algorithm which is used to describe conflicts in the design specification.

4.4 Explanation of design specification conflicts.

Conflicts in the specification are detected at two levels by the CACSD package, in both cases as conflicts in the set of linear constraints generated for the particular design. The expert system has the task of explaining these conflicts, and of suggesting ways for dealing with them. The expert system also deals with other complications which preclude a solution to the design; for example there is the possibility of insufficient RAM memory being available for large design problems.

Some conflicts are detected when the specification is translated into linear constraints. These conflicts relate to linear constraints in the form

$$a^T\mathbf{x} \leq \alpha < 0,$$

with $a = 0,$

which cannot be satisfied by any parameter vector **x**. When the CACSD package detects this type of conflict, the expert system is notified of the response concerned. The message displayed for a conflict in a time domain specification is shown below.

> It is not possible to satisfy the specifications on the RY[1,1] time domain response, due to the characteristics of the plant (usually the plant dead time). You will have to relax the specifications on this response.

Most conflicts are detected when the quadratic programming algorithm (QPSOL) finds no feasible solution to the quadratic programming problem. To facilitate analysis of these conflicts, the linear constraints are grouped according to the design specifications that they represent. Thus all constraints relating to specifications on the RY[2,2] step response will be in one group, and all those on the NU[1,2] frequency response will be in another. QPSOL tackles phase I of the quadratic programming problem by

finding a vector **x** which satisfies all constraints in the first group, then one which satisfies the constraints in the first two groups, and so on, until all groups have been considered, or a conflict has been found.

The expert system has access to the status of each of these groups following a call to QPSOL; The individual linear constraints are given a status of active, satisfied, not satisfied, or unknown, according to the terminology of the active set method. The status of a group is determined from the status of the corresponding linear constraints, using the following rules :

1. If the group has not yet been considered by QPSOL, then its status is unknown.
2. If any linear constraints are not satisfied, then the group is not satisfied.
3. If any linear constraints are active, then the group is active.
4. The status is satisfied.

The expert system uses this data to inform the designer of the conflict. A typical message, generated by the expert system to describe this type of conflict, is shown below :

```
The frequency domain constraints on DY[2,1] could
not be satisfied as they conflict with :
  the frequency domain constraints on DY[1,1]
  the time domain constraints on NU[1,1]
Of these, the Lagrange multipliers hint that the
frequency domain constraints on DY[1,1] are the
most probable cause of the conflict.
It is unlikely that the asymptotic constraints
are responsible for the conflict.
```

The likelihood of the asymptotic constraints contributing to the conflict, as well as the specification most probably causing the conflict, is determined by analyzing the values of the Lagrange multipliers computed in the QPSOL algorithm. Although subject to scaling, the more positive the value of the multiplier, the more likely the corresponding linear constraint is a limiting factor in the design. The inverse

argument is commonly used in active set methods, where the constraint with the most negative multiplier is assumed to have the least impact on the solution, and is deleted from the active set.

In addition to explicitly describing the cause of the conflict as shown above, the expert system can also provide the user with some assistance in resolving the conflict. One way this is done is by diagnosing some particular conflicts. For example, given the close relationship between the NY and DY responses, specifications placed on both responses may conflict.

4.5 Formulating the design specifications.

To assist the user in drawing up and refining a specification, the expert system considers a list of common design attributes, the characteristics of the plant, and the current specification. These features are implemented as a collection of frames (Winston and Horn, 1984). The expert system does not have knowledge of many physical requirements of the control system, such as the limitations of the actuator. For example, while the expert system knows that a good design will often constrain the magnitude of the control signal response following a disturbance, and that to improve the speed of response it may be necessary to relax this constraint, it does not know whether a particular value for the constraint is satisfactory or not. Thus it can only remind the designer of these issues; it is then up to the designer to take the engineering decisions.

There are two mechanisms to assist the designer in formulating and refining the specification. They are invoked using the NEXT STEP and SUGGEST commands respectively, and are discussed in detail below.

The NEXT STEP command initiates and perpetuates a step-by-step design mode to help novice designers produce a reasonable design, and at the same time assist them in

learning to use the design tool effectively. This procedure guides the user through a sequence of design steps, at each stage recommending the next design step and explaining why it may be necessary. The designer is free and encouraged to experiment with other commands during this process. Usually the recommended design step is not executed automatically, but the designer is only shown how to perform it, and is then expected to follow the instructions if required. In this way it is hoped that the user will learn to use the design tool independently.

An example of the use of the NEXT STEP command is shown below :

```
DY[1,1](frequency) >NEXT STEP
Do you want the amplitude of the control signal
to be limited following a step input ? /EXPLAIN?

Constraints on the RU and NU step responses limit
the control action taken following a command
input change. This in turn slows down the closed
loop system.  Since real actuators cannot provide
arbitrarily large control actions, it is good
practice to constrain this response; then
(hopefully) the design will be less susceptible
to saturation problems when implemented.

Do you want the amplitude of the control signal
to be limited following a step input ? YES

You need some specifications which will limit the
control signal at output U following a step input
at R. The best way to do this is to constrain the
time response of the RU signals; usually it is
necessary to place constraints on only the
initial part of this response. Another way to
achieve this is to constrain the frequency
response, in particular the high frequency
portion. The sequence of commands to use is
          >RU
          >TIME
          >EDIT ALL
```

In this example, the suggested sequence of commands first selects the RU step response. The EDIT command is then used to enter or modify the constraints on this response, which are presented in a spreadsheet format by the CACSD package. The ALL qualifier indicates that the constraints set will

apply to all elements RU[i,j] of the response.

The SUGGEST command is similar to the step-by-step design mode. Guided by appropriate questions, and considering the state of the design in terms of the current specifications and performance, the system can offer suggestions on how to deal with conflicting constraints, which specifications to modify in order to improve performance, or how to satisfy specifications not explicitly considered by the CACSD package. For example,

```
RY[1,1](time) >SUGGEST
Checking plant DC gain.
A steady state value of at least 2 will be
required at some input of the plant to achieve
asymptotic tracking following a unit step input.
Is this value acceptable ? YES
Do you require asymptotic rejection of
disturbances at input D ? YES

Complete asymptotic disturbance rejection
specifications have not yet been set.  These may
be specified using the 'SET ASYM REJECTION'
command.

....

DY[1,1](frequency) >SUGGEST
Do you wish to improve the closed loop
performance of the design ? YES
The design is completed in two independent
sections, refered to as Q1 and Q2. These
correspond to the closed loop responses as
follows :
   Q1 : responses RY and RU
   Q2 : responses DY, NY, NU, VY, and VU
Which of these sections (1 or 2) are you
interested in ? 2
Do you want me to check the design for unusual
combinations of optimization specifications ? YES

I did not find any abnormalities in the
optimization specifications.

Which column of the closed loop response are you
interested in ? 2
Are you prepared to relax any of the constraints
in order to improve the design performance ?
/EXPLAIN
The analysis of the Lagrange multipliers from the
last 'SOLVE' command indicates that some of the
constraints are limiting the design performance.
```

```
Are you prepared to relax any of the constraints
in order to improve the design performance ? YES
It should be possible to improve the disturbance
rejection performance by relaxing the time domain
constraints on column 2 of the NU responses.
Would you like to try this ? YES

Use the 'EDIT' command to relax the time domain
constraints on NU[1,2].
```

In most cases a particular suggestion is made only once; the designer is expected to take note of it at that time, and keep it in mind during the remainder of the design process. The decision to do this was taken in view of those situations where the designer does not wish to follow the recommendation given, and should not be repeatedly reminded of it. This should suit experienced designers, but not the novice designer who inadvertently ignores important advice. Some relief for this latter case is available, however; after all the relevant suggestions have been made, the designer is given the option of cycling through the suggestions once more.

There are two other more specific SUGGEST commands. The SUGGEST OPT command provides advice on using the optimization facilities to achieve the required performance. The SUGGEST EDIT command provides advice on the possibilities for changing the performance by placing constraints on the selected response.

4.6 Expanding the scope of the CACSD package.

The expert system expands the scope of the CACSD package through checking on aspects of the design not explicitly covered by the specification, and effectively extending the range of specifications which may be used.

In every design method, some specifications are treated explicitly, and others implicitly. The latter can only be satisfied by judicious choice of the former, making design an art. For example, the CACSD package used deals with constraints on individual closed loop responses directly; to

meet specifications on the singular values of the open or closed loop frequency responses requires careful design of the individual closed loop responses.

There may also be some specifications which a given design method cannot handle; for example the CACSD method used here cannot deal with requirements on the controller structure. A higher level expert system could be used in turn to select the most appropriate design method in view of the characteristics of the plant and specifications.

The expert system provides advice on how to meet some of the implicit constraints, and checks that the design actually satisfies them. For example, the designer may wish to impose constraints on the singular values of the open loop frequency response. These implicit constraints are treated indirectly in the sense that they are not considered by the CACSD package when solving for a controller. However, when set, the expert system can provide advice on how the individual closed loop responses should be constrained in order to meet them. In addition, after a new controller has been computed, the expert system checks that they were satisfied. If not, advice on possible modifications to the specification to account for them is also available if desired. The example below shows the expert system checking on the constraints set on the singular values (SVD) of the open loop (GK) frequency response :

```
DY[1,1](frequency) >SUGGEST
The SVD constraints on GK have not been
satisfied. Do you want help with these ? YES
Is the problem over the low or high frequency
part of the SVD plot ? (enter '/EXPLAIN' to see
the graph) : LOW

Constraints (minimum of the minimum singular
value) on the low frequency part of the GK
response can be met by placing constraints on the
low frequency part of the DY response. In
general, it is necessary to improve the
performance of the Q2 section of the design; this
goal may also be achieved through optimization of
the DY or NY step responses.
```

5. IMPLEMENTATION

The expert system, running under the CXS rule-based expert system shell, has been implemented on a low cost personal computer. It is linked to the memory resident CACSD package on the same computer, using an efficient message passing interface. At present there are approximately 400 rules in the knowledge base. Figure 4 illustrates the overall structure of the design tool; a similar structure is found in each of the applications discussed in the introduction.

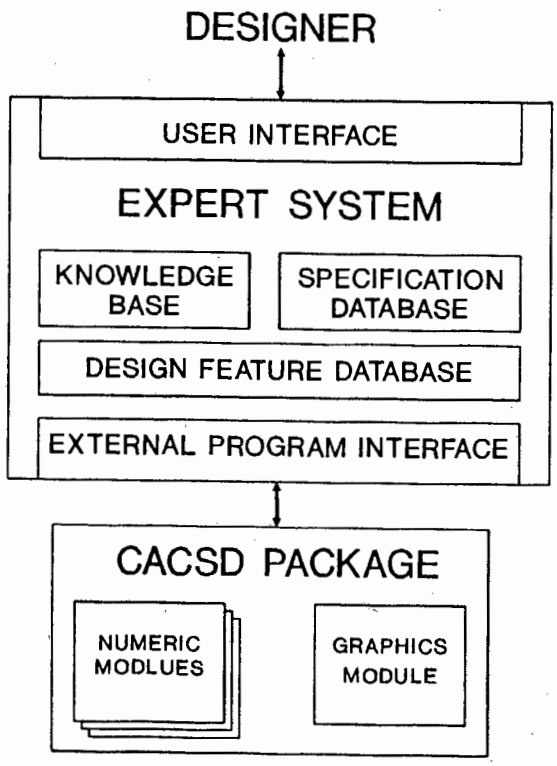


Fig 4. Structure of the design tool.

Expert systems have been effective in diagnosing problems and analyzing situations (Hayes-Roth *et al.*, 1983). Given the state of the problem, they can be programmed to advise a suitable course of action. Control system design, however, is a feedback process (MacFarlane *et al.*, 1989), and thus forms an inherently non-monotonic logic system. Nevertheless, once given a particular set of specifications, an effectively piecewise monotonic logic system results, and the expert system can be designed along similar lines. Here

the situation or state of the problem includes the command line and some history of the commands, the current specifications and performance of the design, the status of each design constraint, and the history of advice already given.

Much of the specification is stored in a CXS database. The records of this database contain the following fields :

DOMAIN	"Time" or "frequency".
RESP	The particular closed loop response, for example "DY".
COL	The input number, 1...n
ROW	The output number, 1...n
Q	The Q matrix which relates to this response, i.e. 1 for Q1 or 2 for Q2.
STATUS	The status of the constraints on this response : "satisfied", "active", "unsatisfied", or "unknown".
ASYM	The asymptotic value required for this response. If none, then the value "none" is stored.
OPT	The optimization weight for this response. If none, then the value "none" is stored.
RELAX	If the designer has indicated that this response cannot be relaxed, the value "no" is stored; otherwise "unknown".

CXS also provides pattern matching facilities which are useful in analyzing the specification. A typical rule from the knowledge base, using this database, is :

```
IF can_relax_some is "yes"
AND opt_q is 1
AND q1_objective is "tracking"
AND opt_col isnt "unknown"
AND FIND spec_db [=T0,"RU",opt_col,*,*,
                        "active",*,*,"unknown"]
AND can_relax_ru isnt "no"
THEN
  sugst_opt is "done"
```

```
  DISPLAY
Use the 'EDIT' command to relax the \%T0% domain
constraints on column \%opt_col% of the RU
response. This should improve the tracking
performance.
END

ASK can_relax_ru ["yes","no"]
It should be possible to improve the tracking
performance by relaxing the \%T0% domain
constraints on column \%opt_col% of the RU
response. Would you like to try this ?
```

This rule is used to suggest that, if the tracking performance is to be further improved, it may be necessary to relax the constraints on the RU response. Note that if the FIND clause finds a record in the database with fields RESP = "RU", COL = opt_col, STATUS = "active", and RELAX = "unknown", the value of the domain field is assigned to the temporary variable T0. Also shown above is the corresponding question which is put to the user.

While the actual constraints on the responses are stored within the CACSD package, the expert system has efficient access to these through the message passing facility. The performance of the design is usually analyzed within the CACSD package, and the results then transfered to the expert system. In some cases the designer is shown a response graph, and then asked qualitative questions about it, for example whether or not the high frequency gain is excessive. Using this knowledge, and rules of the form above, the specification and state of the design is readily analyzed.

6. CONCLUSION

An expert system interface to a CACSD package has been implemented to produce an intelligent, interactive design tool. The expert system assists the designer in using the CACSD package, in formulating and refining the design specification, and in dealing with conflicting constraints. It has also been used to effectively extend the scope of the design method, as well as integrate information from

additional analyses of the design.

Nevertheless there remains a chasm between artificial and real intelligence. Since the designer has real intelligence, the aim of the expert system has been to assist and complement, rather than replace, the designer. This produces a team solution, capitalizing on the strengths of each member.

7. ACKNOWLEDGEMENTS

Financial support for this research from AECI Ltd. and the Foundation for Research Development is gratefully acknowledged.

REFERENCES

Birdwell, J. D. *et al.* (1985). Expert Systems techniques in a Computer-based Control System Analysis and Design Environment. *Proc. 3rd IFAC/IFIP Symp. on Computer Aided Design in Control and Engineering Systems*, Lyngby, Denmark.

Birdwell, J. D. (1987). "An Expert System can aid in the Evolution of a Design Methodology", *Proc. American Control Conference*.

Boyd, S. P. *et al.* (1988). A New CAD Method and Associated Architectures for Linear Controllers. *IEEE Trans. Aut. Control*, **AC-33**, 268-283.

Boyle, J.-M, G. K. H. Pang, A. G. J. MacFarlane (1989). "The development and implementation of MAID: a knowledge based support system for use in control system design", *Transactions of the Institute of Measurement and Control*, 11, no. 1, 25-39.

Edmunds, J. M. (1979). "Cambridge linear analysis and design program", *Proc. 1st IFAC Symposium on Computer-Aided Control System Design*, Zurich, 253-258.

Genesereth, M. R. (1982). The role of plans in intelligent teaching systems. In D. Sleeman and J. S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, London.

Hayes-Roth, F., D. A. Waterman and D. B. Lenat (1983). *Building Expert Systems*. Addison-Wesley, Reading, MA.

Jackson, P., and P. Lefrere (1984). On the application of rule-based techniques to the design of advice-giving systems. In M. J. Coombs (Ed.), *Developments in expert systems*, Academic Press, London.

James, J. R., D. K. Frederick and J. H. Taylor (1987). Use of expert-system programming techniques for the design of lead-lag compensators. *IEE Proc. Part D*, **134**, 137-144.

James, J. R, D. K. Frederick, P. P. Bonissone, J. H. Taylor (1986). "A retrospective view of CACE-III: considerations in co-ordinating symbolic and numeric computation in a rule based expert system", *Proc. 2nd Conf. on A.I. Applications*, Miami Beach, Florida.

Larsson, J. E, P. Persson (1986). "Knowledge Representation by Scripts in an Expert Interface", *Proc. American Control Conf.*, Seattle, USA, 1159-1162.

Little, J. N, A. Emami-Naeini, S. N. Bangert (1985). "CTRL-C and Matrix Environments for the Computer-Aided Design of Control Systems", in M. Jamshidi, C. J. Herget, eds., *Computer-Aided Control Systems Engineering*, Amsterdam: North-Holland, 111-124.

MacFarlane, A. G. J., G. Gruebel and J. Ackermann (1989). Future Design Environments for Control Engineering. *Automatica*, **25**, 165-176.

Nolan, P. J. (1986). "An Intelligent Assistant for Control System Design", *Proc. 1st Int. Conf. on the Application of Artificial Intelligence in Engineering Problems*, University of Southampton, U.K.

Pang, G. K. H. and A. G. J. MacFarlane (1987). *An Expert Systems Approach to Computer-Aided Design of Multivariable Systems*. Springer-Verlag, Berlin.

Pang, G. K. H, M. Vidyasagar, A. J. Heunis (1990). "Development of a New Generation of Interactive CACSD Environments", *IEEE Control Systems Magazine*, **10**, no. 5, 40-44.

Scales, L. E (1985). *Introduction to Non-Linear Optimization*, London: MacMillan Publishers.

Taylor, J. H. and D. K. Frederick (1984). An Expert System Architecture for Computer-Aided Control Engineering. *Proc. IEEE*, **72**, 1795-1805.

Tebbutt, C. D. (1990a). "An Efficient Representation for Linear Constraints", *IEEE Trans. Aut. Control*, **35**, no. 8, 949-951.

Tebbutt, C. D. (1990b). "A Microcomputer Implementation of Multivariable Factorization Theory". Submitted to *IEEE Trans. Aut. Control*.

Tebbutt, C. D (1990c). "Expert Systems Approach to Controller Design", *IEE Proceedings Part D*, to appear.

Tebbutt, C. D, (1990d). "An Expert System for Controller Design", *Trans. (SA)IEE*, **81**, no. 3, 15-19.

Trankle, T. L, P. Sheu, U. H. Rabin (1986). "Expert System Architecture for Control System Design", *Proc. American Control Conf.*, Seatle, USA.

Vidyasagar, M. (1985). *Control System Synthesis: A Factorization Approach*, Cambridge, MA: M.I.T. Press.

Winston, P. H, B. K. P. Horn (1984). *Lisp (2nd edition)*, Reading, MA: Addison-Wesley.