COMPUTER IMPLEMENTATION OF OPTIMAL MULTIVARIABLE CONTROLLER DESIGN IN THE FREQUENCY DOMAIN

By

JOHN EDWARD PERRAULT, JR.

Bachelor of Science in Mechanical Engineering
University of Tulsa
Tulsa, Oklahoma
1975

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1977

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
DOCTOR OF PHILOSOPHY
May, 1981

COMPUTER IMPLEMENTATION OF OPTIMAL MULTIVARIABLE CONTROLLER DESIGN IN THE FREQUENCY DOMAIN

Thesis Approved:

Lynn R Ebbesen
Thesis Adviser

James H Taylor

Larry Litke

Karl M. Reed

Ronald D. Rhodes

Norman N Durham
Dean of the Graduate College

ACKNOWLEDGEMENTS

Financial support for this research was provided by the Air Force Weapons Laboratory (AFWL), Air Force Systems Command, Kirtland Air Force Base, New Mexico, under Contract F29601-78-C-0038. Models for the examples of Chapter IV are from the Airborne Pointing and Tracking System currently under development at AFWL.

I thank my advisor, Dr. Lynn R. Ebbesen, who provided constant encouragement throughout my graduate studies at Oklahoma State University and many enjoyable and enlightening conversations. I also thank the other members of my advisory committee, Dr. Karl N. Reid, Dr. James H. Taylor, Dr. Ronald P. Rhoten, and Dr. Larry D. Zirkle, as well as my many colleagues for their help in the many aspects of my graduate studies.

Charlene Fries provided valuable assistance in the preparation of the final manuscript for this thesis. Thank you.

Finally, I thank my parents for their support and encouragement throughout my entire education, and my wife Debbie for her patience and kind understanding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

$A_p(s)$ - Transfer function matrix used to specify plant in general model representation

$B_p(s)$ - Vector used to specify plant input in general model representation

$C(s)$ - Transfer function matrix of optimal controller to be designed

$d(s)$ - Vector representing plant disturbances

$e(s)$ - Vector representing control system errors

E_s - Mean-square value of sensitive plant input

E_t - Mean-square value of control system errors

E - Total system performance measure, $E=E_t+kE_s$

$F_e(s)$ - Transfer function matrix used to specify feedback sensor pre-equilization

$F_o(s)$ - Transfer function matrix used to specify feedback sensor noise coupling

$F_t(s)$ - Transfer function matrix used to specify feedback sensor dynamics

$G_d(s)$ - Matrix used to specify load disturbance spectral densities

$G_\ell(s)$ - Matrix used to specify feedforward measurement noise spectral densities

$G_m(s)$ - Matrix used to specify feedback measurement noise spectral densities

$G_u(s)$ - Matrix used to specify reference input spectral densities

k - Performance measure input weighting constant

$\ell(s)$ - Vector representing feedforward sensor noise input

$L_e(s)$ - Transfer function matrix used to specify feedforward sensor pre-equalization

$L_o(s)$ - Transfer function matrix used to specify feedforward sensor noise coupling

$L_t(s)$ - Transfer function matrix used to specify feedforward sensor dynamics

$m(s)$ - Vector representing feedback measurements noise input

$n(s)$ - Vector representing system input noise

$P(s)$ - Transfer function matrix used to specify plant dynamics

Q_t - Performance measure error weighting matrix

$r(s)$ - Vector representing command input to plant

$R_p(s)$ - Vector representing all plant output in general model representation

$S(s)$ - Sensitivity matrix, $S(s)=(I+F(s)P(s)C(s))^{-1}$

$u(s)$ - Vector representing filtered system reference input

$u_i(s)$ - Vector representing ideal system reference input

$v(s)$ - Vector representing plant feedback measurements available to controller

$W(s)$ - Transfer function matrix used to specify input pre-filtering

$y(s)$ - Vector representing the plant outputs to be controlled

$z(s)$ - Vector representing plant feedforward disturbance measurements available to controller

CHAPTER I

INTRODUCTION

Since the early 1960's when Kalman [1] introduced state-space methods into optimal control theory, most of the advancements in control system synthesis have utilized the time-domain techniques. The current popularity of the state-space design and analysis theory is evident from the vast amount of literature which has been published. The so-called Linear-Quadratic-Gaussian (LQG) theory [2] is the cornerstone for a large class of significant developments.

Although LQG and related time-domain synthesis techniques still dominate the literature, many control engineers prefer frequency-domain design methods. Results are usually easier to interpret and compare in the frequency-domain and engineering design specifications are simpler and more practical. Because of its continued use in practice, frequency-domain synthesis theory is beginning to reappear in the literature and recently has been gaining more attention.

A variety of frequency-domain design methods exists such as trial and error, pole shifting (modal), and optimal multivariable techniques. Of these techniques, the optimal methods are the only true synthesis methods relying mostly on mathematics to provide suitable controllers while the other types require a fair amount of design experience to arrive at satisfactory results. Optimal design techniques are used to find controllers which optimize some predetermined measure of overall

system performance. Performance measures for frequency-domain design methods usually consist of minimization of the mean square steady-state error between system input and output.

Optimal design methods in the frequency-domain parallel the LQG techniques in the time domain; however, the frequency-domain theory offers several advantages. Among these advantages the major ones are:

1. Plants do not require state-space representations, only rational transfer functions are needed.

2. Dynamical sensors can easily be incorporated into the design.

3. Colored noise does not have to be treated as a special case.

4. Simpler controllers can often be found.

Frequency-domain methods have some drawbacks which may make the theory difficult to utilize. One drawback is the need for accurate plant models including good rational transfer function approximations for details such as process lags. Load disturbances and measurement noise must be representable by rational spectral density functions, and these are not always available or easily obtainable. These problems are present in most optimal design procedures although they can often be circumvented such that valid results can be obtained.

The most serious obstacles to the successful application of frequency-domain multivariable controller design are the required algebraic computations. These computations include spectral factorization, inversion, canonical decomposition, and partial fraction expansion of rational polynomial matrices. Additionally, the basic polynomial operations of addition, subtraction, multiplication, division, and the calculation of the greatest common divisor between two or more polynomials have inherent numerical problems which add to the difficulties of the over-all

computation. These computations are difficult to perform manually even for the design of simple systems and are virtually impossible to do manually for more complex multivariable designs.

The digital computer offers a viable tool to aid in the computation of optimal controllers. Once a computer program has been developed which is capable of performing the entire computation there should be a sizeable increase in the amount and types of application of the optimal theory. The intent of this research was to study the development of such a program.

Scope and Objectives

The scope and objectives of this study are summarized as follows:

1. Pick from the available optimal frequency-domain theory the one method which would yield the most benefit once implemented in a computer program.

2. Develop a generalized method for the representation of the plant model and the introduction of its associated transfer function matrices into the design process.

3. Investigate the various methods which could be used to represent polynomials in a computer program. Investigate the numerical problems associated with each method of representation. Select the method which will function best in the overall design program in terms of numerical accuracy.

4. Develop a general prototype computer program which will compute the optimal controller based on the theory selected under the first objective. The resulting program should be general enough to allow testing

of various basic algorithms and accommodate a moderate range of multivariable systems.

5. Demonstrate the program with an example. Compare the performance of the resulting controller with that of controllers that already exist. Use computer simulations of the system response for the comparison.

Plan of Presentation

Chapter II provides background information related to this study. Major historical developments related to optimal frequency-domain controller design are presented in the chapter as well as a review of current literature related to theory and algorithmic procedures. The first section of Chapter III describes the design theory which was implemented in the program with Appendix A providing the remaining details. The last section of Chapter III describes the generalized model representation theory developed by this study.

During the course of this research, three major algorithmic techniques were considered for use in the controller design program. Chapter IV summarizes the advantages and disadvantages of each method. Chapter V presents the algorithmic technique finally chosen and outlines the manner in which various operations, such as partial fraction expansion and polynomial matrix inversion are computed in the prototype program. Appendix B describes the mechanical structure of the program. An example illustrating the design process and use of the program is presented in Chapter VI and the conclusions and recommendations for future study are given in Chapter VII.

CHAPTER II

BACKGROUND

Historical Developments

The major impetus to optimal frequency-domain control theory seems to have arisen out of Wiener's famous work in filtering and prediction [3]. In this work, Wiener demonstrated the solution of the Wiener-Hopf integral equation which results from the minimization of the mean-square error between the actual output of a filter and the desired or ideal output. By working in the frequency-domain and using a technique known as spectral factorization, he was able to solve the equation and obtain the realizable filter transfer function directly.

Later, Newton, Kaiser, and Gould [4] published a text demonstrating how mean-square error minimization and the Wiener-Hopf solution could be used to obtain optimal compensators for single-input, single-output feedback systems. The text appears to be the first publication to thoroughly discuss the optimal design of control systems in the frequency domain, addressing such problems as sensor dynamics, process and measurement noise, and plant saturation. Their methodology suffered from a major drawback that only open-loop stable, single-input, single-output plants could be accommodated. Their work considered the solution of the fixed-configuration, semi-free-configuration, free-configuration Wiener problems.

A number of related papers were later published which extended the

work of Newton et al. [4]. Amara [5] solved the multivariable free-configuration Wiener problem and demonstrated the use of matrix spectral factorization. Hsieh and Leondes [6] first developed a solution for the semi-free-configuration Wiener problem which required solving a set of simultaneous algebraic equations avoiding the need to perform spectral factorization. However, they did not prove that a solution to their equations existed and it was later shown by Davis [7] that their method failed in some cases. Bongiorno [8] also solved the semi-free-configuration problem attempted by Hsieh and Leondes using matrix spectral factorization.

All of the previous design methods were unable to accommodate unstable plants and required the plant or process being controlled to be open-loop stable from the start. Concurrently, several researchers were investigating the questions of stability and physical realizability associated with the synthesis of multivariable feedback control systems [9, 10, 11, 12]. Right-half plane pole-zero cancellations within a feedback loop were considered first by Ragazzini and Franklin [13] in their early work with sampled data systems. An analogous treatment for continuous-time systems was presented by Bigelow [14]. Even with these investigations, it was still some time later before the questions of stability were fully understood and the restrictions removed from frequency-domain synthesis methods.

The next largest advance in the theory appears to have occurred with the study of Weston and Bongiorno [15] who extended the work of Newton et al. [4] to the multivariable system. Their investigation determined the manner in which load disturbance, measurement noise, and plant saturation effects could be incorporated into multivariable

design processes. The plant matrix could be rectangular but was subject to the condition that the number of plant output did not exceed the number of input. The method also required that the plant be open-loop stable.

Several other contributions to the frequency-domain optimal control theory exist and have been published in various journals [16, 17] and texts [18, 19, 20]. However, these developments have been overshadowed by more recent ones. Various investigations into other methods which are not strictly optimal have also been reported. Examples include the inverse Nyquist array method of Rosenbrock [21] and the characteristic loci methods of Belletrutti and MacFarlane [22, 23]. Others include the pole shifting or modal techniques [24]. The use of these types of methods usually require a greater amount of design experience and are often incorporated into interactive type computer design programs [25].

Two complete surveys have been published briefly describing the various optimal and nonoptimal design techniques which have been investigated and reported over the previous years [26, 27].

Current Status

A significant result in optimal frequency-domain synthesis theory has recently been published by Youla, Bongiorno, and Jabr [28, 29]. This work has contributed greatly to the overall optimal frequency-domain design theory and appears to be the most comprehensive frequency-domain synthesis technique to date. The questions of stability have been answered as well as other engineering considerations such as steady-state error and sensitivity. The method is general enough to accommodate open-loop unstable and/or non-minimum phase plants with no restrictions

on the number of input and output. Both colored and white noise can be accommodated as well as plant saturation effects. The method applies to both single-input, single-output, and multivariable plants.

The duality between the time-domain and frequency-domain methods for the solution of stochastic, multivariable, optimal control problems has been demonstrated by MacFarlane [30], Barrett [31], and Shaked [32]. Youla et al. [29] also showed the duality between their methods and time-domain methods. They further demonstrated the manner in which simpler, suboptimal controllers could be found by their method and not by the time-domain methods.

Optimal frequency-domain synthesis requires factorization and manipulation of polynomial matrices which present formidable computational difficulties. For these reasons, implementation of the methods requires the use of automatic computers to carry out the calculations, even if the order of the plant is relatively low. Any simplifications of the design techniques can be useful in reducing the computational burden.

A few recent studies have been made which consider simplifications to the methods of Youla et al. [29]. Grimble [33] describes a method which he reports to be easier to implement than that of Youla et al. [29]. The advantages seem to be cancelled by the fact that his method requires calculating three separate controllers, two of which are open-loop and are not quite satisfactory in terms of sensitivity. His work, however, answers some important questions about inputs consisting of both deterministic and stochastic components. Another work by Bongiorno [34] demonstrates how the theory in reference [29] can be used in part to obtain satisfactory controllers, but the method described is not optimal and requires intuition on the part of the user.

Studies related to the computational aspects of and the numerical problems associated with a complete optimal controller synthesis program do not exist. However, some results have been published describing algorithms for computing various parts comprising the overall problem. In part, the object of this research was to explore the problems which arise when the various computational parts are combined into one complete procedure.

Most of the studies in the literature related to computations involving rational polynomials and rational polynomial matrices fall into one of two general categories. The first category is comprised of exact computation methods. These methods assume the coefficients of the polynomials can be represented as exact rational fractions with the solution represented likewise. The second category consists of the methods which utilize the more usual floating-point arithmetic.

Unique to the exact methods is a special purpose programming language known as REDUCE 2 [35]. REDUCE is a very powerful symbolic manipulator whose primary function is the algebraic manipulation of rational polynomials. The main disadvantage of this programming system is its inability to factor polynomials or perform division of polynomials, two necessary computations required for spectral factorization, co-prime decomposition, and partial fraction expansion of rational polynomial matrices. The use of REDUCE 2 is considered in Chapter IV.

Basic principles of exact polynomial arithmetic are summarized in two texts [36, 37]. Recent contributions are directed toward more specific algorithms, such as those of McClellan [38], Horowitz and Sahni [39], and Gentleman and Johnson [40], all of which are concerned with the computation of the determinant of polynomial matrices. These

algorithms require the coefficients of polynomials to be represented as rational integer fractions. Operations are then performed using both the numerator and denominator of each coefficient. During the course of the operations, the numerator-denominator pair must be constantly reduced to its lowest prime form to prevent excessive coefficient growth. Coefficient growth, also known as "intermediate expression swell" [38], is the greatest difficulty in the use of exact computation methods.

The use of alternative number systems for exact computations has also been investigated by a few authors. Knuth [37] presents a complete treatment of modular or residue arithmetic. Addition, subtraction, and multiplication are easily performed using residue arithmetic; however, division cannot be performed in any similar manner.

Rao [41] has proposed the use of finite field transforms using a p-adic number system. His approach to exact arithmetic combines the best features of the usual p-ary number system and residue arithmetic. Some additional work has been done using this type of arithmetic which is directly related to the computation of optimal controllers [42, 43]. Again, these methods seem hampered by the coefficient growth problem mentioned above and, for purposes of this study, by lack of an explicit spectral factorization algorithm.

Many algorithms dealing with rational polynomial matrices and using floating-point arithmetic have been published. Matrix spectral factorization, a critical step in the optimal controller synthesis process, was first developed into a numerical algorithm by Youla [44]. Later, Tuel [45], devised an algorithm for spectral factorization based on an iterative procedure used to solve a set of equations similar to

steady-state matrix Riccati equations. Anderson, Hitz, and Diem [46] also devised a recursive technique that is similar to Tuel's algorithm. Davis [47] and Grimble [48] have reported spectral factorization algorithms which are of a non-recursive nature; however, Tuel's algorithm remains the most popular.

The inversion of rational polynomial matrices, also a key step in controller synthesis, has been addressed by Downs [49], and Munko and Zakian [50]. The decomposition of polynomial matrices to Smith form is discussed by Pace and Barnett [51, 52]. More basic algorithms pertaining to polynomial arithmetic are also available [37, 53, 54].

The calculation of the greatest common divisor between two polynomials is an extremely important calculation in the controller synthesis theory, and efficient algorithms are mandatory. There exist ample studies related to the greatest common divisor problem [55, 56]. However, the lack of adequate error analysis, and information pertaining to the range of problems which can be successfully handled by the algorithms makes the validity and usefulness of the procedures questionable. In fact, most of the algorithms which utilize floating-point arithmetic were demonstrated with rather trivial examples and lacked adequate error analysis and range of problem information. As a result some of these algorithms, when implemented as presented in the literature, are not usable in the overall controller synthesis design program.

CHAPTER III

THE CONTROL SYSTEM MODEL

Optimal Controller Design Problem

The multivariable controller synthesis theory of Youla, Bongiorno, and Jabr [29] was selected for use in this study. The theory is general enough to accommodate a large class of both single-input single-output and multivariable design problems. Additionally, the computations required by the various steps of this design process are representative of those required by most of the optimal frequency-domain synthesis theory in existence. By implementing the selected theory in a digital computer program a general problem has been considered. Later development of programs for less complex theories (or suboptimal theories) should present few problems.

The remainder of this section outlines the control system model on which this study was based. The theoretical details of the actual synthesis procedure are provided in Appendix A.

The following notation will be used in the remainder of this thesis. The transpose, inverse, trace, and determinate of a matrix A will be denoted by A^T, A^{-1}, TrA, det A, respectively. I_n represents the n x n identity matrix and 0_{nm} represents the n x m zero matrix.

The control system configuration considered by Youla, Bongiorno, and Jabr [29] and in this research is shown in Figure 1. In the figure, P(s) is an n x m matrix of rational transfer functions representing the system plant. F(s) is an n x n matrix containing the feedback sensor dynamics.

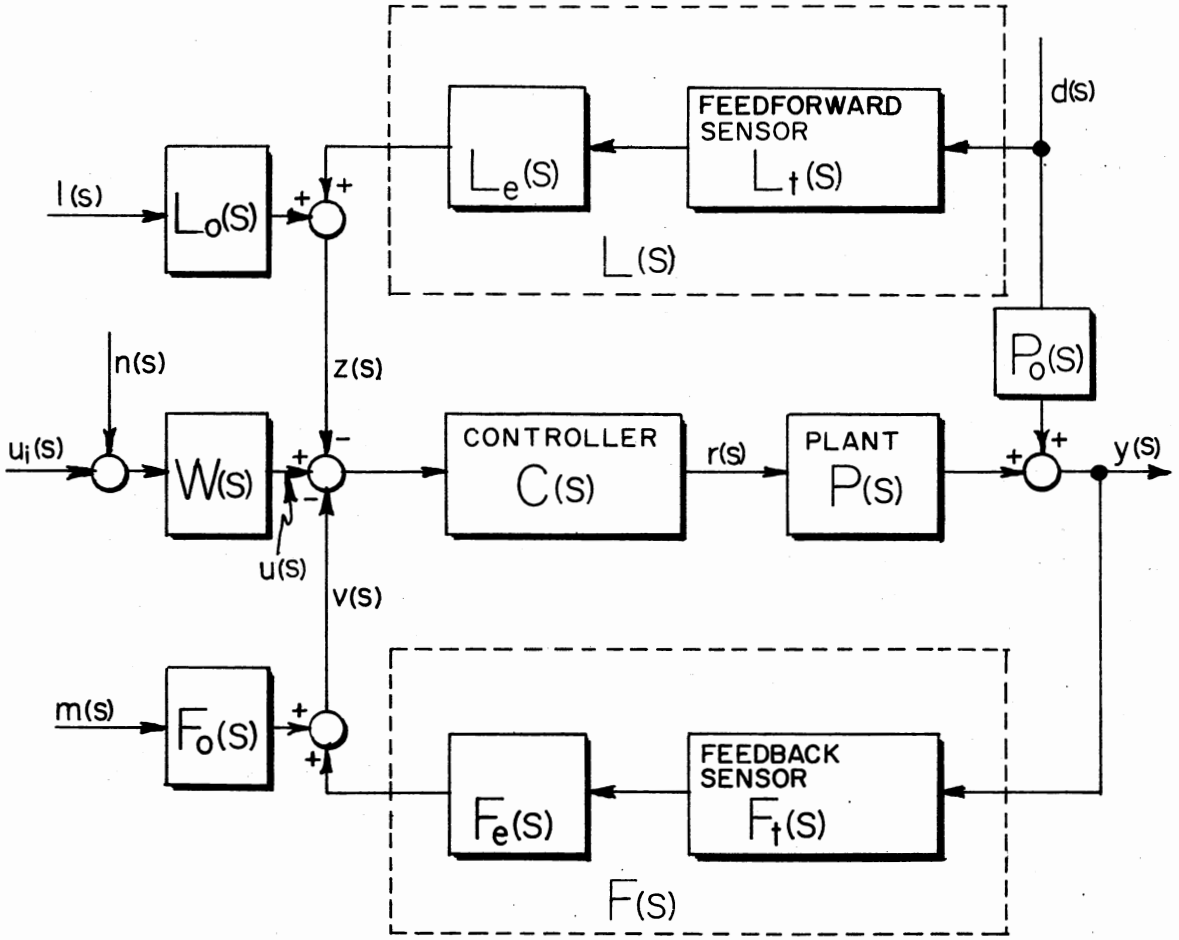

Figure 1. Multivariable Control System Configuration

$L(s)$ is optional and represents disturbance feedforward sensor dynamics. Matrix $C(s)$ is the $m \times n$ controller to be determined.

Plant disturbance and measurement noise are included by assuming that

$$y(s) = P(s)\, r(s) + P_o(s)\, d(s) \tag{3.1}$$

$$v(s) = F(s)\, y(s) + F_o(s)\, m(s) \tag{3.2}$$

$$z(s) = L(s)\, d(s) + L_o(s)\, \ell(s) \tag{3.3}$$

where $P_o(s)$, $F(s)$, $F_o(s)$, $L(s)$, and $L_o(s)$ are also real rational matrices and are of compatible dimension.

In some control system designs, feedback alone will not suffice in the suppression of load disturbance and feedforward is advisable. This feedforward is accomplished by measuring the disturbance via the sensor matrix $L_t(s)$. In many practical problems the choices of physical sensing devices $L_t(s)$ and $F_t(s)$ is restricted and dictated by the problem. Low power pre-equalizers $L_e(s)$ and $F_e(s)$ can and in many cases should be used to improve stability margin, to assure zero steady-state error, and to incorporate delay in the feedback path [29].

It is assumed $P(s)$, $P_o(s)$, $L_t(s)$, $L_o(s)$, $F_t(s)$, $F_o(s)$ are known;

$$F(s) = F_e(s)\, F_t(s) \tag{3.4}$$

and

$$L(s) = L_e(s)\, L_t(t). \tag{3.5}$$

Additionally, the spectral densities of $u(s)$, $d(s)$, $\ell(s)$, and $m(s)$ must be specified and are denoted by $G_u(s)$, $G_d(s)$, $G_\ell(s)$, and $G_m(s)$, respectively.

If $y_d(s)$ is the desired closed-loop response, it can be related to

the actual set point input signal $u_i(s)$ by

$$y_d(s) = T_d(s)\ u_i(s) \tag{3.6}$$

where $T_d(s)$ is an ideal transfer matrix. If $T_d(s)$ is embedded within the prefilter matrix $W(s)$ and $W(s)$ is selected in advance, then

$$u(s) = W(s)\ (u_i(s) + n(s)) \tag{3.7}$$

is the best available approximation of $y_d(s)$. The performance measure is based on the vector error

$$e(s) = u(s) - y(s) \tag{3.8}$$

where $y(s)$ is the actual plant output.

The performance criteria is given as

$$E_t = \frac{1}{2\pi j} \mathrm{Tr} \int_{-j\infty}^{j\infty} < e(s)\ Q_t e^T(-s) > ds \tag{3.9}$$

where Q_t is a non-negative definite weighting matrix and $<\cdot>$ denotes ensemble average. Similarly, if $P_s(s)$ represents the transfer matrix coupling the plant input, $r(s)$, to the plant states which must be protected against saturation effects, then

$$E_s = \frac{1}{2\ j} \mathrm{Tr} \int_{-j\infty}^{j\infty} < P_s(s)\ r(s)\ r^T(-s)\ P_s^T(-s) > ds \tag{3.10}$$

is a proven penalty function [4]. Hence, the total cost can be formulated as

$$E = E_t + kE_s \tag{3.11}$$

where k is a positive adjustable constant used to trade off linear performance with system accuracy [29].

The General Control System Model Representation

As mentioned in the previous section, a user must supply the plant matrix, P(s), the feedback matrix, F(s), the feedforward matrix, L(s), and the additional transfer function matrices, $P_o(s)$, $F_o(s)$, $L_o(s)$, and $P_s(s)$ before the synthesis process begins. However, in larger multivariable plants which have a high degree of interconnection and several inner control loops the required transfer function matrices may not easily be determined. In this section a generalized method for representing the plant model, which can be used by a computer program to automatically determine the necessary transfer function matrices, is outlined.

The procedure is best explained with an example. Figure 2 is the block diagram of a plant and measurement system for which a controller is to be designed. The blocks labeled G1, G2, etc. represent various known transfer functions within the plant. Blocks F1, F2, etc. represent sensor transfer functions and all blocks are assumed rational in the Laplace variable s.

The plant input is indicated by r_1 and r_2, disturbance input by d_1 and d_2, and measurement noise input by n_1, n_2, and n_3. Selected plant input and output are represented as elements of the vector R_p. With these definitions, the following equation set may be written:

$$R_p(1) = G1\ r_1 + d_1 - R_p(2) \tag{3.12a}$$

$$R_p(2) = G2\ R_p(1) \tag{3.12b}$$

$$R_p(3) = G3\ R_p(2) \tag{3.12c}$$

$$R_p(4) = F1\ R_p(2) + n_1 \tag{3.12d}$$

$$R_p(5) = F2\ R_p(7) + n_2 \tag{3.12e}$$

$$R_p(6) = G4\ r_2 + d_2 \tag{3.12f}$$

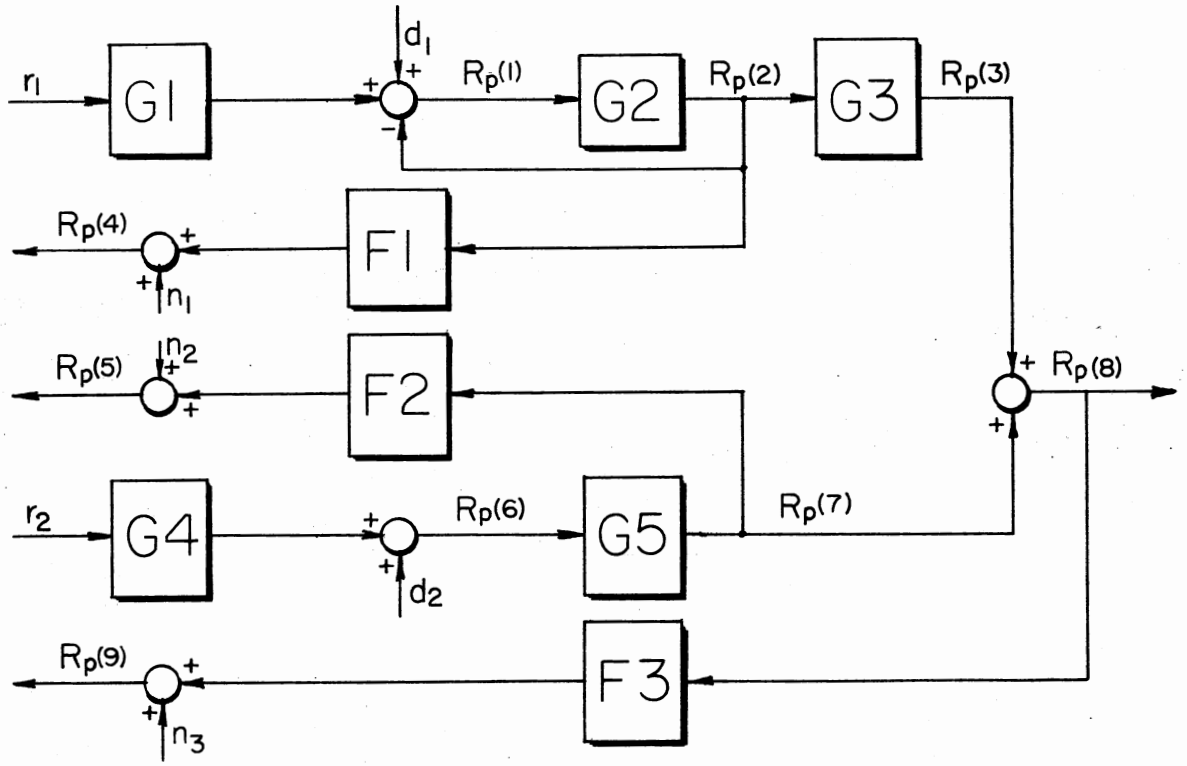

Figure 2. Example Illustrating the General Model Representation

$$R_p(7) = G5\ R_p(6) \tag{3.12g}$$

$$R_p(8) = R_p(3) + R_p(7) \tag{3.12h}$$

$$R_p(9) = F3\ R_p(8) + n_3 . \tag{3.12i}$$

This equation set represents a set of simultaneous equations which after rearranging can be written in matrix form as

$$A_p(s)\ R_p(s) = B_p(s) \tag{3.13}$$

with the matrix $A_p(s)$ defined as

$$A_p(s) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -G2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -G3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -F1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -F2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -G5 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -F3 & 1 \end{bmatrix} \tag{3.14}$$

and

$$B_p(s) = \begin{bmatrix} G1\ r_1 + d_1 \\ 0 \\ 0 \\ n_1 \\ n_2 \\ G4\ r_2 + d_2 \\ 0 \\ 0 \\ n_3 \end{bmatrix} \tag{3.15}$$

Equation (3.13) describes the plant and feedback measurement system completely. It should be noted here that the elements of $A_p(s)$ and $B_p(s)$ are rational transfer functions.

The next step is to determine $A_p^{-1}(s)$. For this example the inverse is computed as

$$A_p^{-1}(s) = \begin{bmatrix} \frac{1}{G2+1} & \frac{1}{G2+1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{G2+1} & \frac{1}{G2+1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{G3G2}{G2+1} & \frac{G3}{G2+1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{F1G2}{G2+1} & \frac{F1}{(G2+1)} & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & G5F2 & F2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & G5 & 1 & 0 & 0 \\ \frac{G3G2}{G2+1} & \frac{G3}{G2+1} & 1 & 0 & 0 & G5 & 1 & 1 & 0 \\ \frac{F3G3G2}{G2+1} & \frac{F3G3}{G2+1} & F3 & 0 & 0 & F3G5 & F3 & F3 & 1 \end{bmatrix} \tag{3.16}$$

To determine the plant matrix P(s), it is first necessary to designate the input to be used and the output to be controlled. For now, let the plant input be r_1 and r_2 and the output to be controlled be $R_p(8)$. Setting r_1 equal to one and the remaining input (r_2, d_1, d_2, n_1, n_2, and n_3) to zero, the $B_p(s)$ vector becomes

$$B_p'(s) = \begin{bmatrix} G1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} . \tag{3.17}$$

By solving

$$R_p(s) = A_p^{-1}(s)\ B_p'(s) \tag{3.18}$$

the transfer functions from the input r_1 to each of the plant output can be obtained. It is not necessary to find the entire $R_p(s)$ vector since only $R_p(8)$ is desired. Therefore, multiplying the eighth row of $A_p^{-1}(s)$ by the $B_p'(s)$ vector, the 1 x 2 plant matrix with only the first element determined is

$$P(s) = \left[\frac{G1G3G2}{G3+1} \quad \ldots\right] . \tag{3.19}$$

Now by setting r_2 to one and r_1 and the other input to zero and repeating the above process, element $P_{12}(s)$ of the plant matrix is obtained resulting in

$$P(s) = \left[\frac{G1G3G2}{G2+1} \quad G4G5\right] . \tag{3.20}$$

In a similar manner of setting each of the various input, disturbances, and noises in turn to one and using the appropriate elements of $R_p(s)$, the matrices $P_o(s)$, $F_o(s)$, $F(s)\ P(s)$, $F(s)\ P_o(s)$, $P_s(s)$, $L(s)$ and $L_o(s)$ can be obtained. Notice that the $F(s)$ matrix cannot be obtained directly by this representation. This is not of concern since only the

products $F(s)\ P(s)$ and $F(s)\ P_o(s)$ are actually needed in the synthesis calculation.

Several comments are in order at this point. First, the procedure requires the inversion of the rational polynomial matrix, $A_p(s)$. While this may seem somewhat complicated, it should be noted that the matrix is generally sparse and there exists a few efficient methods for performing this inversion (for example, REDUCE 2 [35]). Also, by careful selection of the output and input, a number of different plant input-output configurations can be utilized by the controller design program with a single inversion of $A_p(s)$. Considering Figure 2 again, it may be desired to design a controller for the single-loop plant which has r_1 as its input and $R_p(2)$ as its output. Using the procedure outlined previously and the same $A_p^{-1}(s)$ matrix, the plant is easily obtained as

$$P(s) = \left[\frac{G1G2}{G2+1}\right]. \tag{3.21}$$

Once implemented in an efficient computer program, this generalized model representation allows many designs to be investigated with minimal user effort.

A second comment is that methods similar to this have been used in other frequency domain control system analysis programs [57, 58]; however, its use in a synthesis program as described herein is new. If only a state-space representation of the plant is available, it can easily be related to the transfer function form of Figure 2 [59]. In fact, if a state-space representation of the plant is available, then a very general $A_p^{-1}(s)$ matrix is obtained, making it possible to consider all plant input-output configurations.

To illustrate the state-space approach, consider the familiar time-invariant state-space equation set:

$$\dot{x}(t) = A_s x(t) + B_s u(t) + D_s w(t) \tag{3.22}$$

$$y(t) = C_s x(t) + E_s z(t) \tag{3.23}$$

where $x(t)$ is the state vector, $u(t)$ is the input vector, $y(t)$ is the output vector, $w(t)$ is the disturbance vector, and $z(t)$ is the measurement noise vector. After taking the Laplace transform of Equations (3.22) and (3.23), they can be written as

$$x(s) = (sI - A_s)^{-1} (B_s u(s) + D_s w(s)) \tag{3.24}$$

and

$$y(s) = C(sI - A_s)^{-1} (B_s u(s) + D_s w(s)) + E_s z(s). \tag{3.25}$$

These equations can now be related to Equation (3.13) by letting

$$R_p(s) = y(s) \tag{3.26}$$

$$A_p(s) = (sI - A_s) \tag{3.27}$$

and

$$B_p(s) = (B_s u(s) + D_s w(s)). \tag{3.28}$$

The remaining calculations are then based on the equation

$$R_p(s) = C\, A_p(s)^{-1}\, B_p(s) + E_s z(s) \tag{3.29}$$

and the necessary transfer functions are obtained by alternatively setting the various input, disturbances, and noises to one and performing the multiplications and additions as before. The only difference is the presence of the additional vector, $E_s z(s)$, representing the measurement noise process.

As a final comment, the example of Figure 2 is trivial in that the required transfer functions can easily be computed manually. Chapter VI contains a more complicated example and demonstrates the effectiveness of this model representation theory. Although no examples are provided showing the use of a disturbance feedforward system, its inclusion in the model representation is straightforward.

CHAPTER IV

DIGITAL COMPUTATION AND RATIONAL POLYNOMIALS

During the course of this research, two different schemes for representing rational polynomial matrices and for performing the related arithmetic within a computer program were investigated. These investigations were carried out with the knowledge that the results would subsequently be applied in the development of a computer program for controller synthesis. Since the synthesis program implements the design theory described in Appendix B, the resulting scheme had to accommodate an algorithm for matrix spectral factorization, canonical decomposition of polynomial matrices, and partial fraction expansion of rational polynomials.

The schemes investigated are classified as the exact method and the floating-point method. The remainder of this chapter describes each of the methods separately along with their advantages and disadvantages. The final section of this chapter compares each of the methods, and shows which method was chosen as the best for the overall synthesis program. Appendix B defines the logical structure of the prototype program developed by this study. The program allowed each of the various representation and arithmetic schemes presented here to be easily tested within the general framework of the overall synthesis program.

Exact Methods

Rational Arithmetic

The use of rational arithmetic provides a means of performing exact

computations within a digital computer program. The basic concepts are well known and can be found in many texts (see References [36] and [37]).

Rational arithmetic as applied to polynomials requires that each of the coefficients of the polynomial be represented as a rational fraction. Consider a general nth-order polynomial

$$p(s) = a_o + a_1 s + a_2 s^2 + a_3 s^3 + \ldots + a_n s^n. \qquad (4.1)$$

To use rational arithmetic each coefficient must be represented as

$$a_i = \frac{q_i}{r_i}; \qquad i = 0, 1, 2, 3, \ldots, n \qquad (4.2)$$

where each q_i and r_i is an integer.

Rational fraction representation requires two integer numbers be stored in a computer program for each coefficient of each polynomial. The number of digits in each of these integer numbers will easily exceed the normal integer wordsize of current computers. For example, an IBM/370 can, in a single integer word, accurately represent at most nine digits. The use of only nine digit integers by the synthesis program would allow only the most trivial of problems to be solved. To illustrate, consider a 7th order polynomial whose roots are of magnitude greater than 100. The low order coefficient of the polynomial has a magnitude of approximately 100^7 and requires at least 15 digits to represent it accurately.

The wordsize limitation can be overcome by using several computer words to represent a single integer number. The arithmetic must then be performed by software since the normal machine arithmetic on most computers operates only the prescribed machine wordsize.

Polynomial arithmetic is done in the usual manner except addition,

subtraction, multiplication, and division of individual coefficients must take into account their fractional representation. The addition of two coefficients represented as in Equation (4.2) actually requires three multiplies, an addition, and a reduction of the resulting fraction to its lowest form. Reduction of a rational fraction to its lowest form means dividing out of the numerator and denominator, their greatest common divisor (GCD). This prevents the number of digits in the coefficients from becoming larger than necessary. Multiplication, division, and subtraction of coefficients are performed in a similar way.

Rational arithmetic is highly desirable for use in the controller synthesis program, only for the reason that exact computation is possible. The exactness of the various computations comprising the synthesis procedure directly determines its success. There is, however, one important drawback to the use of rational arithmetic known as intermediate coefficient swell. When rational arithmetic was implemented to perform the inversion of the example system matrix ($A_p(s)$) of Chapter VI, the number of digits required to represent some intermediate coefficients grew to over 70. Coefficient growth results in greatly increased computer computation times and uses large amounts of memory.

A more subtle illustration of coefficient growth is provided by the spectral factorization of a polynomial. Consider the polynomial

$$p(s) = 2 - s^2 \tag{4.3}$$

which has $\sqrt{2}+s$ and $\sqrt{2}-s$ as its spectral factors. Any attempt to do the factorization of Equation (4.3) using rational arithmetic and any conventional factorization algorithm (for example, see Tuel [45]) will fail due to the irrational coefficient $\sqrt{2}$. The number of digits required to represent the irrational coefficient is infinite.

Alternative Number Systems

The use of alternative number systems has been proposed recently as a means of implementing exact arithmetic within a computer program. Two methods were investigated for use in the synthesis program. The first method was the use of a residue number system [37] and the second was the use of a finite segment p-adic number system [41].

The use of either of these methods requires the polynomials to be representable as in Equation (4.1) with rational fraction coefficients. The main advantage of using one of these number system is that computer memory requirements are reduced. Basic operations, addition, subtraction, multiplication, and division, however, must be done by software which increases the execution time of the program.

Unfortunately, the same problems which hinder the rational arithmetic described earlier, specifically coefficient growth during spectral factorization, are not eliminated by the use of these alternative number systems. In fact, additional problems are introduced, especially by the use of the p-adic representation. These additional problems lie in the conversion of numbers from their alternate representation back to a readable decimal representation. The conversion process is very time consuming, and the need for the synthesis program to output various intermediate data requires many repeated conversions.

REDUCE Programming System

REDUCE [35] offers a very powerful means with which to manipulate rational polynomials and rational polynomial matrices. It can perform symbolic calculations as well as exact numerical computation. REDUCE uses a high-level language similar to Pascal which makes programming

relatively easy. The ability to perform symbolic calculations is the greatest asset of REDUCE; however, computer execution is slow and large amounts of memory are required. Also, REDUCE is not available on many computer systems, and its implementation on some systems, such as the IBM 360 and 370 computers, is incomplete.

Since REDUCE uses, basically, the same rational arithmetic described earlier to do exact numerical calculations, it suffers from the same coefficient growth problem. This problem is easily avoided by the use of symbols for the polynomial coefficients. REDUCE may then perform a desired series of calculations and return the answer in terms of the original symbols. To obtain actual numerical values for a solution, REDUCE can be made to write its answer in the form of a FORTRAN subprogram which, when supplied with the numerical values for the original symbols, can be called to calculate numerical values for the solutions.

REDUCE is well suited as a preprocessor type system for the controller synthesis program. It can be used to solve the generalized model representation Equation (3.13) symbolically and write a FORTRAN subroutine which is called by the synthesis program to obtain the various transfer function matrices required (i.e., $P(s)$, $P_o(s)$, etc.). A REDUCE program was set up to do this for the pointing and tracking system example of Chapter VI (see Figure 9) and it proved to work very well. Total execution time was approximately two minutes on an IBM 370/168; however, since the program must only be executed once for a particular plant, the execution time may be acceptable.

The use of REDUCE to do the entire controller synthesis computation was also investigated. The major difficulties encountered were the lack of algorithms to do the spectral factorization, partial fraction

expansion, and the coprime decompositions of Equation (A.3). The development of these algorithms will require the addition of some basic capabilities to REDUCE such as polynomial synthetic division and polynomial factoring. If such capabilities become available for REDUCE and suitable algorithms develop, it may be possible for REDUCE to solve the entire controller design problem symbolically, giving the resulting controller in terms of the original plant symbols. This would be a very ideal solution due to the fact that when any plant parameter's value is changed, the controller is immediately known. Also, if the various weightings of the design process were symbolic, the controller would also contain these symbols and trade-off studies for various weighting values could be done very easily.

Although REDUCE is very powerful, its use for the controller synthesis process is, at present, limited to the role of a preprocessor for plant determination. As its capabilities are expanded and it becomes more machine portable, it most likely will become a major tool for frequency-domain controller design.

Floating-Point Methods

The use of finite precision floating-point or real arithmetic is most advantageous from the standpoint of availability of algorithms such as the spectral factorization algorithm of Tuel [45]. Real arithmetic methods are also relatively easy to implement in computer programs with well-known languages like FORTRAN. However, computations involving polynomials with real coefficients suffer a multitude of numerical problems.

The numerical problems became evident when direct implementation of floating-point arithmetic was attempted in the controller synthesis

program. Numerical inaccuracies in the results of one computation were propagated and amplified by subsequent computations. The numerical errors would eventually become so large that further computation became impossible and the program would terminate before any solution was found.

In the remainder of this section, the use of floating-point arithmetic for polynomial operations is discussed. Some problems are identified and means to overcome the problems are outlined.

Direct Polynomial Representation

Direct polynomial representation means polynomials are represented in a computer program by storing the n + 1 coefficients of an n-order polynomial as an ordered set of real numbers. Later in this section an alternative representation is discussed in which the roots of polynomials are stored along with a gain value.

In order to analyze the numerical problems associated with floating-point polynomial computations, it is first necessary to examine the nature of the rational polynomials which arise from linear systems. A rational transfer function is represented by

$$K_g \frac{\prod_{i=1}^{m} \left(\frac{s}{\alpha_i} + 1\right)}{\prod_{j=1}^{n} \left(\frac{s}{\beta_j} + 1\right)} \tag{4.4}$$

where K_g is a constant gain and α_i and β_j represent the zeros and poles of the system. The α_i and β_j can be real or complex and if complex they occur as conjugate pairs.

Both the numerator and denominator, written as polynomials, become

$$\frac{a_o + a_1 s + a_2 s^2 + a_3 s^3 + \ldots + a_m s^m}{b_o + b_1 s + b_2 s^2 + b_3 s^3 + \ldots + b_n s^n} \tag{4.5}$$

where

$$a_o = K_g \tag{4.6a}$$

$$a_1 = K_g \sum_{\ell=1}^{m} \frac{1}{\alpha_\ell} \tag{4.6b}$$

$$a_2 = K_g \sum_{\ell=1}^{m-1} \sum_{k=\ell+1}^{m} \frac{1}{\alpha_\ell \alpha_k} \tag{4.6c}$$

$$a_3 = K_g \sum_{\ell=1}^{m-2} \sum_{k=\ell+1}^{m=1} \sum_{j=k+1}^{m} \frac{1}{\alpha_\ell \alpha_k \alpha_j} \tag{4.6d}$$

$$a_n = \sum_{\ell=1}^{m-m+1} \sum_{k=\ell+1}^{m-m+2} \sum_{j=k+1}^{m-m+3} \cdots \sum_{i=m}^{m} \frac{1}{\alpha_\ell \alpha_k \alpha_j \cdots \alpha_i} \tag{4.6e}$$

and the b_o, b_1, $b_2 \ldots b_n$ are defined similarly without the K_g term. If

$$|\alpha_i| > 1 \qquad \text{for } i = 1, 2, 3, \ldots, m \tag{4.7}$$

then the magnitude of coefficient a_o can be large compared to coefficient a_m. Consider, for example, the poles β_j each having magnitudes of the order of 10^3 (which is not unreasonable for a very large class of linear systems). The magnitude of the low-order coefficient of the denominator is 1 while the magnitude of the high-order coefficient is 10^{-3n}. As the order of the polynomial increases, the difference in magnitude between the high-order and low-order coefficient increases. Normalization of the rational polynomial of Equation (4.5) using the high-order coefficient of either the numerator or denominator will not reduce this difference. This large magnitude difference is one of the major difficulties in the use of finite-precision, floating-point arithmetic.

A second major difficulty arises from the finite-precision which computers use for floating-point computation. Precision affects two important polynomial calculations directly, synthetic division and the calculation of the GCD between two polynomials. To illustrate this effect, consider the low-order coefficient a_o from the polynomial

$$p_1(s) = \prod_{i=1}^{n} (s + \alpha_i) \tag{4.8}$$

which is defined as

$$a_o = \alpha_1\alpha_2\alpha_3 \cdots \alpha_n . \tag{4.9}$$

Assume, for simplicity, that each α_i is real and is accurate to two significant digits and that the precision of the machine arithmetic is assumed to be four significant digits. Calculation of the a_o coefficient is done by successive multiplications with the results of each multiplication being chopped to four digits. (Chopping is the worst case applicable to finite-precision arithmetic. This is the technique used in a majority of computers, although some employ a rounding scheme [60].) If no noise is introduced by the multiply, the first product $\alpha_1\alpha_2$ will have no error. The second product $\alpha_1\alpha_2\alpha_3$ will be chopped to four digits; hence, the result of the finite-precision multiply becomes

$$\rho_2 = \alpha_1\alpha_2\alpha_3 - \varepsilon_2 \tag{4.10}$$

where ε_2 is the error introduced into the result by chopping. Proceeding with the remaining products the final product becomes

$$a_o \approx \rho_n = (\cdots(((\alpha_1\alpha_2\alpha_3 - \varepsilon_2)\alpha_4 - \varepsilon_3)\alpha_5 - \varepsilon_4)\cdots)\alpha_n - \varepsilon_{n-1} \tag{4.11}$$

and the total error in ρ_n is

$$E = \varepsilon_2\alpha_4\alpha_5 \cdots \alpha_n + \varepsilon_3\alpha_5\alpha_6 \cdots \alpha_n + \cdots \varepsilon_{n-1}. \tag{4.12}$$

The remaining coefficients of the polynomial in Equation (4.8) involve the sums of products, and in addition to chopping error introduced by multiplication, additional error is added due to the addition process.

Error due to chopping is introduced into the coefficients of the polynomials as they are computed using Equation (4.6). For some applications, this error may not represent a problem. However, for optimal controller synthesis it is a very significant problem, since the success of the computations depends on the ability of arithmetic to factor a high-order polynomial into its lower order factors.

Returning now to the polynomial formed from Equation (4.8), suppose that it is desired to divide out the polynomial

$$p_2(s) = \prod_{i=1}^{n-1} (s + \alpha_i) \tag{4.13}$$

leaving

$$p_3(s) = s + \alpha_n . \tag{4.14}$$

The effect of precision error in multiplication can be demonstrated by working with only the low-order coefficients of the participating polynomials. Previously, the low-order coefficient of Equation (4.8) was determined to be ρ_n from Equation (4.11). The low-order coefficient of Equation (4.13) can be determined in a similar manner as

$$\rho'_{n-1} = (\cdots((\alpha_2\alpha_3\alpha_4 - \varepsilon'_2)\alpha_5 + \varepsilon'_3)\cdots)\alpha_n - \varepsilon'_{n-1} . \tag{4.15}$$

Dividing ρ_n by ρ'_{n-1} and again chopping the result to four significant digits, α_n for Equation (4.14) becomes

$$\bar{\alpha}_n = \frac{\rho_n}{\rho'_{n-1}} = \frac{\alpha_1\alpha_2\alpha_3\cdots\alpha_n - (\varepsilon_2\alpha_4\alpha_5\cdots\alpha_n + \varepsilon_3\alpha_5\alpha_6\cdots\alpha_n + \varepsilon_{n-1})}{\alpha_2\alpha_3\cdots\alpha_n - (\varepsilon'_2\alpha_5\alpha_6\cdots\alpha_n + \varepsilon'_3\alpha_6\alpha_7\cdots\alpha_n + \varepsilon'_{n-1})} + \varepsilon_d \tag{4.16}$$

where ε_d is introduced by chopping after the division operation.

To illustrate quantitatively the size of this error, a numerical example can be used. Let $n = 5$, and each α_i be real and contain two significant digits and their numerical values be given as (which the machine carries as 4 digits)

$$\alpha_1 = 11.00 \tag{4.17a}$$

$$\alpha_2 = 22.00 \tag{4.17b}$$

$$\alpha_3 = 68.00 \tag{4.17c}$$

$$\alpha_4 = 35.00 \tag{4.17d}$$

$$\alpha_5 = 4.20\,. \tag{4.17e}$$

The low-order coefficient of Equation (4.8) becomes (with 4 digit arithmetic)

$$\rho_5 = 2417.0 \times 10^3 \tag{4.18}$$

which has a total error equal to 2032. The low-order coefficient of Equation (4.13) is

$$\rho'_4 = 5757.0 \times 10^2 \tag{4.19}$$

with a total error equal to 56. Carrying out the division of Equation (4.16) and chopping the result to 4 digits $\bar{\alpha}_5$ is obtained as

$$\bar{\alpha}_5 = 4.198 \tag{4.20}$$

which is in error by 0.002 or 0.05 percent.

This error may seem somewhat small, so the next example illustrates the effect of these errors, coupled with the errors introduced by

addition and subtraction, as they are accumulated during the entire polynomial division process. The polynomial in Equation (4.8) calculated with four digit arithmetic using the values given by Equation (4.17) is

$$p_1(s) = 2.417 \times 10^6 + 1.009 \times 10^6 s + 1.286 \times 10^5 s^2 + 6.592 \times 10^3 s^3 + 1.402 \times 10^2 s^4 + s^5 \quad (4.21)$$

and the polynomial given by Equation (4.13) becomes

$$p_2(s) = 5.757 \times 10^5 + 1.034 \times 10^5 s + 6.021 \times 10^3 s^2 + 1.36 \times 10^2 s^3 + s^4 . \quad (4.22)$$

Performing the division in the following manner (see Equation (4.23) below) the result is

$$\frac{P_1(s)}{P_2(s)} = 4.2 + s - \frac{900s + 80s^2 + 0.2s^3}{P_2(s)} \quad (4.24)$$

The quotient of Equation (4.24) is in error due to the presence of the nonzero remainder term, even though the term 4.2 + s is correct. This example illustrates how the errors of multiplication are amplified by addition. The effects of finite-precision error become more pronounced as the order of the polynomials increases.

In general, the result of the division will not yield as exact an answer as in this example. If p_1 from Equation (4.8) is divided by the first-order polynomial p_3 from Equation (4.14), using again the same numerical values, the result becomes

$$\frac{P_1(s)}{P_3(s)} = 5752 + 1033s + 6020s^2 + 136s^3 + s^4 + \frac{2000}{P_3(s)} . \quad (4.25)$$

When compared with Equation (4.22), the error becomes evident. While

$$
\begin{array}{r}
4.2 + s \\
p_2(s)\,\big/\;\overline{2.417\times10^{6} + 1.009\times10^{6}s + 1.286\times10^{5}s^{2} + 6.592\times10^{3}s^{3} + 1.402\times10^{2}s^{4} + s^{5}} \\
\underline{-(5.575\times10^{6}s + 1.034\times10^{5}s^{2} + 6.021\times10^{3}s^{3} + 1.360\times10^{2}s^{4} + s^{5}} \\
2.417\times10^{6} + 4.333\times10^{5}s + 2.520\times10^{4}s^{2} + 5.710\times10^{2}s^{3} + 4.2\times10^{0}s^{4} \\
\underline{-(2.417\times10^{6} + 4.342\times10^{5}s + 2.528\times10^{4}s^{2} + 5.712\times10^{2}s^{3} + 4.2\times10^{0}s^{4}} \\
- 9.000\times10^{2}s - 8.000\times10^{1}s^{2} - 2.000\times10^{-1}s^{3}
\end{array}
\tag{4.23}
$$

these results by themselves may be tolerable, the controller synthesis computation requires many of these types of operations. As error-contaminated results of one calculation are used in subsequent calculations, the errors propagate and become amplified. When the operations are done using polynomial matrices, finite-precision arithmetic results in even greater errors.

The third major problem is caused by the manner in which the fractional part of a number is represented with floating-point arithmetic. Only radix fractions can be represented exactly (up to the number of digits of precision) using floating-point arithmetic.[1] All other fractions must be approximated. This need to approximate certain fractions causes additional error in the results of the multiplication and addition operations. This additional error is called noise and in general will reduce the number of digits of precision actually available on a certain machine [60].

Root Representation

In light of the problems associated with finite-precision, floating-point arithmetic, an alternative method of representing polynomials was investigated. Instead of storing the coefficients of the polynomials directly, the actual roots (complex and real) are stored along with a gain value.

There are several advantages associated with the root representation method. First, the magnitudes of roots are smaller than the magnitudes

---

[1] A radix fraction is a fraction which can be expressed as some multiple of $1/p^{\sigma}$, where p is the base of the machine arithmetic and σ is a positive integer.

of coefficients. Second, for most engineering problems, roots usually need only to be accurate to two or three significant digits requiring less precision than coefficients. Third, the multiplication of polynomials is simple and introduces no error since no actual arithmetic is required. Finally, the GCD between polynomials is easily determined by comparison of their roots.

The main disadvantage of the root representation method are polynomial addition and division. Addition of polynomials in the root representation requires the evaluation of an equation of the form

$$K_1(s+\alpha_1)(s+\alpha_2)\ldots(s+\alpha_j) + K_2(s+\beta_1)(s+\beta_2)\ldots(s+\beta_k)$$
$$= K_3(s+\lambda_1)(s+\lambda_2)\ldots(s+\lambda_\ell)\,. \qquad (4.26)$$

Here the K_1 and K_2 are known gains of the two polynomials to be added and the α_i and β_i are their known roots. Addition requires the determination of the gain K_3 and the unknown roots designated λ_i. The only method which could be devised to solve this equation required the use of a factoring routine.

Two options were investigated to do the factoring. The first option was to use an analytic root finding routine to determine the roots of the function

$$f(s) = K_1(s+\alpha_1)(s+\alpha_2)\ldots(s+\alpha_j) + K_2(s+\beta_1)(s+\beta_2)\ldots(s+\beta_k) \qquad (4.27)$$

directly. The other option was to first compute the coefficients of each polynomial term, add the two terms, then use a polynomial factoring routine to determine the desired roots.

The second option nullified the benefits of the root representation method which was developed to avoid the problems associated with

representing polynomial coefficients. The first option was difficult because the number of roots, ℓ, in the sum was sometimes difficult to determine. Both methods were very time consuming since polynomial addition is required often during the controller synthesis.

The algorithms employed to do the factorization produced results of insufficient accuracy and the controller computation deteriorated faster than with the direct polynomial representation method. Polynomial division suffers the same ill effects that addition does because it requires a series of polynomial subtracts. Until a suitable method to do polynomial addition and division is found, the root representation method cannot be used. Once the addition and division problems are solved, new algorithms for spectral factorization and partial fraction decomposition will still have to be developed.

Summary

The various techniques outlined in this chapter were all investigated during this research in an effort to determine which could be applied to the controller synthesis program. It was determined that the exact methods were extremely reliable; however, the real floating-point methods yielded comparable results when the problems mentioned in the previous section were accounted for correctly. The exact methods were found to be usable for the preliminary steps in the synthesis process, such as the inversion of the $A_p(s)$ matrix of the generalized model representation. Of the exact methods, the REDUCE programming system proved the most valuable. The other exact methods require complex programming to perform the arithmetic and large amounts of memory to carry the large number of digits in the coefficients which accumulated during computations.

Once suitable methods were found for avoiding the problems of floating-point arithmetic (i.e., scaling, increased precision, reduced noise) it was discovered that for many small scale control systems, floating-point arithmetic could be used to solve the entire synthesis problem. It was also possible to solve the generalized model representation, Equation (3.18), using floating-point, for the smaller control system. The next chapter describes the various algorithms employed in the controller synthesis.

During the investigation of the numerical problems associated with the use of polynomial arithmetic in the controller synthesis program, some major computation trouble areas were identified. First, accurate calculation of the GCD of polynomials is of utmost importance to the success of the overall synthesis program. The GCD calculation using floating-point arithmetic and direct polynomial representation, is the most numerically difficult computation of all the basic polynomial operations. Second, the accuracy to which the more sophisticated computations, such as spectral factorization, can be performed directly affects the success of the overall synthesis. Finally, once any significant error contaminates the polynomial coefficients during the controller synthesis calculation, it is very rapidly propagated and amplified and the ability of the program to determine a solution controller is greatly impaired. The error propagation problem becomes worse when controllers for multivariable systems are being computed by the program, due to the fact that the computations involve rational polynomial matrices instead of simple rational polynomials.

CHAPTER V

DIGITAL COMPUTER IMPLEMENTATION OF THE OPTIMAL CONTROLLER DESIGN THEORY

Direct polynomial representation and finite-precision, floating-point arithmetic were selected as best for the controller synthesis program. This selection was made because: (1) the available algorithms for spectral factorization utilized floating-point arithmetic; (2) methods were devised as a result of this research to avoid the major problems associated with finite-precision, floating-point arithmetic; (3) the partial fraction expansion algorithm developed by this study required floating-point arithmetic. Also, the use of direct polynomial representation and floating-point arithmetic led to a computer program which was less complex and more efficient than the program which would have resulted had any of the exact theory been implemented. Direct polynomial representation was chosen instead of root representation due to the lack of an accurate polynomial addition routine for the latter.

This chapter describes the algorithms used in the prototype synthesis program and the details related to floating-point arithmetic. The information presented here is one of the major contributions of this work.

Basic Operations

Polynomial Arithmetic

In order to avoid the problems associated with the use of finite-

precision floating-point arithmetic, three important features were implemented in the synthesis program. First, the precision of the entire program was increased to the maximum allowed by IBM/370 FORTRAN. This was accomplished with the aid of the extended precision option of the IBM FORTRAN Level H compiler and resulted in reliable precision to 34 significant digits [61]. Second, scaling was implemented to avoid large magnitude differences between the coefficients of various polynomials. Finally, an algorithm was developed and implemented for the addition and subtraction of individual numbers which employed rounding to prevent propagation of calculation noise.

The first feature increased the precision of every polynomial coefficient to 34 significant digits and also increased the precision of machine level arithmetic to 34 significant digits. The second feature, scaling, was used to reduce magnitude difference between the coefficients of a polynomial. Scaling, sometimes called frequency scaling, has the effect of dividing all the roots of a polynomial by a constant such that their magnitude approaches one. Scaling can be applied to polynomials by

$$s' = rs \tag{5.1}$$

where r is a scaling constant. The value of r is chosen to obtain minimal range in the coefficient magnitudes. As an example, the polynomial of Equation (4.21) can be scaled by letting

$$s' = 10s. \tag{5.2}$$

Equation (4.21) now becomes

$$p_1(s') = 2.417\times10^6 + 1.009\times10^7 s + 1.286\times10^7 s^2 + 6.592\times10^6 s^3 + 1.402\times10^6 s^4 + 1.0\times10^5 s^5 \tag{5.3}$$

and the magnitude difference between coefficients is greatly reduced. This technique works well with rational polynomials since normalization of either the numerator or denominator polynomial will further reduce the magnitude of all of the coefficients. Properly adjusted coefficient magnitudes help prevent exponent overflow and underflow errors during subsequent computations.

Unfortunately, scaling does not help the precision problem, and will usually make it worse. As the rational polynomials are scaled and normalized, the magnitudes of the coefficients tend to become less than unity, which results in additional calculation noise and therefore reduced precision. As the order of the polynomials increases, the precision required for accurate representation may exceed the hardware capability of the computer. The only solution to insufficient precision is an increase in precision. Hence, large optimal control problems cannot be solved unless the computer floating-point precision is increased or software type multiple-precision arithmetic [62] is employed.

The basic polynomial operations were implemented by algorithms which are similar to those of Reference [54] except for the GCD operation. The GCD algorithm was a version of the one proposed by Matthew [57]. The third feature was included in all of the polynomial arithmetic routines. This feature proved to be a major contribution to the success of the overall synthesis program and it is discussed here in detail.

Whenever the basic polynomial routines require addition or subtraction of two numbers, a special routine is called to perform and monitor the required operation. Even though every number in the program is stored as an extended precision word (34 significant digits), the algorithm developed considers only part of the total number of digits valid during

addition or subtraction. The remaining digits are considered to be calculation noise. The addition and subtraction routine first adds or subtracts the numbers normally, then tests the results. The test is performed in such a way that the effect is the same as if the numbers were first rounded to some preset number of valid digits before they are added or subtracted. If the test determines the result should be zero, the routine sets the result to identically zero.

The number of digits to be considered significant in addition or subtraction is set at the start of program execution and can be adjusted at various points during the controller computation. This value is initially set to a value which prevents calculation noise from being propagated and provides for the greatest number of significant digits for all polynomial coefficients. A value between 26 and 28 was found to be the maximum which could be used with FORTRAN extended precision arithmetic. The ability to alter the number of digits at various points within the synthesis program allowed the precision of the numbers to be progressively reduced to account for errors introduced by the various computation steps of the program.

For purposes of this research, addition and subtraction were also monitored to determine whether the magnitudes of the two numbers were compatible. For example, if the number of valid digits is set to 30, numbers whose magnitudes differ by more than 10^{29} were considered incompatible. The addition and subtraction routine merely reported the occurrence of this condition. This monitoring was done to verify correct operation of the synthesis program and did not always indicate an error condition.

Both the above feature and the increased machine precision contributed

significantly to improved performance of all the basic polynomial operations. Scaling, in general, does not have any significant effect on arithmetic operations except to increase calculation noise slightly and reduce the possibility of exponent underflow or overflow. The operations of polynomial division and especially the GCD calculation were most sensitive to precision.

GCD Calculation

Accurate calculation of the GCD between two polynomials is one of the most critical calculations of the entire synthesis process. It is used to keep the numerator and denominator of all rational polynomials prime. It is critical because the validity of the synthesis theory requires that the rational polynomial elements of certain matrices have no common factors between numerator/denominator pairs. The matrices defined by Equations (A.12) and (A.13) in Appendix A are examples of these critical matrices.

Since the GCD calculation is critical, the algorithm which was employed by the synthesis program is outlined here. The algorithm is given the coefficients of two polynomials for which the GCD is to be computed. The following steps are then performed:

1. The zero valued roots of both polynomials are removed by inspecting the low-order coefficients. The number of zero roots common to both polynomials is retained.

2. Each polynomial is normalized such that its low-order coefficient is unity.

3. The polynomial of lowest order is subtracted from the other.

The subtraction is done by calling the polynomial subtraction subroutine which employs the special addition and subtraction routine.

4. The results of the subtraction is then checked for the zero polynomial. If the result is the zero polynomial, either of the two polynomials is the correct GCD. The GCD is made monic by high-order normalization and the number of zero roots retained in step 1 are inserted. If the result is not the zero polynomial, the calculation proceeds.

5. The zero root is removed from the polynomial obtained in step 3.

6. At this point there are three polynomials. The polynomial having the highest order is discarded and the calculation repeats from step 2 using the two remaining polynomials.

The theory which supports this algorithm is outlined in Reference [57].

The following example demonstrates the effect of noise in the above algorithm and the effect of precision. The two polynomials are those used for the example in Reference [57] and are defined as

$$\begin{aligned} p_1(s) &= (s^3 + 3s^2 + 9s - 4)(s^5 + s^4 - 3s^2 + 2s + 2) \\ &= s^8 + 4s^7 + 12s^6 + 2s^5 - 11s^4 - 19s^3 + 36s^2 \\ &\quad + 10s - 8 \end{aligned} \tag{5.4}$$

and

$$\begin{aligned} p_2(s) &= (s^3 + 3s^2 + 9s - 4)(s^2 - 10s + 5) \\ &= s^5 - 7s^4 - 16s^3 - 79s^2 + 85s - 20. \end{aligned} \tag{5.5}$$

The calculation of the GCD of $p_1(s)$ and $p_2(s)$ was carried out using IBM double precision (16 digits) arithmetic and the algorithm described above. Notice that each coefficient has two significant digits. By setting a special variable (named NDIG) to the value of 2, the polynomial

subtraction routine is instructed to consider only two digits in each coefficient of the polynomials being subtracted to be significant. With NDIG set equal to 2, the algorithm calculated the GCD of $p_1(s)$ and $p_2(s)$ to be 1.0 (no common factors). When NDIG was set to values between 3 and 12 inclusive, the GCD was obtained to be (with all actual 16 digits)

$$\begin{aligned} GCD(p_1(s),\ p_2(s)) = {} & 1.000000000000000\ s^3 \\ & + 3.000000000000006\ s^2 \\ & + 9.000000000000019\ s \\ & - 3.999999999999942\ . \end{aligned} \tag{5.6}$$

When NDIG was set to any number greater than 12, the GCD was again calculated to be 1.0. This result implies that at least three significant digits are required to compute the correct result and that noise prevents the result from being obtained with more than 12 significant digits.

Rational Polynomial Matrix Arithmetic

Rational polynomial matrices can be represented in one of two ways. Let

$$P(s) = \begin{bmatrix} p_{11}(s) & p_{12}(s) & \cdots & p_{1\ell}(s) \\ p_{21}(s) & p_{22}(s) & \cdots & p_{2\ell}(s) \\ \vdots & \vdots & & \vdots \\ p_{k1}(s) & p_{k2}(s) & \cdots & p_{k\ell}(s) \end{bmatrix} \tag{5.7}$$

where each $p_{k\ell}(s)$ is a rational polynomial with coefficients of the form of Equation (4.2). If the least common multiple (LCM) of all the denominator polynomials of P(s) is calculated and labeled g(s), then by

multiplying each element of P(s) by g(s), the matrix can also be represented in the form

$$P(s) = \frac{1}{g(s)} \begin{bmatrix} p'_{11}(s) & p'_{12}(s) & \cdots & p'_{1\ell}(s) \\ p'_{21}(s) & p'_{22}(s) & \cdots & p'_{2\ell}(s) \\ \vdots & \vdots & & \vdots \\ p'_{k1}(s) & p'_{k2}(s) & \cdots & p'_{k\ell}(s) \end{bmatrix} \tag{5.8}$$

where each $p'_{k\ell}(s)$ is polynomial.

Through experience during the course of this research, the representation of Equation (5.8) was found to be the most suitable. The reason this representation was selected is that this form is generally required by the special algorithms (such as canonical decomposition, matrix inversion, and spectral factorization) of the synthesis program. The matrix form of Equation (5.8) is easier to manipulate within a computer program, and matrix arithmetic requires fewer polynomial operations than the alternative representation of Equation (5.7). The disadvantage of the representation of Equation (5.8) is that polynomials $p'_{k\ell}(s)$ and g(s) will generally be of higher order and the magnitude of their coefficients will be larger than the polynomials in Equation (5.7).

Special Matrix Operations

Rational Polynomial Matrix Inversion

Given a polynomial matrix $P'_1(s)$ and its scalar polynomial divisor, the inverse of matrix $P_1(s)$, defined as

$$P_1(s) = \frac{1}{g(s)} [P_1'(s)], \tag{5.9}$$

is calculated in the following manner. By using a procedure similar to the one described by Gantmacher [63], $P_1'(s)$ is reduced to a diagonal matrix by a series of row and column operations. The result of all row operations is represented as an elementary matrix U(s) and the column operations as an elementary matrix V(s). The operation is represented by the equation

$$U(s)\ P_1'(s)\ V(s) = P_{diag}(s). \tag{5.10}$$

The inverse of $P_1(s)$ is defined as

$$P_1^{-1}(s) = g(s)\ [P_1'(s)]^{-1} \tag{5.11}$$

Using Equation (5.5), the inverse of $P_1'(s)$ is

$$[P_1'(s)]^{-1} = V(s)\ P_{diag}^{-1}(s)\ U(s). \tag{5.12}$$

Once U(s), V(s), and $P_{diag}(s)$ are obtained, it is a simple matter of inverting $P_{diag}(s)$ and carrying out the required multiplications. The final result is then put into the required rational polynomial matrix form.

The main problem experienced with this algorithm was the very high-order polynomials during the diagonalization process. When the order of these polynomials increased to the point where their coefficients could no longer be accurately represented, the algorithm failed.

Coprime Decomposition of Rational Polynomial Matrices

A coprime decomposition algorithm is required to do the operations

shown as Equations (A.3) and (A.4) in Appendix A. The algorithm developed follows the theory of Jabr [64] which is outlined here.

By a suitable set of elementary row and column operations, the rational polynomial matrix F(s) P(s) can be reduced to its Smith-McMillian form [44] and becomes[1]

$$F(S)\ P(s) = U(s)(\Omega_c(s) \oplus 0_{n-k,m-k})\ V(s) \tag{5.13}$$

where

$$\Omega_c(s) = \text{diag}\ \frac{n_1(s)}{d_1(s)}\ ,\ \frac{n_2(s)}{d_2(s)}\ ,\ \ldots,\ \frac{n_k(s)}{d_k(s)} \tag{5.14}$$

The subscript k equals the normal rank of the n x m matrix F(s) P(s). Each numerator polynomial $n_i(s)$ is relatively prime to its denominator, $d_i(s)$; $n_i(s)$ divides $n_{i+1}(s)$ without remainder and $d_{i+1}(s)$ divides $d_i(s)$ without remainder.

This reduction is accomplished algorithmically using the method described by Gantmacher [63] for reduction of matrices to canonical form. The F(s) P(s) matrix is assumed to be in the form

$$F(s)\ P(s) = \frac{1}{g(s)}\ [P_1'(s)] \tag{5.15}$$

where $P_1'(s)$ is a polynomial matrix and g(s) is the LCM of all the denominator polynomials of F(s) P(s). $P_1'(s)$ is reduced to canonical form resulting in

$$P_1'(s) = U(s)(\Omega_c'(s) \oplus 0_{n-k,m-k})\ V(s) \tag{5.16}$$

where U(s) and V(s) are elementary polynomial matrices representing the

---

[1] The symbol $\oplus$ implies "direct matrix sum."

row and column operations used for the reduction. The matrix $\Omega_c'(s)$ is of the form

$$\Omega_c'(s) = \text{diag}[n_1'(s),\ n_2'(s),\ \ldots,\ n_k'(s)]\ . \tag{5.17}$$

Now by calculating the GCD between $n_1'(s)$ and $g(s)$ and dividing

$$n_i(s) = \frac{n_i'(s)}{\text{GCD}(n_1'(s),\ g(s))}\ ;\quad i = 1, 2, \ldots, k \tag{5.18}$$

and

$$d_i(s) = \frac{g(s)}{\text{GCD}(n_1'(s),\ g(s))}\ ;\quad i = 1, 2, \ldots, k \tag{5.19}$$

are obtained. Since each $n_i(s)$ is relatively prime to its mate $d_i(s)$, there exist two polynomials $p_i(s)$ and $q_i(s)$, $q_i(s) \neq 0$, such that

$$p_i(s)\ n_i(s) + q_i(s)\ d_i(s) = 1;\quad i = 1, 2, \ldots, k\ . \tag{5.20}$$

Each $p_i(s)$ and $q_i(s)$ are obtained algorithmically by solving k separate sets of simultaneous equations. A suitable equation solver is employed which includes an iterative solution improver for accuracy and can return the number of significant digits available in the solution.

Each set of simultaneous equations is set up in the following manner. Let the order of $n_i(s)$ be j and the order of $d_i(s)$ be ℓ and

$$n_i(s) = a_o^i + a_1^i s + a_2^i s^2 + \ldots + a_j^i s^j \tag{5.21}$$

and

$$d_i(s) = b_o^i + b_1^i s + b_2^i s^2 + \ldots + b_\ell^i s^\ell \tag{5.22}$$

Assume the order of $p_i(s)$ and $q_i(s)$ to be $\ell - 1$ and $j - 1$, respectively, and

$$p_i(s) = c_o^i + c_1^i s + c_2^i s^2 + \ldots + c_{-1}^i s^{-1} \tag{5.23}$$

and

$$q_i(s) = e_o^i + e_1^i s + e_2^i s^2 + \ldots + e_{j-1}^i s^{j-1} \ . \tag{5.24}$$

The matrix equation

$$\begin{bmatrix} \begin{vmatrix} a_o^i & 0 & \ldots & 0 \\ a_1^i & a_o^i & & \vdots \\ a_2^i & a_1^i & & \vdots \\ \vdots & \vdots & & \\ \vdots & a_2^i & & a_o^i \\ a_j^i & \vdots & & a_1^i \\ 0 & a_j^i & & a_2^i \\ \vdots & 0 & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & \vdots & & \end{vmatrix} & \begin{vmatrix} b_o^i & 0 & \ldots & 0 \\ b_1^i & b_o^i & & \vdots \\ b_2^i & b_1^i & & \vdots \\ \vdots & & & \\ \vdots & b_2^i & & b_o^i \\ b_\ell^i & \vdots & & b_1^i \\ 0 & b_\ell^i & & b_2^i \\ \vdots & 0 & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & \vdots & & \end{vmatrix} \end{bmatrix} \begin{bmatrix} c_o^i \\ c_1^i \\ c_2^i \\ \vdots \\ c_{\ell-1}^i \\ e_o^i \\ e_1^i \\ e_2^i \\ \\ e_{j-1}^i \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \\ 0 \\ \vdots \\ \\ \end{bmatrix}$$

$(\ell+j-2)\times(\ell-1)$ $\qquad$ $(\ell+j-2)\times(j-1)$

(5.25)

is the desired simultaneous equation set which must be solved to obtain $c_o^i, c_1^i, \ldots, c_{\ell-1}^i, e_o^i, e_1^i, e_2^i, \ldots, e_{j-1}^i$. Now defining

$$n = \text{diag}\,[n_1(s), n_2(s), \ldots, n_k(s)] \tag{5.26}$$

$$d = \text{diag}\,[d_1(s), d_2(s), \ldots, d_k(s)] \tag{5.27}$$

$$p = \text{diag}\,[p_1(s), p_2(s), \ldots, p_k(s)] \tag{5.28}$$

$$q = \text{diag}\,[q_1(s),\ q_2(s),\ \ldots,\ q_k(s)] \tag{5.29}$$

the desired coprime factors are obtained as

$$A(s) = (d \oplus I_{n-k})\ U(s)^{-1} \tag{5.30}$$

$$B(s) = (n \oplus 0_{n-k,m-k})\ V(s) \tag{5.31}$$

$$A_1(s) = V^{-1}(s)(d \oplus I_{m-k}) \tag{5.32}$$

$$B_1(s) = U(s)(n \oplus 0_{n-k,m-k}) \tag{5.33}$$

$$X(s) = U(s)(q \oplus I_{n-k}) \tag{5.34}$$

and

$$Y(s) = V^{-1}(s)\ (p \oplus 0_{m-k,n-k})\ . \tag{5.35}$$

Note that since U(s) and V(s) are elementary matrices, their inverses are easily generated during the canonical reduction phase of the algorithm.

Matrix Spectral Factorization

The two spectral factorizations in Equations (A.12) and (A.13) are computed by the synthesis program using the algorithm developed by Tuel [45]. While spectral factorizations are crucial to the success of the synthesis program, they are difficult to compute numerically. If the computed factors do not contain sufficient accuracy, the synthesis program may fail to compute any valid controller.

Matrix spectral factorization is outlined briefly here and full details of Tuel's algorithm can be obtained by consulting Reference [45]. Given an r x r spectral matrix, $G_1(s)$, whose elements consist of rational

polynomials, and which has the following properties:[2]

1. $G_1(s)$ is real, i.e., $\bar{G}_1(s) = G_1(\bar{s})$;

2. $G_1^T(-s) = G_1(s)$;

3. $G_1(s)$ is of normal rank r almost everywhere;

4. $G_1(j\omega)$ is positive semidefinite for every finite ω;

then the spectral factor, H(s), can be computed such that

$$G_1(s) = H^T(-s)\, H(s)\,. \tag{5.36}$$

Since Tuel's algorithm can factor only polynomial elements and $G_1(s)$ contains rational elements, G(s) is written alternatively as

$$G_1(s) = \frac{1}{g(s)}\, G_1'(s) \tag{5.37}$$

where $G_1'(s)$ and g(s) are, respectively, matrix and scalar polynomials. This form is compatible with the rational polynomial matrix representation of the synthesis program. The spectral factor, h(s), is computed for g(s) and H'(s) for $G_1'(s)$ such that

$$g(s) = h(-s)\, h(s) \tag{5.38}$$

$$G_1'(s) = H'^T(-s)\, H'(s)\,. \tag{5.39}$$

The matrix spectral factor H(s) becomes

$$H(s) = \frac{1}{h(s)}\, H'(s)\,. \tag{5.40}$$

Briefly stated, the spectral factorization G'(s) or g(s) is performed by first mapping the continuous plane onto the discrete plane, solving

---

[2]The overbar denotes complex conjugation.

the discrete factorization problem using an iterative procedure, then mapping the solution back to the continuous plane. Particular attention was paid to accuracy when this algorithm was incorporated into the synthesis program. The effects of arithmetic precision and frequency scaling were studied and the following conclusions were obtained:

1. The accuracy of the continuous plane to discrete plane mapping depends heavily on the precision of the arithmetic employed.

2. The convergence of the iterative equations used to do the discrete factorization is directly affected by the range of coefficient magnitudes in the polynomials of G'(s) or in g(s). Preliminary frequency scaling of G'(s) or g(s) can result in significantly faster convergence.

The effect of scaling is easily demonstrated with an example. The left-hand side of Equation (A.13) (from Appendix A) which results for Example 1 of Chapter VI is

$$\begin{aligned} A(s)\,G(s)\,A^{T}(-s) = (&5.76\times10^{8} - 7.32\times10^{8}s^{2} + 1.5325\times10^{8}s^{4} \\ &- 1.567708\times10^{6}s^{6} + 1.306\times10^{3}s^{8} \\ &- 1.0s^{10})/(100. - s^{2}) . \end{aligned} \tag{5.40}$$

The numerator polynomial in this equation was factored using the spectral factorization algorithm of Tuel without scaling. The factorization required 460 iterations to converge to an answer accurate to 16 significant digits. The denominator required only a single iteration to converge due to the fact that it is only a second-order polynomial. The matrix $\Omega(s)$ from Equation (A.13) was obtained as

$$\begin{aligned} \Omega(s) = (&2.4\times10^{4} \\ &+ 3.9708866441249 43\times10^{4}s \\ &+ 1.759987654268718\times10^{4}s^{2} \end{aligned}$$

$$+ 2.014812383065610\times10^{3}s^{3}$$
$$+ 7.304536101718718726\times10^{1}s^{4}$$
$$+ 1.0s^{5})/(1.0\times10^{1} + 1.0s) . \qquad (5.41)$$

Using a scale factor of 10.0 and the change of variable defined by Equation (5.1), the spectral function in Equation (5.40) becomes

$$A(s)\,G(s)\,A^{T}(-s) = (5.76\times10^{8} - 7.32\times10^{10}s^{2} + 1.5325\times10^{12}s^{4}$$
$$- 1.567708\times10^{12}s^{6} + 1.306\times10^{11}s^{8}$$
$$- 1.0\times10^{10}s^{10})/(100.0 - 100.0s) . \qquad (5.42)$$

Notice that the range of coefficient magnitudes is approximately half what it is in Equation (5.40). The spectral factorization of the numerator of Equation (5.42) required only 96 iterations to converge with the same 16 significant digit accuracy. The unscaled solution for this case is identical to Equation (5.41).

The effect of scaling was as dramatic for matrix factorization problems as it was for the above scalar problem. Example 3 of Chapter VI demonstrates the performance of the factorization problem for the matrix case.

The factorization of Equation (A.13) requires a slightly modified approach. Repeated here, Equation (A.13) is

$$A(s)\,G(s)\,A^{T}(-s) = \Omega(s)\,\Omega^{T}(-s) . \qquad (5.43)$$

The spectral factorization algorithm, however, computes the factor $\Omega'(s)$ such that

$$A(s)\,G(s)\,A^{T}(-s) = \Omega'^{T}(-s)\,\Omega'(s) \qquad (5.44)$$

which is not the desired result. In order to obtain the correct factor, the synthesis program first computes an intermediate matrix as

$$P_1(s) = [A(s)\, G(s)\, A^T(-s)]^T \tag{5.45}$$

and the factorization is carried out using $P_1(s)$. The factorization yields the matrix $\Omega''(s)$ such that

$$P_1(s) = [A(s)\, G(s)\, A^T(-s)]^T = \Omega''^T(-s)\, \Omega''(s)\,. \tag{5.46}$$

Transposing each matrix in Equation (5.46) yields

$$P_1^T(s) = A(s)\, G(s)\, A^T(s) = \Omega''^T(s)\, \Omega''(-s)\,. \tag{5.47}$$

Comparing Equation (5.47) with Equation (5.43), the desired factor is

$$\Omega(s) = \Omega''^T(s)\,. \tag{5.48}$$

Once the factorization of $P_1(s)$ is complete, the synthesis program must then transpose the resulting matrix factor to obtain the correct factor.

Partial Fraction Expansion of Rational Polynomial Matrices

The required partial fraction expansions are shown in Equation (A.19). The general problem can be stated as follows. Let $P_1(s)$ be a matrix of rational polynomials. Then the equation

$$P_1(s) = \{P_1(s)\}_\infty + \{P_1(s)\}_- + \{P_1(s)\}_+ \tag{5.49}$$

represents the partial fraction expansion of $P_1(s)$, where $\{P_1(s)\}_\infty$ is the part associated with the pole at infinity; $\{P_1(s)\}_-$ is the part which has all of its poles in Real(s) $\geq$ 0; and $\{P_1(s)\}_+$ is the part which has all of its poles in Real(s) < 0.

An algorithm was developed to do the partial fraction expansion which is based on the solution of a set of simultaneous equations. A

rational polynomial matrix is represented by the synthesis program in the form

$$P_1(s) = \frac{1}{g(s)} P_1'(s) \tag{5.50}$$

where $P_1'(s)$ and $g(s)$ are, respectively, matrix and scalar polynomials. The algorithm expands each element of $P_1(s)$ separately; therefore, it is necessary for the algorithm to compute each rational polynomial element using $g(s)$ and $P_1'(s)$. This is done in the following manner. Let $P_{ij}'(s)$ be a polynomial element of $P_1'(s)$. Compute the GCD of $g(s)$ and $P_{ij}'(s)$ and divide to obtain

$$n(s) = \frac{P_{ij}'(s)}{GCD(P_{ij}'(s), g(s))} \tag{5.51}$$

and

$$d(s) = \frac{g(s)}{GCD(P_{ij}'(s), g(s))} \tag{5.52}$$

which yields the desired rational polynomial element defined as $n(s)/d(s)$. This insures $n(s)$ and $d(s)$ to be relatively prime and the number of simultaneously equations to be solved minimal.

The denominator polynomial $d(s)$ must be split into two polynomials, one containing the roots which lie in Real(s) ≥ 0, and one containing the roots which lie in Real(s) < 0. These are designated $d^+(s)$ and $d^-(s)$, respectively. The algorithm must now compute $a(s)$, $b(s)$, and $c(s)$ such that

$$\frac{n(s)}{d(s)} = c(s) + \frac{a(s)}{d^+(s)} + \frac{b(s)}{d^-(s)} . \tag{5.53}$$

The above equation can now be rearranged into a form similar to Equation (5.20) and stated as

$$n(s) = c(s)\, d^{+}(s)d^{-}(s) + a(s)d^{-}(s) + b(s)d^{+}(s) . \qquad (5.54)$$

The algorithm assumes the order of polynomial a(s) to be one less than the order of $d^{+}(s)$, the order of b(s) to be one less than the order of $d^{-}(s)$, and the order of c(s) to be the difference between the order of n(s) and d(s). If the order of n(s) is less than the order of d(s), then c(s) is assumed to be the zero polynomial.

Matrices are set up representing the simultaneous equation set in a manner similar to those of Equation (5.25). The difference is that the vector of unknowns contain the coefficients of a(s), b(s), and c(s); the solution vector contains the known coefficients of n(s); and the constant coefficient matrix is formed using the coefficients of $d^{+}(s)$ and $d^{-}(s)$. The equation set is then solved using a high accuracy linear equation solver and the coefficients of the unknown polynomials are obtained.

The above process is repeated for each element of $P_1(s)$. As the expansion of each element is computed, the desired part of the expansion is placed into a matrix. The algorithm then returns the solution matrix in the standard matrix representation form similar to Equation (5.50).

The main problem experienced using this algorithm was the inaccuracies in splitting each d(s) into its corresponding right- and left-hand s-plane parts. The splitting was done by first factoring d(s) and then forming the coefficients of $d^{+}(s)$ and $d^{-}(s)$ with the resulting roots. In general, the partial fraction expansion algorithm was the major source of inaccuracy within the synthesis program and further research is needed to improve the algorithm.

CHAPTER VI

EXAMPLES

Three separate examples are presented in this chapter to illustrate the performance and application of the controller synthesis program which has been developed. The first example, from Youla, Bongiorno, and Jabr [29], is a single-input single-output controller design problem. Their design problem is done here to demonstrate the performance of the program developed by this research. Controllers computed by the program are compared with the controller obtained through hand calculation by Youla et al. [29].

The second example shows an application of the synthesis program to a real, nontrivial controller design problem. The program is used to design two separate single-input single-output controllers for a stabilization loop of an airborne laser pointing and tracking system. The two controllers are computed for different values of the saturation weighting parameter k (see Equation (3.11)). The performance of both is then compared with the performance of the stabilization system with its original controller. The results of this example illustrate the usefulness of the frequency-domain controller synthesis program.

The third example is an extension of the application in example two to the multivariable case. Although the current version of the synthesis program was unable to completely determine a controller, the example is useful in identifying specific problems in the numerical procedure.

Identification of these problems areas is useful for establishing future research directions.

Example One

The example presented here is from the work of Youla, Bongiorno, and Jabr [29], and a complete treatment of the problem can be found in the thesis of Jabr [64]. The results computed by Jabr were used to verify the controller computed by the synthesis program. The results presented here comprise the first complete machine computation of a controller utilizing optimal frequency-domain synthesis theory.

A block diagram of the plant used for this example is shown in Figure 3. The plant matrix is defined as

$$P(s) = \frac{s - 1}{s(s - 2)} \tag{6.1}$$

and is both unstable and non-minimum phase. The feedback sensor consists of a pure delay element,

$$F(s) = e^{-0.1s} , \tag{6.2}$$

which cannot be accommodated as is by the design theory.

To make the feedback sensor compatible the pure delay must be approximated by a suitable rational transfer function. For this example a Páde approximation is used and the feedback transfer function becomes

$$F(s) = F_e(s)\, F_t(s) = \frac{1200 - 60s + s^2}{1200 + 60s + s^2} . \tag{6.3}$$

There is no feedforward in this example; therefore,

$$L(s) = L_o(s) = G_\ell(s) = 0 . \tag{6.4}$$

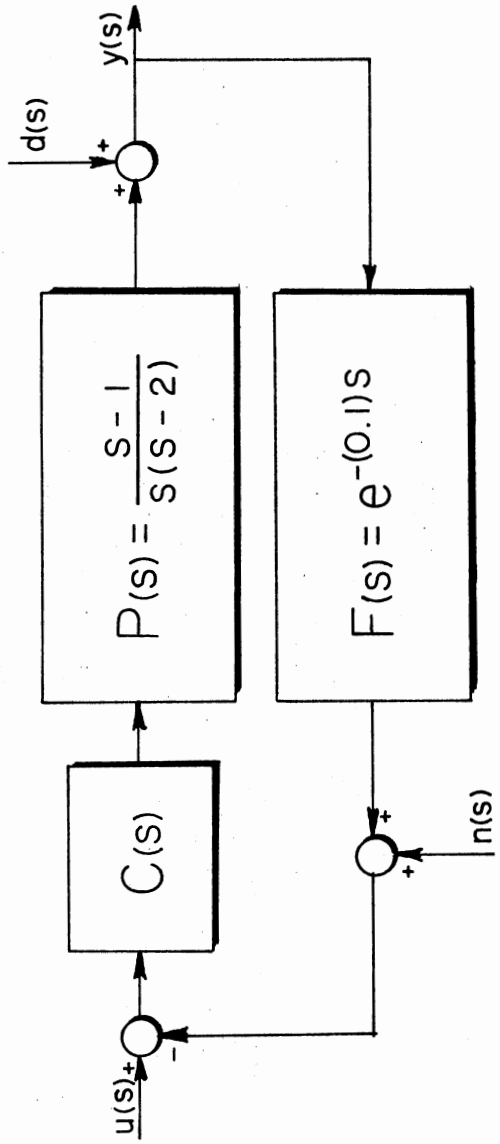

Figure 3. Example One Control Loop

Other related transfer functions are defined as

$$P_o(s) = 1 \tag{6.5}$$

$$F_o(s) = 1 \tag{6.6}$$

The F(s) P(s) transfer function has a pole at the origin; therefore, the closed-loop system can track a step input with zero steady error and the input spectral density becomes

$$G_u(s) = \frac{-1}{s^2} \tag{6.7}$$

In this example, the input itself must be protected from saturation; therefore,

$$P_s(s) = 1 \tag{6.8}$$

and

$$k = 4 . \tag{6.9}$$

The disturbance spectral density and the spectral density of the measurement noise are

$$G_d(s) = \frac{1}{100 - s^2} \tag{6.10}$$

and

$$G_m(s) = 1 , \tag{6.11}$$

respectively.

The controller obtained by Jabr [63] for this system was defined as

$$C(s) = \frac{k_o(s-\alpha_1)(s-\alpha_2)(s-\alpha_3)(s-\bar{\alpha}_3)}{(s-\beta_1)(s-\beta_2)(s-\beta_3)(s-\beta_4)(s-\bar{\beta}_4)} \tag{6.12}$$

where

$$k_o = 67.228808 \quad . \tag{6.13a}$$

$$\alpha_1 = 0.0148746 \tag{6.13b}$$

$$\alpha_2 = -9.9999638 \tag{6.13c}$$

$$\alpha_3 = -30. + j17.320508 \tag{6.13d}$$

$$\beta_1 = 2.41327103 \tag{6.13e}$$

$$\beta_2 = -9.9806404 \tag{6.13f}$$

$$\beta_3 = -33.654632 \tag{6.13g}$$

$$\beta_4 = -18.0573239 + j14.991623 \quad . \tag{6.13h}$$

The validity of this controller was verified by Jabr, making it a suitable reference which the program generated controllers may be compared against. The model in this example is not complicated enough to warrant use of the generalized model preparation program.

The synthesis program was executed several times under identical conditions except that the number of significant digits (variable NDIG) was changed for each run. NDIG was initially set to 26 in all of the runs. NDIG was then reduced after the spectral factorization to a different value for each run. This allowed an investigation of the importance of precision to the controller computation.

Table I shows the roots of the resulting controllers as they were computed for various values of NDIG. The roots were obtained directly from the numerator and denominator polynomials of the controller computed by the program. The table shows that for some values of NDIG the numerator and denominator of the controller contain identical roots. This is due to the fact that, with NDIG significant digits, the synthesis program could not reduce the controller polynomials any further.

TABLE I

CONTROLLERS COMPUTED FOR EXAMPLE ONE

NDIG	Gain	Numerator Roots	Denominator Roots
24	67.2288	-0.244256	-0.244256
		0.0148746	2.41327
		-2.04703	-2.04703
		2.00000	2.00000
		2.00000	2.00000
		2.00000	2.00000
		-9.99996	-9.98064
		-10.0000	-10.0000
		-30.0000 ±j17.3205	-33.6546
		-30.0000 ±j17.3205	-30.0000 ±j17.3205
		-30.0000 ±j17.3205	-30.0000 ±j17.3205
			-18.0573 ±j14.9916
20	67.2288	0.0148746	-0.244256
		5.04536	-2.04703
		-9.99996	7.91363×10^{-19}
			-10.0000
15	67.2288	-0.244256	-0.2244256
		0.0148746	2.41327
		-2.04703	-2.04703
		2.00000	2.00000
		-9.99996	-9.98064
		-1.00000	-1.00000
		-30.0000 ±j17.3205	-33.6546
			-18.0573 ±j14.9916
10	67.2288	-0.244256	-0.244256
		-2.04703	-2.04703
		0.0148746	2.41327
		-9.99996	-9.98054
		-30.0000 ±j17.3205	-33.6546
			-18.0473 ±j14.9916
7	67.2288	0.0	6.39717×10^{-3}
		0.0	6.50306×10^{-3}
		-0.247569	-0.247569
		3.05433×10^{-3}	3.05433×10^{-3}
		-0.0148741	2.41403
		-2.04681	-2.04681
		2.00000	1.99912
		-10.0000	-9.98069
		-30.0000 ±j-17.3205	-33.6546
			-18.0573 ±j14.9916
6	67.2288	0.0	
		-0.244338	-0.244338
		0.0148743	2.41308
		-2.04698	-2.04698
		2.00000	2.00017
		-10.0000	-4.42325×10^{-6}
		-30.0000 ±j17.3205	-9.98069
			-33.6546
			-18.0573 ±j14.9916
5	67.2288	0.0149125	3.87469
			-11.2113
4	67.2288	0.0149125	3.87469
			-11.2113

Note: Underlined values are the roots of the actual optimal controller.

These results show that for this particular example the controller can be reliably computed with no less than 10 significant digits and no more than 15 significant digits. As NDIG is increased upward past 20 digits, the order of the polynomials in the controller grows, since the noise in the coefficients prevents further reduction of the polynomials. This can become a serious problem if the order of the controller becomes too large for it to be analyzed.

Example Two

This example illustrates the application of the frequency-domain synthesis program to a real, non-trivial system. The system under consideration here is a rate stabilized control loop of an airborne pointing and tracking system and is shown in block diagram form in Figure 4. Table II contains the definitions of the various blocks shown on the figure.

TABLE II

DEFINITION OF FUNCTIONS FOR THE CONTROL LOOP OF FIGURE 4

Function	Definition
RIG	$1/(s(1+s/1667)(1+1.2s/3769.9+(s/3769.9)^2)$
J	268.5
DM	2.24
V1	$0.02(1+s/1847)/((1+s/25)(1+s/3562)(1+s/12485)(1+s/9425+(3/9425)^2))$
G1	$1666.67/(1+s/1920)$
B	18000
K	921000
C(s)	$443000(s/35+1)(s/75.4+1)/(((s+1)(s/2557+1)(s/697+1)(s/2055+1)(1+1.2s/1094+(s/1094)^2))$

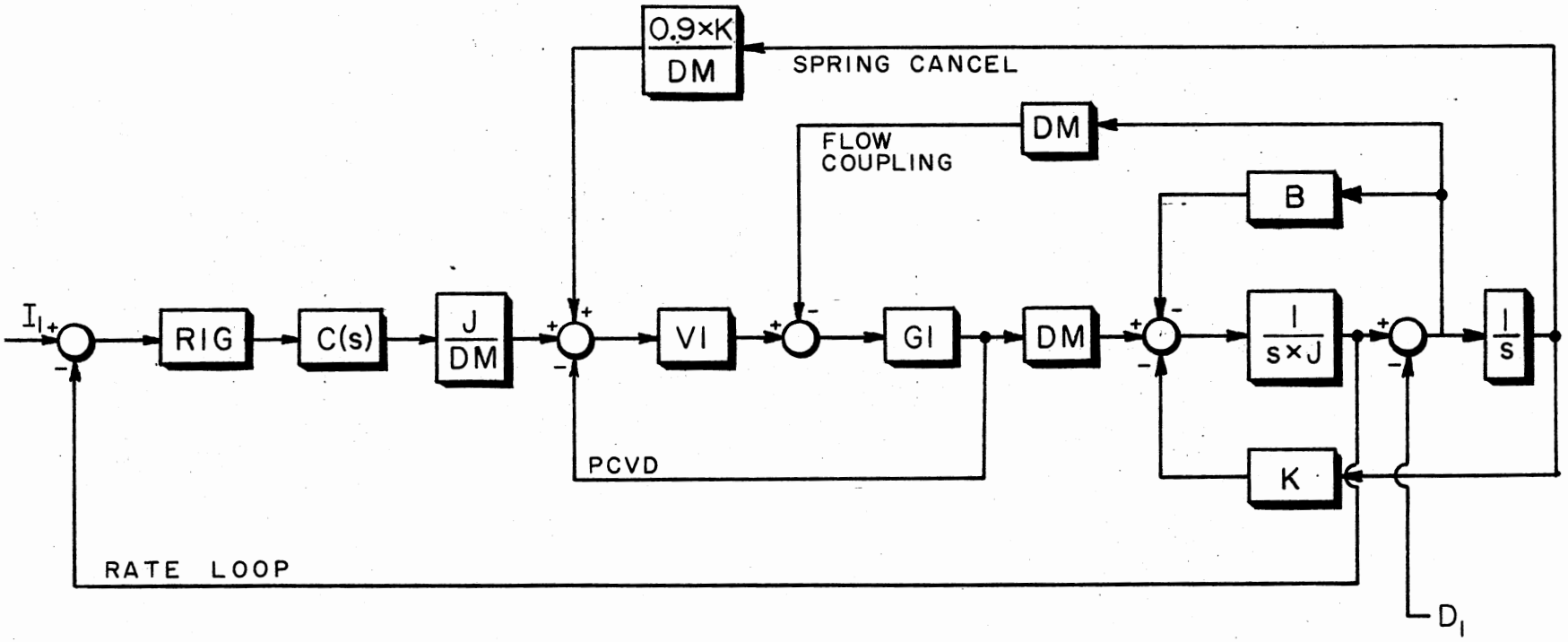

Figure 4. Stabilization Loop for Example Two

This system is in actual existence and utilizes the controller form shown in Table II. The plant is the inner-azimuth gimbal of four gimbal, two degrees of freedom pointing system. The control loop formed around the blocks marked V1 and G1 is a pressure controlled hydraulic drive system for the gimbal. Constant J is the moment of inertia in the gimbal, DM is the effective moment arm of the hydraulic actuator, B is the effective damping in the gimbal mounting, and K is the spring force. The loop marked "SPRING CANCEL" is used to cancel the effects of the spring and causes the rate loop to approximate a Type 1 system. The spring cancel is not 100 percent and the rate loop is not a true Type 1. The block diagram shows the gimbal inertia to be modeled with one angular degree of freedom. The model used for comparisons in this chapter actually had a two-degree of freedom gimbal structure with a resonance near 110 Hz.

The rate loop is driven with a rate command at the point marked I_1 which is generated by a tracker system (not shown). A rate integrating gyro (block RIG) serves as the main sensor element. Motion disturbance in the outer gimbal system enters the stabilization loop at the point marked D_1. The primary concern of this study is the aircraft yaw vibration which is transmitted through the outer gimbal to the inner gimbal. Figure 5 shows the power spectral density (PSD) of the actual rate disturbance entering the loop at point D_1. The approximation is obtained from the function

$$G_D(\omega) = \frac{1.0\times10^{-12}\ ((\omega/0.4)^2 + 1((\omega/320+1)^2 + 1)}{((\omega/56.)^2+1)^2} \qquad (6.14)$$

and serves as the spectral density functions used in the controller synthesis described later.

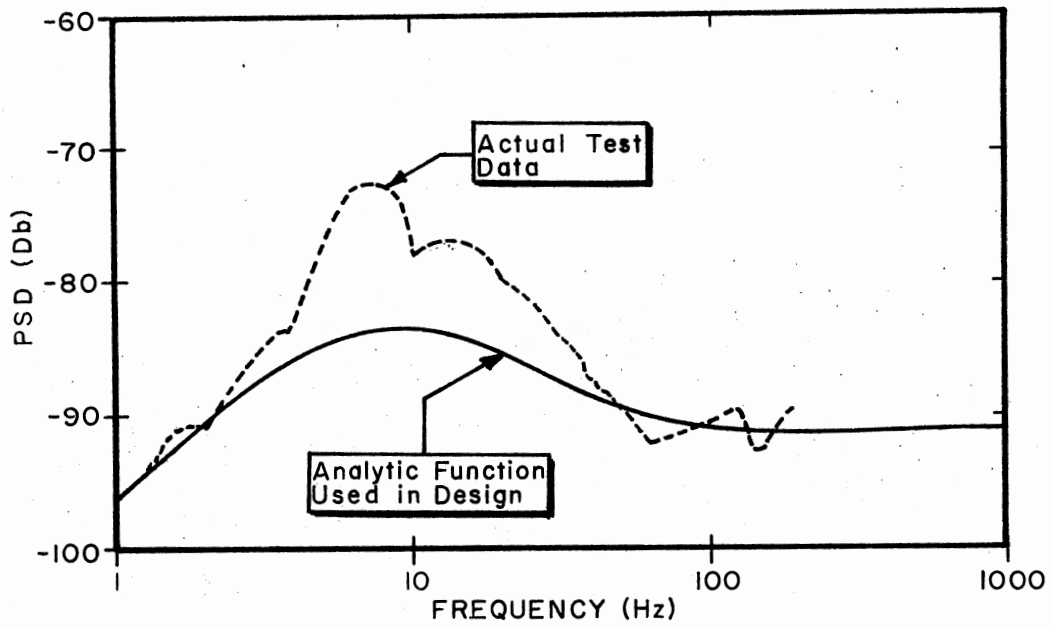

Figure 5. Actual and Analytic PSD of Disturbance Entering the Rate Stabilization Loop of Example Two

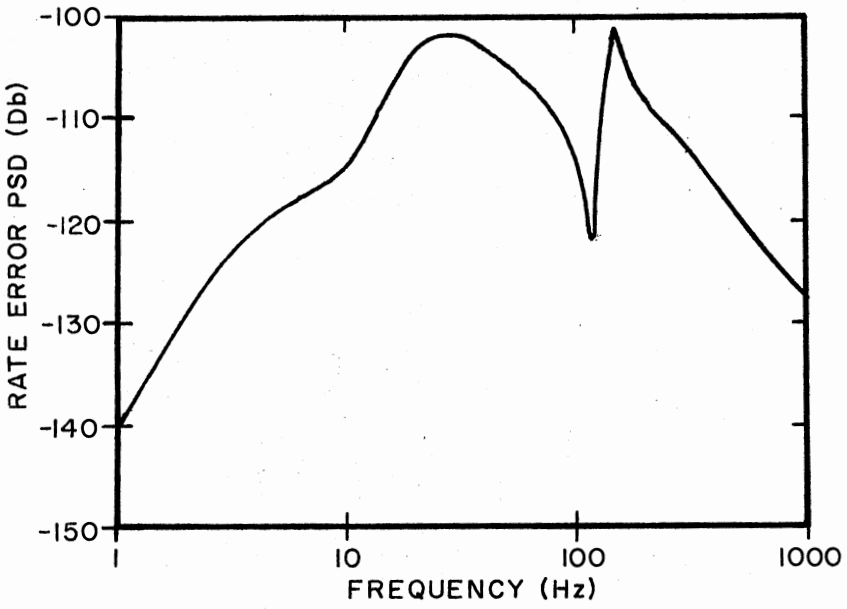

Figure 6. PSD of the Rate Error of the Stabilization System

The performance of the stabilization loop of Figure 4 is shown in Figure 6, which is the rate error response at point R_1 (see Figure 4) to the PSD function $G_D(\omega)$ (Equation (6.14)) applied at point D_1. Figures 7 and 8 show the open and closed loop response of the stabilization system.

The objective of this example was to design a new controller, C(s), which will optimally improve the performance of the system shown. Figure 9 shows a block diagram of the plant considered for the design process. The associated definitions of the blocks in the figure are listed in Table III. The plant is the same as the plant in Figure 4, except that the various transfer functions have been simplified as shown in Table III. These simplifications were necessary as an aid to reducing the numerical difficulties during the controller synthesis process. Because of these changes the synthesis process will produce a somewhat suboptimal controller design. The integrator at the plant input is used to account for the rate integrating gyro which must be considered as part of the plant. The rate integrating gyro cannot be included in the feedback measurement system because the synthesis theory allows only stable measurement systems. The effects of spring have been removed so that a true Type 1 plant is possible.

The general model representation is now formulated for this plant. Even though the plant may not warrant the general model representation, its use makes calculation of the necessary transfer functions easy. The model equations can now be written as described in Chapter III:

$$R_p(1) = \frac{r_1 \; J}{s \; DM} - R_p(s) \; G1 \tag{6.15a}$$

$$R_p(2) = R_p(1)V1 - s \; R_p(4) \; DM \tag{6.15b}$$

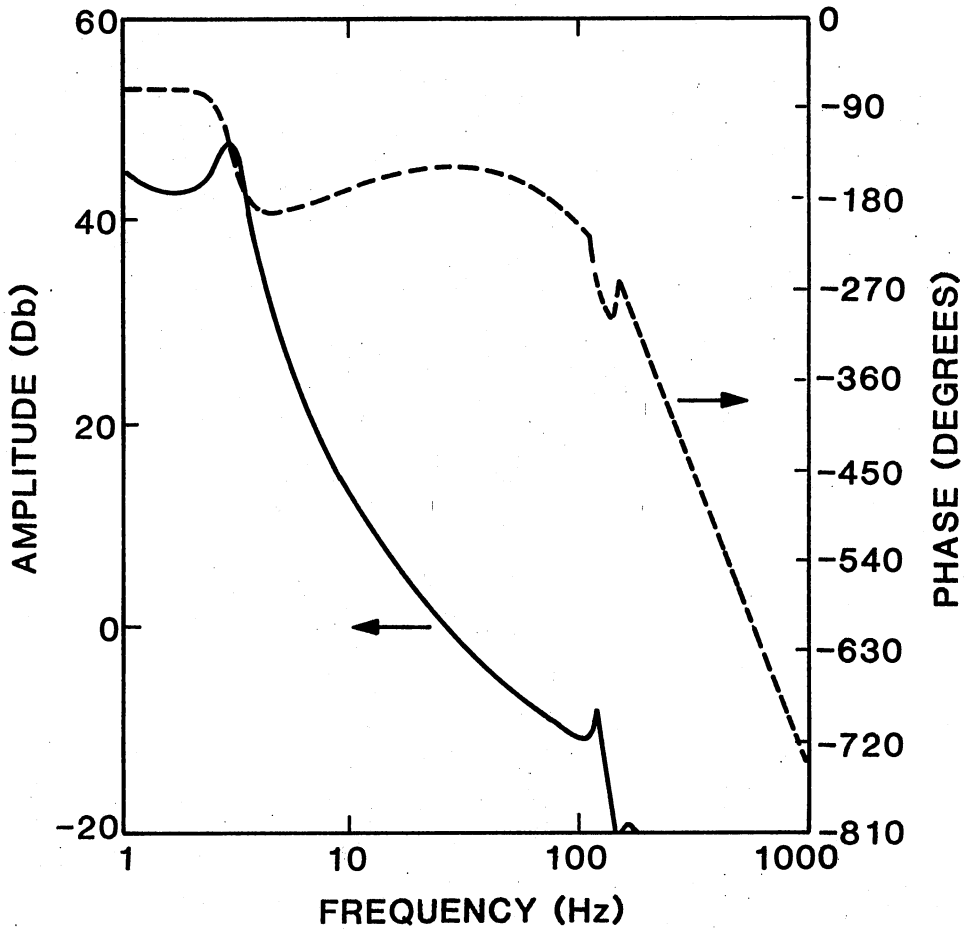

Figure 7. Open Loop Response of the Rate Stabilization System

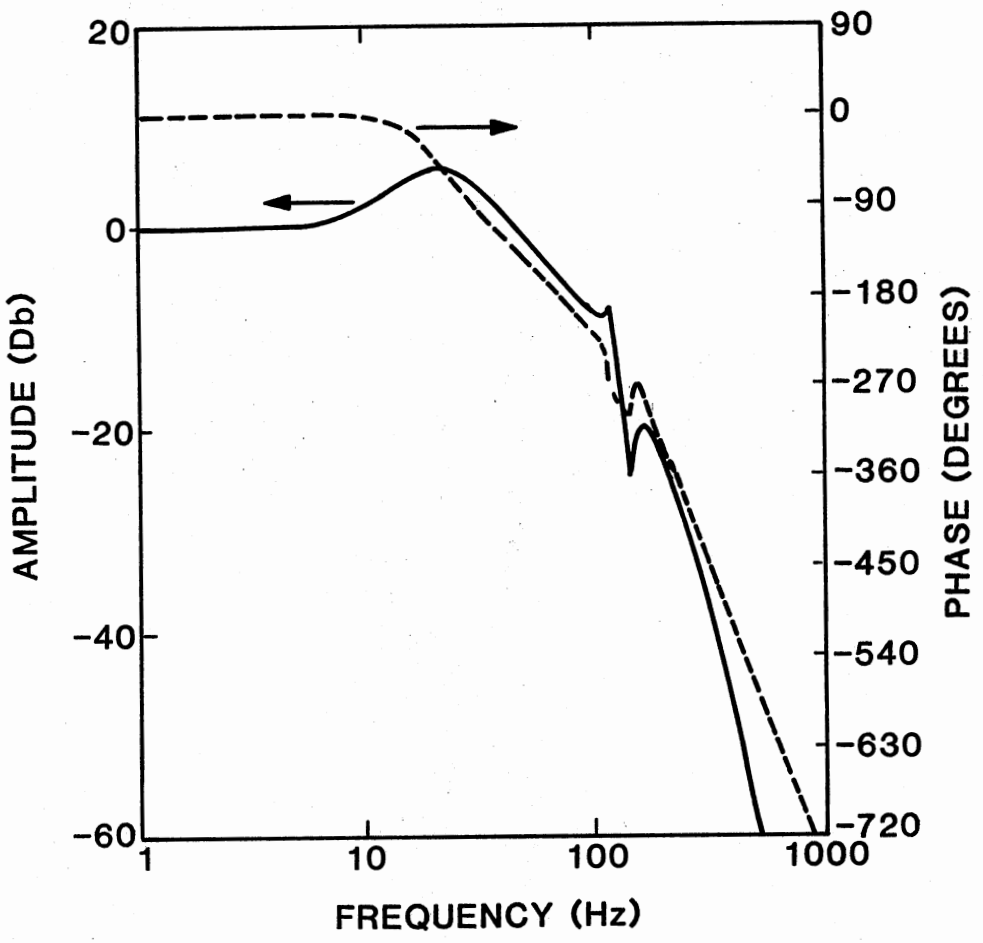

Figure 8. Closed-Loop Response of the Rate Stabilization System

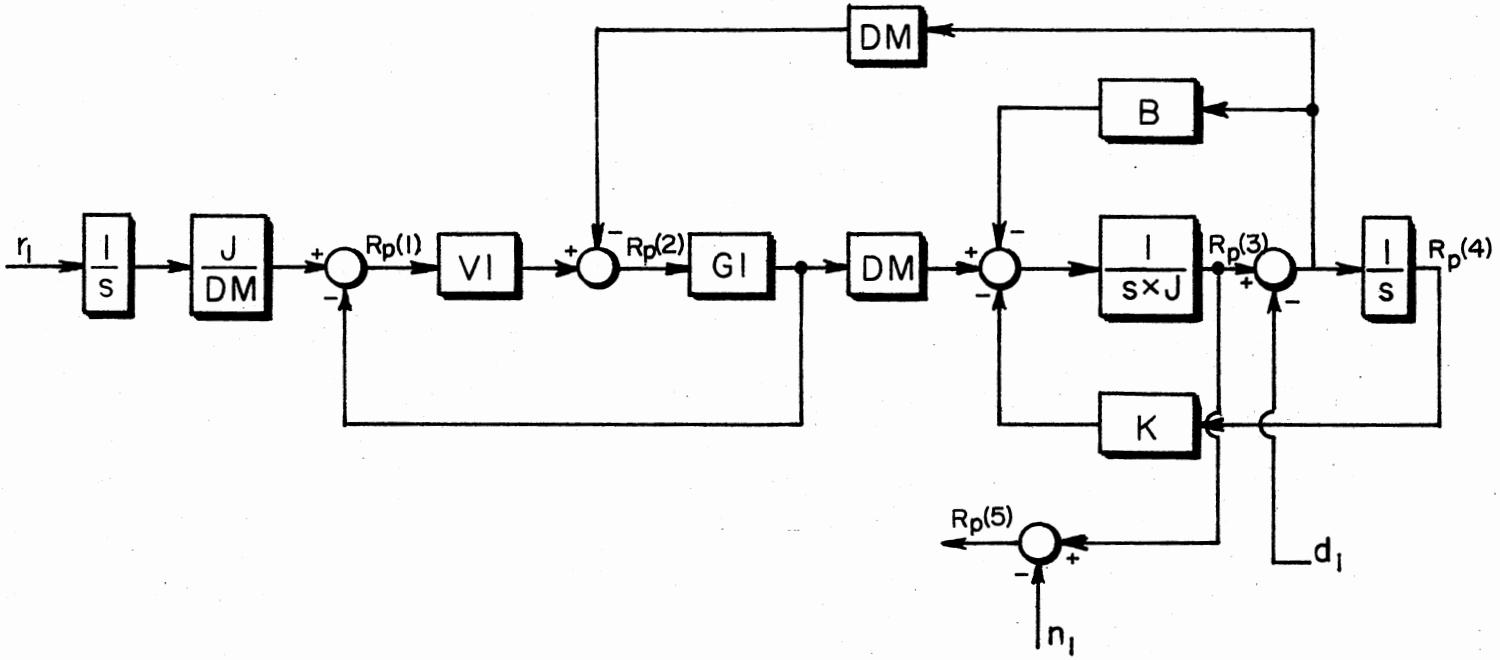

Figure 9. Block Diagram of the Plant Used for the Controller Synthesis Process of Example Two

TABLE III

DEFINITIONS OF FUNCTIONS FOR THE PLANT OF FIGURE 9

Function	Definition
J	260.0
DM	2.2
V1	0.02 (1 + s/1800)/((1 + s/25)(1 + s/3500))
G1	1666.67/(1 + s/1900)
B	18000.0
K	0.0 (spring cancelled)

$$R_p(3) = R_p(2)\,\frac{G1\ DM}{s\ J} - R_p(4)\,\frac{(s\ B + K)}{s\ J} \tag{6.15c}$$

$$R_p(4) = R_p(3)/s - d_1/s \tag{6.15d}$$

$$R_p(5) = R_p(3) + n_1 \tag{6.15e}$$

Thus, the $A_p(s)$ and $B_p(s)$ matrices are defined as

$$A_p(s) = \begin{bmatrix} 1 & G1 & 0 & 0 & 0 \\ -V1 & 1 & 0 & s\ DM & 0 \\ 0 & \frac{-G1\ DM}{s\ J} & & \frac{(s\ B + K)}{s\ J} & 0 \\ 0 & 0 & -\frac{1}{s} & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{bmatrix} \tag{6.16}$$

$$B_p(s) = \begin{bmatrix} \frac{r_1\ J}{s\ DM} \\ 0 \\ 0 \\ -d_1/s \\ n_1 \end{bmatrix} \tag{6.17}$$

A FORTRAN version of the general model representation preprocessor program (see Appendix B) was used to compute the transfer functions for the synthesis program. Since this is a single-input single-output system, the plant input is designated to the model preprocessor as r_1 and the plant output was designated as $R_p(3)$. The measured output was designated as $R_p(5)$, the disturbance input as d_1, and the measurement noise as n_1. The preprocessor used the algorithm described in Chapter V to compute the inverse $A_p(s)$ matrix. Since the FORTRAN preprocessor was used, the $A_p(s)$ and $B_p(s)$ matrices had to be defined with the actual

numerical values for the polynomial coefficients. The program used a frequency scaling value of 1000.0 and easily computed the required transfer functions as (shown scaled):

$$P(s) = (5.7\times10^{-6} + 3.1667\times10^{-6}s)/(0.0 + 3.9651\times10^{-1}s + 6.7530\times10^{0}s^2 + 1.0569\times10^{1}s^3 + 5.592\times10^{0}s^4 + s^5) \quad (6.18)$$

$$F(s)\ P(s) = P(s) \quad (6.19)$$

$$L(s) = L_o(s) = 0 \quad (6.20)$$

$$P_o(s) = (3.9651\times10^{-1} + 8.8205\times10^{-1}s + 4.2510\times10^{-1}s^2 + 6.6667\times10^{-2}s^3)/(3.9651\times10^{-1} + 6.7531\times10^{0}s + 1.0569\times10^{1}s^2 + 5.5917\times10^{0}s^3 + s^4) \quad (6.21)$$

$$F_o(s) = 1 \quad (6.22)$$

$$P_o(s) = 1 \quad (6.23)$$

$$F(s)\ P_o(s) = P_o(s)\ . \quad (6.24)$$

The $P_s(s)$ matrix was set to unity by the preprocessor program, but the saturation point to be protected was at the output of the gyro integrator. $P_s(s)$ was set manually to be

$$P_s(s) = 1.0\times10^{-3}/s\ . \quad (6.25)$$

Since the effects of the spring K were removed, the system was capable of tracking a step type input with zero steady-state error; therefore, $G_u(s)$ was defined as

$$G_u(s) = -1.0\times10^{-7}/s^2\ . \quad (6.26)$$

The spectral density $G_d(s)$ was defined to be the function shown in Equation

(6.14) and when scaled is defined as

$$G_d(s) = (9.8345\times10^{-18} - 6.1466\times10^{-11}s^2 + 6.0025\times10^{-10}s^4)/$$
$$(9.8345\times10^{-6} - 6.272\times10^{-3}s^2 + 1.0s^4) \ . \qquad (6.27)$$

The spectral density of the measurement noise was assumed as

$$G_m(s) = 1.0\times10^{-3} \qquad (6.28)$$

and

$$G_\ell(s) = 0.0 \ . \qquad (6.29)$$

Matrix Q_t, the transient weighting matrix, was set as

$$Q_t = 1.0 \qquad (6.30)$$

and for the first design to be tested, the saturation weighting constant, k, was set as

$$k = 1.0 \ . \qquad (6.31)$$

The synthesis program was set to initially use 24 significant digits and then to use 10 significant digits after the spectral factorization steps. The program then computed the controller to be

$$C(s) = \frac{K(s/\alpha_1 + 1)}{(s/\beta_1 + 1)} \qquad (6.32)$$

where

$$K = 597.24 \qquad (6.33a)$$

$$\alpha_1 = 6.51\times10^{-2} \qquad (6.33b)$$

$$\beta_1 = 7.59\times10^{-2} \ . \qquad (6.33c)$$

The roots shown are the scaled roots obtained from the synthesis program and the unscaled roots are obtained by multiplying these values by 1.000.

Figures 10 and 11 show the open and closed loop response of the original, unsimplified stabilization system resulting from the use of the computed controller. Figure 12 shows the PSD of the rate error due to the disturbance. Comparison of Figure 12 with Figure 6 shows that the use of the new controller resulted in significant reduction of system performance.

The synthesis process was repeated with a saturation weighting value, k, set equal to 1.0×10^{-8}. When the synthesis program was executed for this run, the number of significant digits had to be reduced to seven after the spectral factorization steps. The program computed the new controller to be (unscaled)

$$C(s) = \frac{K(s/\alpha_1+1)(s/\alpha_2+1)(s/\alpha_3+1)(s/\bar{\alpha}_3+1)}{(s/\beta_1+1)(s/\beta_2+1)(s/\beta_3+1)(s/\bar{\beta}_3+1)} \tag{6.34}$$

where

$$K = 18571.2 \tag{6.35a}$$

$$\alpha_1 = 65.4 \tag{6.35b}$$

$$\alpha_2 = 1258.0 \tag{6.35c}$$

$$\alpha_3 = 2134.0 + j533.3 \tag{6.35d}$$

$$\beta_1 = 3800.0 \tag{6.35e}$$

$$\beta_2 = 1806.0 \tag{6.35f}$$

$$\beta_3 = 1788.0 + j2634.0 \; . \tag{6.35g}$$

It was observed that the numerator roots α_2 and α_3 were fairly close to the denominator roots β_2 and β_3, so the alternate controller

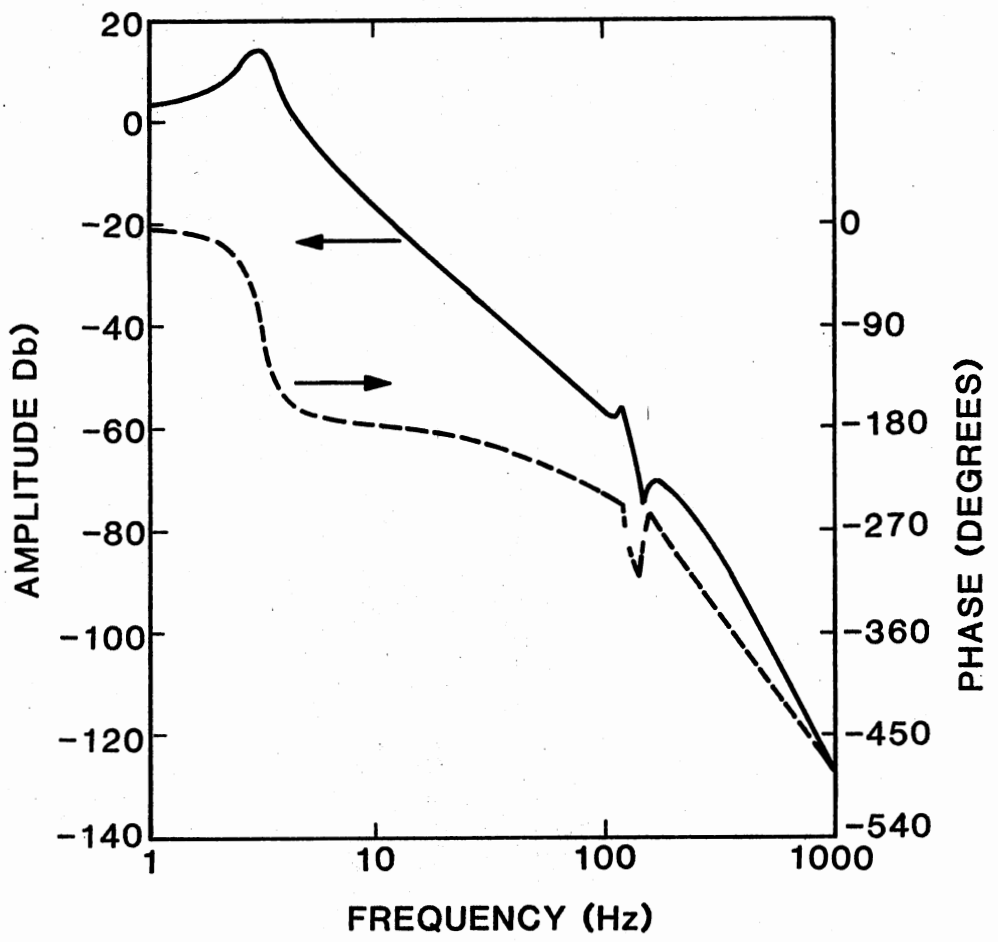

Figure 10. Open Loop Response of Original Stabilization System With New Controller for k = 1.0

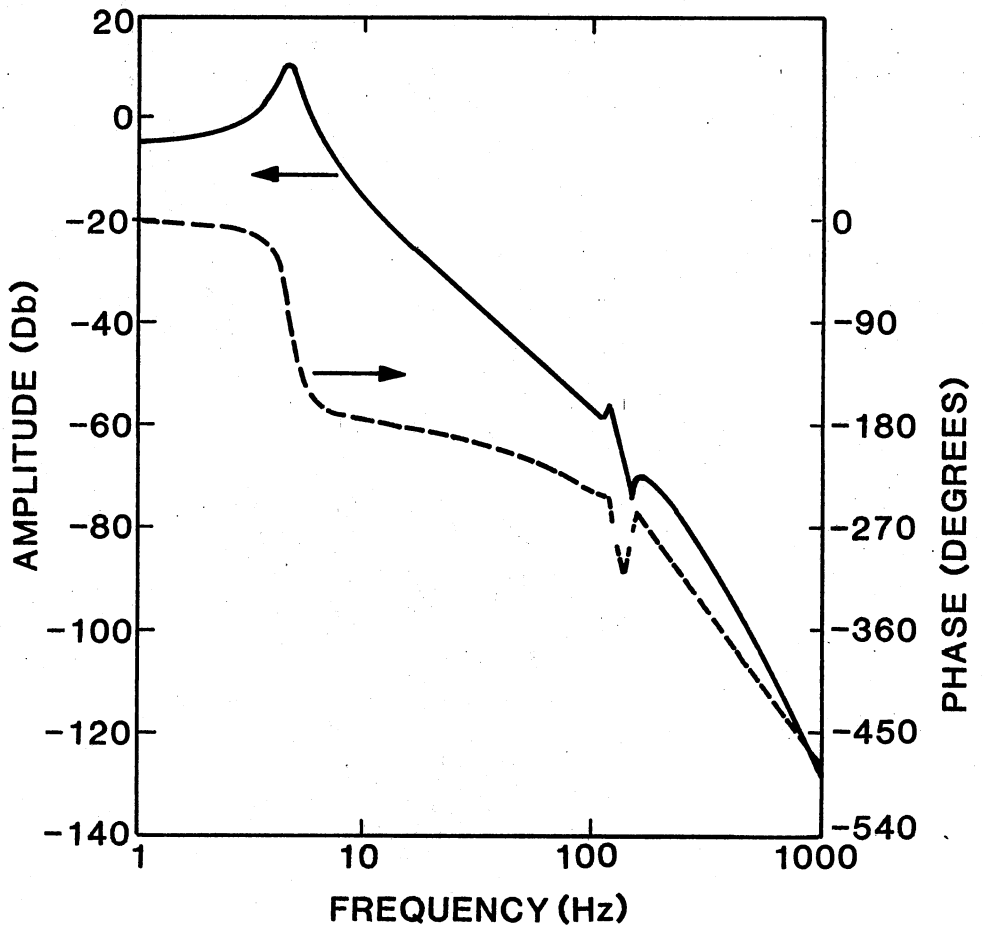

Figure 11. Closed Loop Response of Original Stabilization System With New Controller for k = 1.0

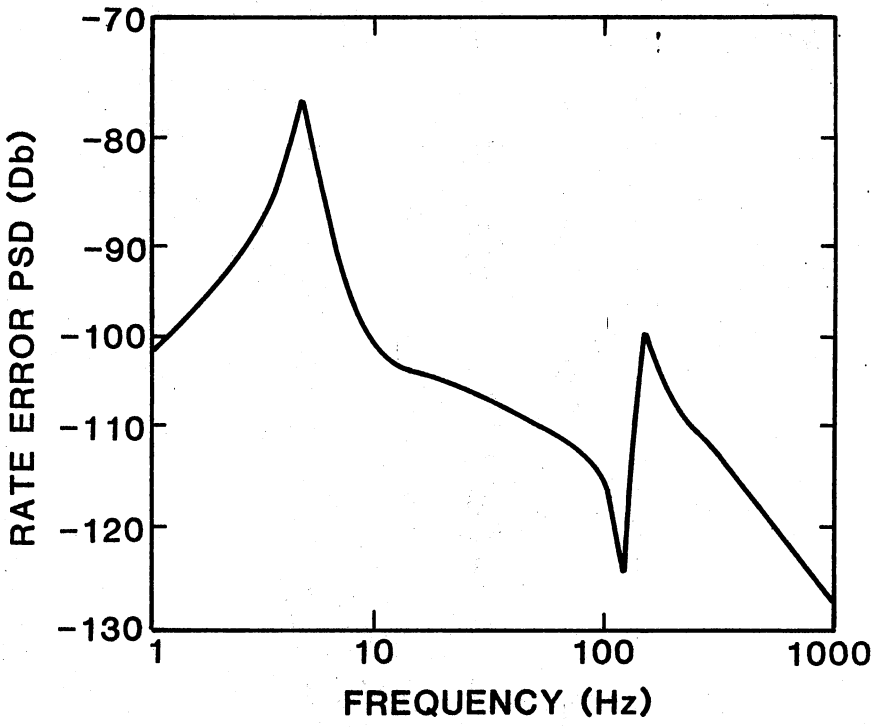

Figure 12. PSD of Rate Error in Original Stabilization System With New Controller and k = 1.0

$$C(s) = \frac{K(s/\alpha_1 + 1)}{(s/\beta_1 + 1)} \qquad (6.36)$$

where

$$K = 18571.2 \qquad (6.37a)$$

$$\alpha_1 = 65.3 \qquad (6.37b)$$

$$\beta_1 = 1806.0 \qquad (6.37c)$$

was analyzed along with the controller of Equation (6.34). The result of the analysis showed very little difference in performance, so the results of the analysis for the simpler, suboptimal controller is presented here.

Figures 13 and 14 show the open loop and closed loop responses of the original stabilization system resulting from the use of the controller defined in Equation (6.36). When the closed-loop response of Figure 14 is compared with the original closed-loop response of Figure 8, it can be seen that the bandwidth has been increased and the resonance peak around 10 Hz has been significantly reduced.

Figure 15 shows the PSD of the rate error due to the disturbance. Comparison of Figure 15 with Figure 6 shows significant improvement in the ability of the stabilization system to reject the disturbance entering the loop.

An interesting measure of performance is illustrated in the comparisons of Figure 16. In this figure the cumulative RMS power in the rate error of the stabilization system for the different controllers is plotted. The cumulative RMS levels at 1000 Hz are representative of the total RMS in the rate error resulting from the disturbance. The objective of the optimal controller design was the minimization of the mean-square rate error; therefore, the curves in Figure 16 provide an

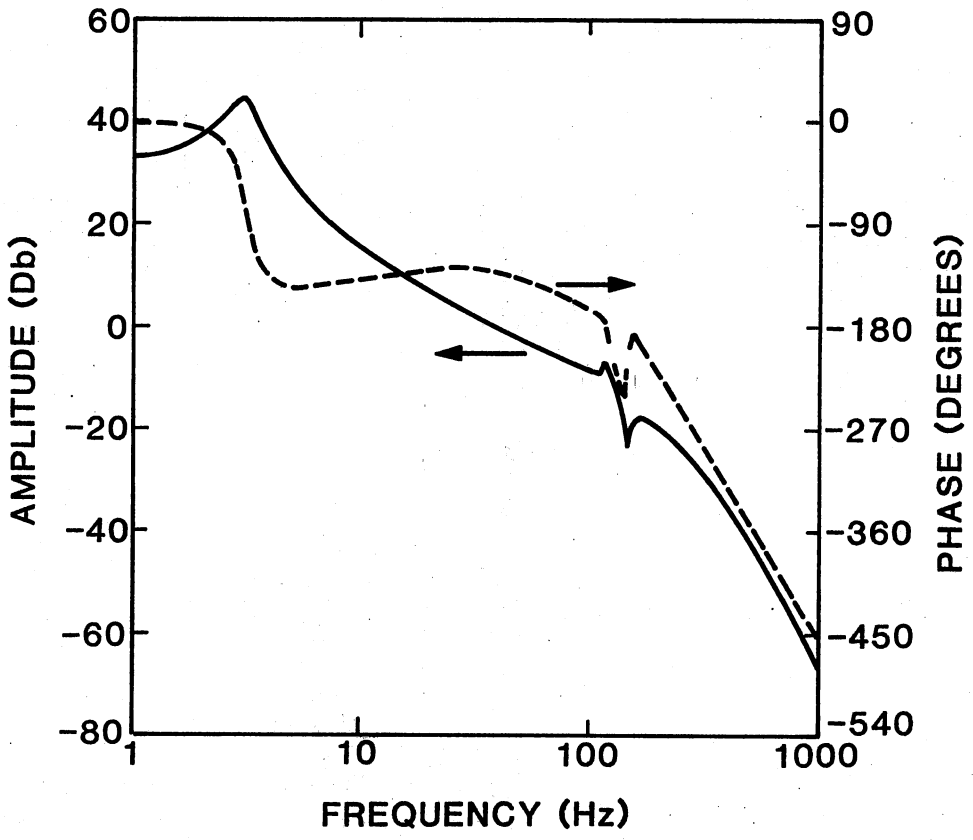

Figure 13. Open Loop Response of the Original Stabilization System With New Controller and $k = 1.0 \times 10^{-8}$

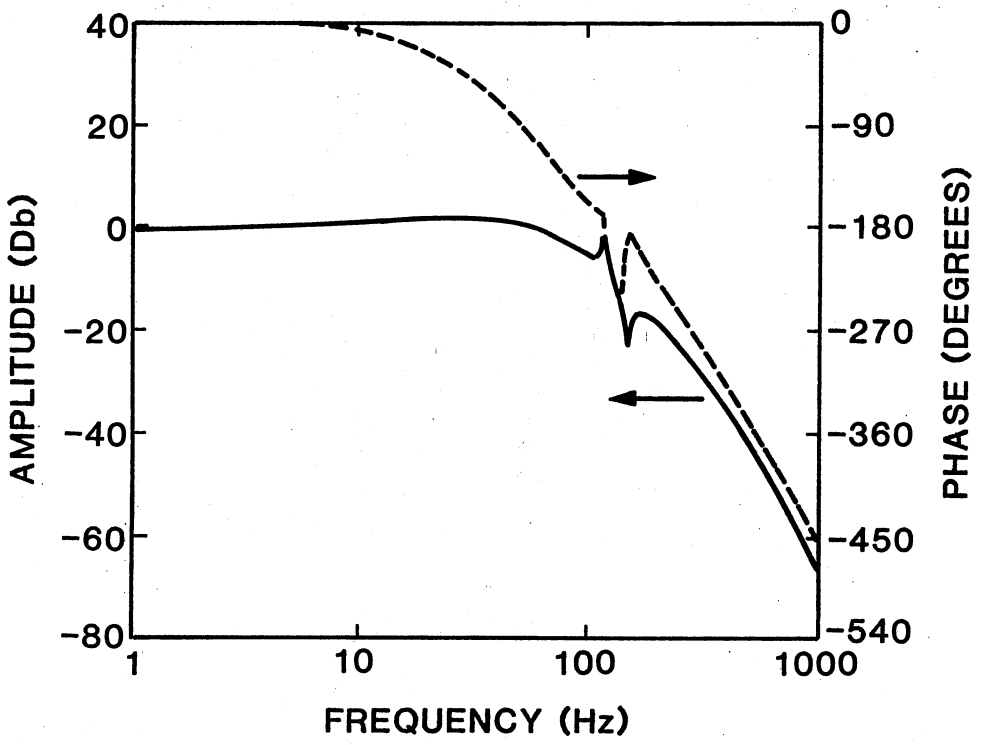

Figure 14. Closed Loop Response of the Original Stabilization System With New Controller and $k = 1.0 \times 10^{-8}$

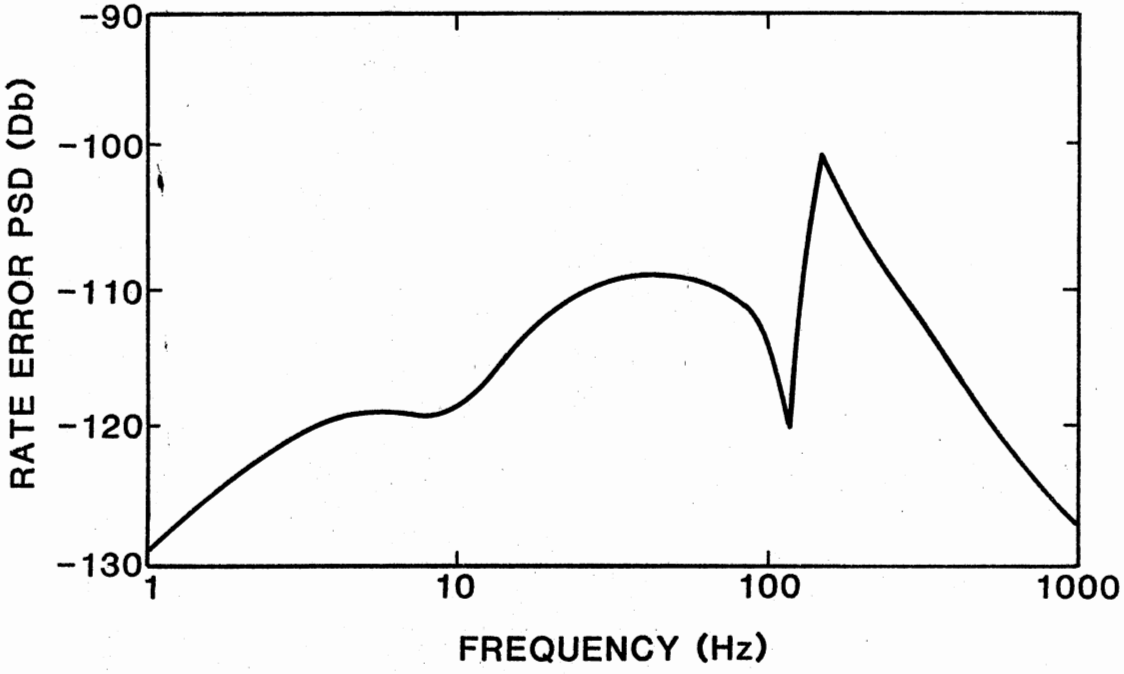

Figure 15. PSD of Rate Error in Original Stabilization System With New Controller and $k = 1.0 \times 10^{-8}$

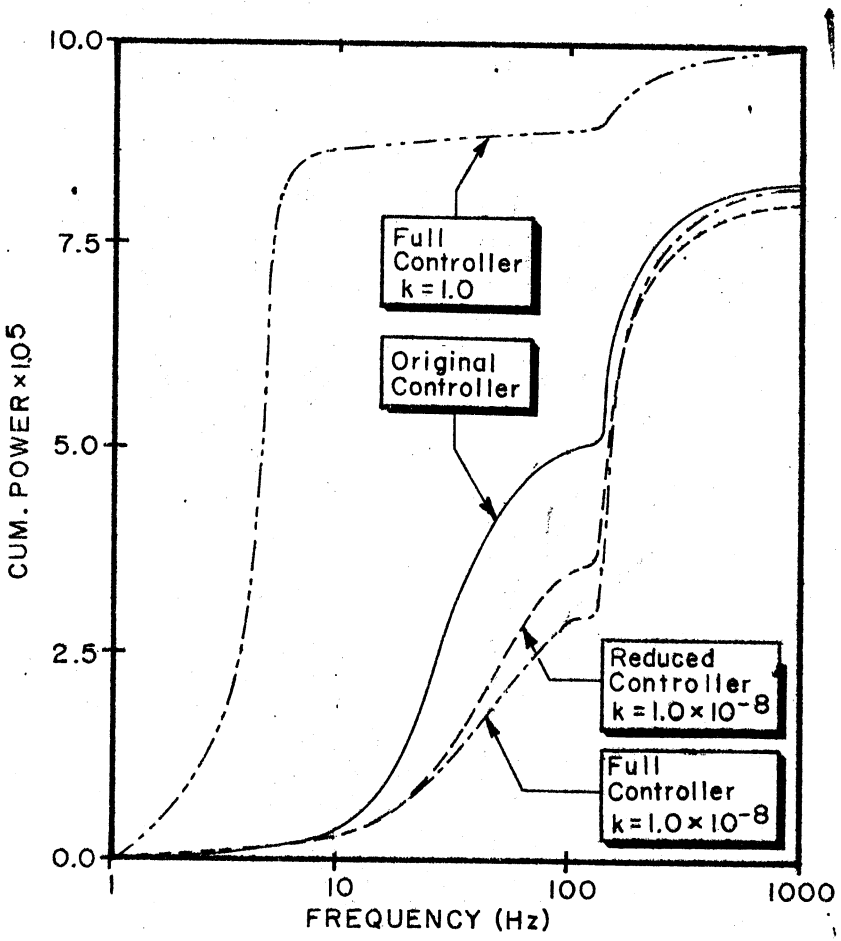

Figure 16. Comparison of the Cumulative RMS Power in the Rate Error of the Original Systems for Various Controllers

indication of how well that objective was achieved. It is also interesting to note that the use of the new controllers (with $k = 1.0 \times 10^{-8}$) tends to reduce the power levels in the lower frequency region.

It should be noted that the analytic disturbance PSD function (Equation (6.14)) is flat after 100 Hz. This causes the rate error RMS levels of Figure 16 to be high; however, if the PSD function were made to decrease after 100 Hz, the levels would be lower. The flat PSD function was used for the synthesis because actual PSD data were uncertain after 100 Hz.

Finally, the open and closed loop responses of the simplified stabilization system with the controller of Equation (5.56) are shown in Figures 17 and 18, respectively. Comparison of these figures with Figures 14 and 15 shows the use of the simplified model for the controller designs was reasonable.

Example Three

In this example, the design of two-input two-output controller for the plant shown in Figure 19 was attempted. The function definitions for the various blocks shown in the figure are the same as those for Figure 9 and are listed in Table III. In this example, the simplified plant of example two has been expanded to include an ideal tracker. The integrator representing the rate integrating gyro has been removed from the front of the gimbal drive system and its output was designated as a plant output.

The general model representation was used to compute the transfer function matrices for the synthesis program. The plant input was designated as r_1 and r_2, the plant output was designated as $R_p(5)$ and $R_p(6)$.

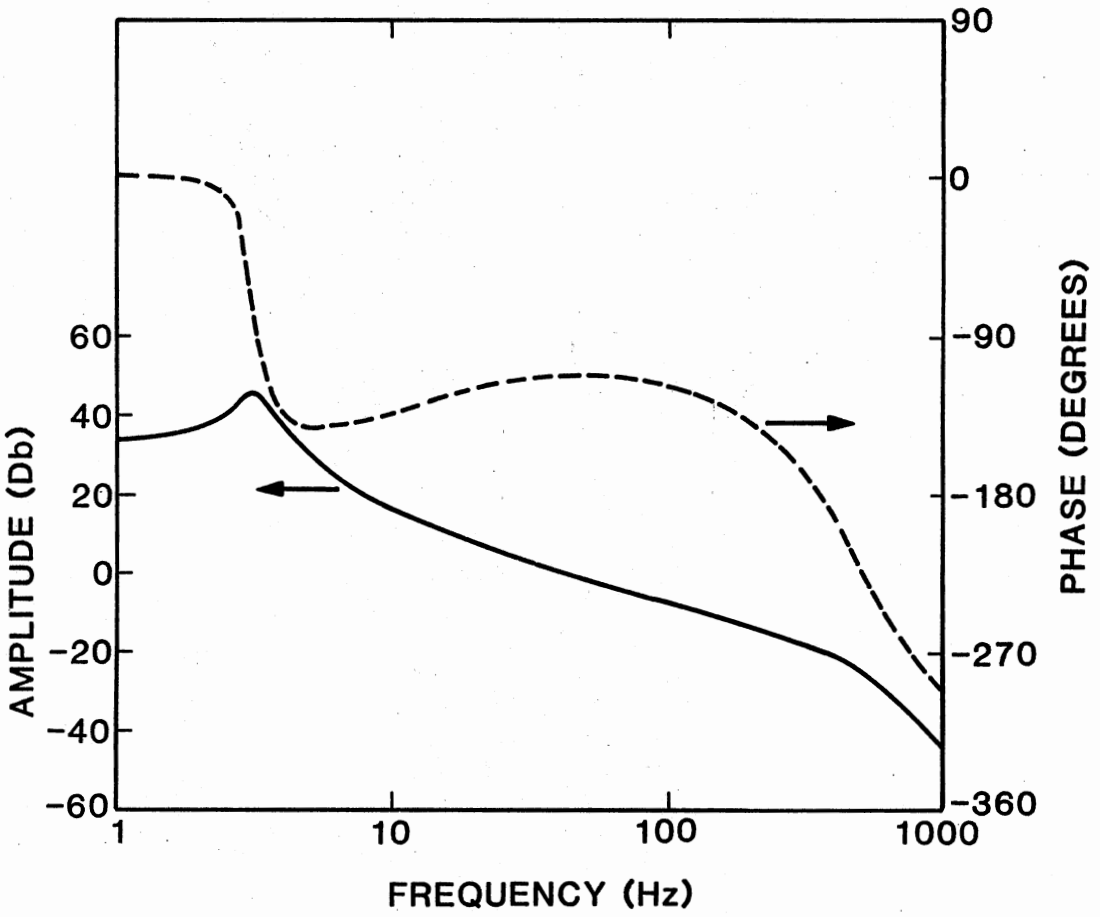

Figure 17. Open Loop Response of the Simplified Stabilization System With New Controller and $k = 1.0 \times 10^{-8}$

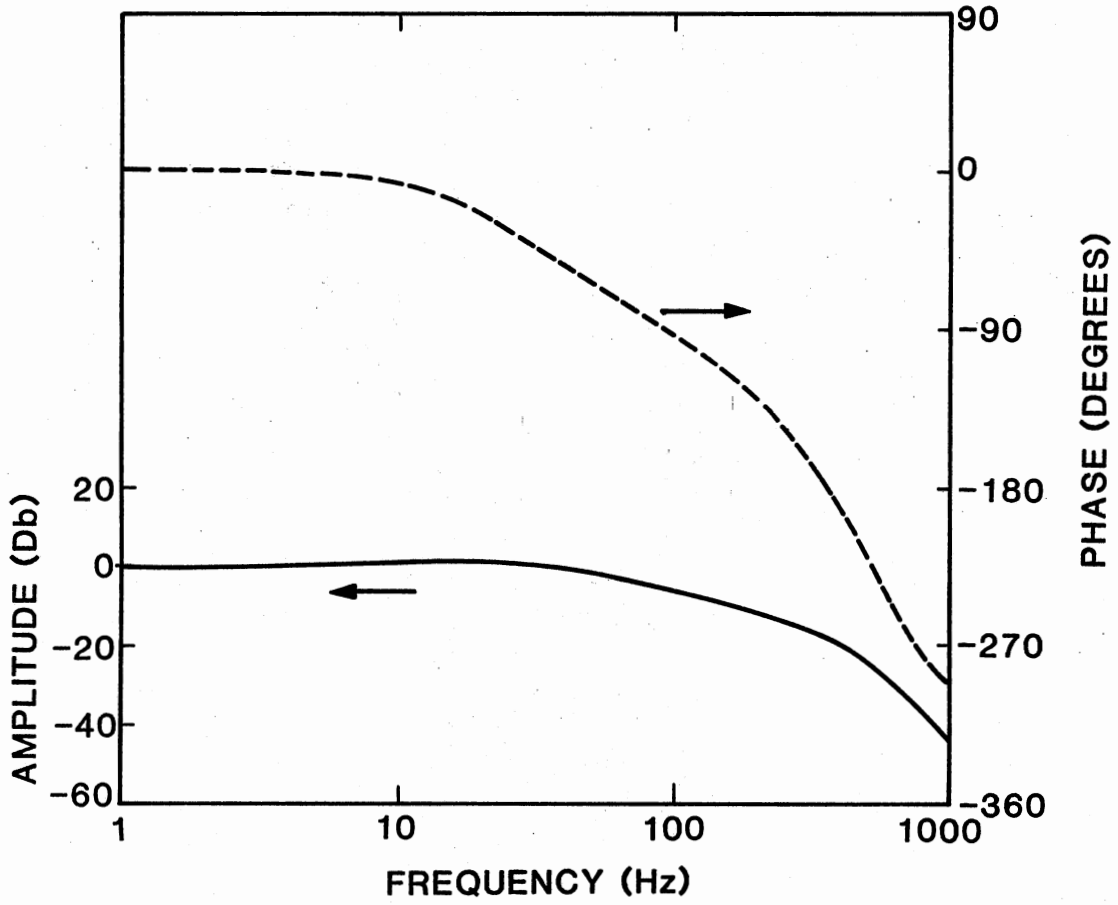

Figure 18. Closed Loop Response of the Simplified Stabilization System With New Controller and $k = 1.0 \times 10^{-8}$

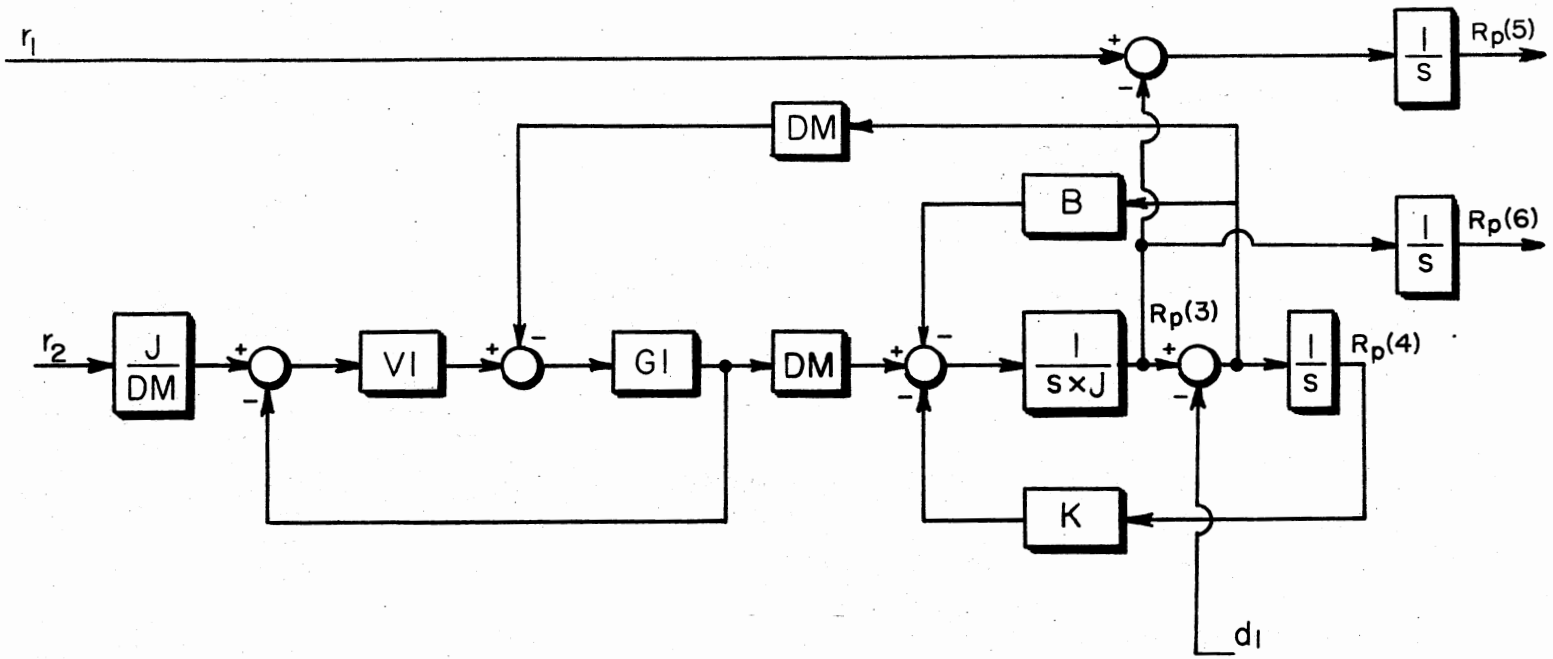

Figure 19. Block Diagram of the Plant Used for the Controller Synthesis Process of Example Three

The disturbance input was again designated to be d_1. No noise input was considered in this example. The FORTRAN version of the model preprocessor was used and the polynomials were frequency scaled using a scale factor of 100.0. The necessary transfer function matrices were computed as:

$$P(s) = \frac{1}{p'(s)} \begin{bmatrix} p_{12}(s) & p_{12}(s) \\ p_{21}(s) & p_{22}(s) \end{bmatrix} \tag{6.38}$$

where

$$p'(s) = 0.0 + 3.9651\times10^{3}s + 6.7530\times10^{3}s^{2} + 1.0569\times10^{3}s^{3} + 5.591667\times10^{1}s^{4} + 1.0s^{5} \tag{6.39a}$$

$$p_{11}(s) = 3.9651\times10^{1} + 6.7530\times10^{1}s + 1.0569\times10^{1}s^{2} + 5.591667\times10^{-1}s^{3} + 1.0\times10^{-2}s^{4} \tag{6.39b}$$

$$p_{12}(s) = -5.7\times10^{-1} - 3.1667\times10^{-2}s \tag{6.39c}$$

$$p_{22}(s) = -p_{12}(s) \tag{6.39d}$$

$$p_{21}(s) = 0 \tag{6.39f}$$

$$F(s)\ P(s) = P(s) \tag{6.40}$$

$$L(s) = L_o(s) = 0 \tag{6.41}$$

$$P_o(s) = \frac{1}{p'(s)} \begin{bmatrix} q_{11}(s) & q_{12}(s) \\ q_{21}(s) & q_{22}(s) \end{bmatrix} \tag{6.42}$$

and

$$q_{11}(s) = -3.9651\times10^{1} - 8.8205\times10^{0}s - 4.2510\times10^{-1}s^2 - 6.6667\times10^{-3}s^3 \quad (6.43b)$$

$$q_{12}(s) = 0 \quad (6.43b)$$

$$q_{21}(s) = 0 \quad (6.43c)$$

$$q_{22}(s) = -q_{11}(s) \quad (6.43d)$$

$$F_o(s) = 0 \quad (6.44)$$

$$F(s)\ P_o(s) = P_o(s) \quad (6.45)$$

$$P_s(s) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} . \quad (6.46)$$

The spectral density matrix for the input was defined as

$$G_u(s) = \begin{bmatrix} 0 & 0 \\ 0 & \dfrac{-1.0\times10^{-7}}{s^2} \end{bmatrix} . \quad (6.47)$$

The entries in this matrix indicated that the output of the gyro is required to stay close to zero and that the pointing angle is required to follow a step type input. The same disturbance function used for example two was used in this example; therefore,

$$G_d(s) = (9.8345\times10^{-14} - 6.1466\times10^{-9}s^2 + 6.0025\times10^{-10}s^4)/(9.8345\times10^{-2} - 6.272\times10^{-1}s^2 + 1.0s^4) . \quad (6.48)$$

The remaining spectral density matrices were zero.

The transient weight matrix was assumed to be

$$Q_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.49}$$

and the saturation weighting constant, k = 1, was assumed.

The prototype synthesis program was not able to compute the optimal controller due to convergence problems encountered at the matrix spectral factorization steps. The underlying reasons are best illustrated by examining the numerical values of one of the matrices which had to be factored. The matrix from Equation (A.12) was computed by the program to be

$$A_1^T(-s)(P^T(-s)Q_tP(s) - kP_3^T(-s)P_s(s))A_1(s) = \begin{bmatrix} r_{11}(s) & r_{12}(s) \\ r_{21}(s) & r_{22}(s) \end{bmatrix} \tag{6.50}$$

where the polynomial elements are (5 significant digits are shown but the computations were done with 24 significant digits):

$$\begin{aligned} r_{11}(s) = {} & 1.393\times10^{3} - 4.2196\times10^{8}s^2 + 2.1921\times10^{11}s^4 \\ & - 5.1772\times10^{11}s^6 + 7.6912\times10^{9}s^8 - 4.3172\times10^{7}s^{10} \\ & + 1.2033\times10^{5}s^{12} - 1.6969\times10^{2}s^{14} + 9.9723\times10^{-2}s^{16} \end{aligned} \tag{6.51a}$$

$$\begin{aligned} r_{12}(s) = {} & 3.6845\times10^{-4} - 1.0315\times10^{-2}s - 1.8375\times10^{4}s^2 \\ & + 5.0004\times10^{5}s^3 - 8.4279\times10^{3}s^4 - 1.1804\times10^{6}s^5 \\ & + 1.1925\times10^{5}s^6 + 8.5845\times10^{3}s^7 - 1.1878\times10^{3}s^8 \\ & - 7.9867\times10^{-1}s^9 + 3.2551\times10^{0}s^{10} - 5.4130\times10^{-2}s^{11} \\ & - 3.2150\times10^{-3}s^{12} + 8.4791\times10^{-5}s^{13} \end{aligned} \tag{6.51b}$$

$$r_{21}(s) = r_{12}(-s) \tag{6.51c}$$

$$\begin{aligned} r_{22}(s) = {} & 2.3423\times10^{-8} - 1.1356\times10^{0}s^{2} + 2.6835\times10^{0}s^{4} \\ & - 2.6661\times10^{-2}s^{6} + 7.3018\times10^{-5}s^{8} \\ & - 7.2094\times10^{-8}s^{10} \ . \end{aligned} \tag{6.51d}$$

Based on the investigations of spectral factorization and frequency scaling, it is believed that the large range in the magnitudes of the coefficients within each polynomial was the major reason that the spectral factorization failed. The values shown in Equation (6.51) resulted from the scale factor of 100.0 mentioned earlier. Other attempts used scale factors of 1000.0 and 10000.0 with no success.

The failure of the synthesis program in this example does not mean that smaller order designs will fail. There is every indication that the program can succeed for smaller, well-conditioned problems.

CHAPTER VII

SUMMARY AND RECOMMENDATIONS

Summary

Computer implementation of frequency-domain controller design theory has been accomplished by this research. The underlying causes of many of the numerical problems associated with the manipulation of rational polynomial matrices have been identified and arithmetic and algorithmic improvements demonstrated.

The system size which can be successfully treated with the synthesis program is limited but significantly greater than that which can be conveniently treated manually. The capability of the synthesis program and theory was demonstrated in the second example of Chapter VI. It is believed that the success attained by the program in that example will provide the stimulus needed for continued research in this area.

Investigations made during the course of this research indicate that the best arithmetic for use in frequency-domain controller design programs is the floating-point arithmetic. As the prototype program developed by this study evolves toward a production oriented program, the use of symbolic systems such as REDUCE combined with floating-point type systems may prove to be beneficial. The precision of floating-point arithmetic is the main factor limiting the size of systems which can be treated by the synthesis program. The computation of controllers for large multivariable systems requires arithmetic precision which

exceeds the hard capabilities of all modern computers. The use of software to increase the precision of the arithmetic is an alternative which needs to be investigated.

Contributions

The main contributions of this research are summarized as follows:

1. A structure for frequency-domain controller design programs has been defined and a prototype program has been developed around this structure.

2. A generalized model representation has been developed and demonstrated.

3. Direct polynomial representation and floating-point computation have been determined best for the controller synthesis program. This choice was made based upon investigations of the various exact computation methods and the various floating-point computation methods.

4. Precision, calculation noise, and large magnitude differences in polynomial coefficients were identified as the underlying causes of numerical difficulties associated with the use of floating-point arithmetic in the synthesis program. Methods were devised which can be used to overcome these difficulties.

5. It was shown that frequency-scaling significantly improved the performance of the synthesis program and especially the spectral factorization.

6. The machine computation of controllers for a real, non-trivial plant has been demonstrated for the first time.

7. The performance of the rate stabilization loop of an airborne

laser pointing and tracking system has been improved by the use of the synthesis program.

Recommendations

It is recommended that research be continued as follows:

1. Investigate the manner in which coefficient inaccuracies affect the roots of polynomials and develop techniques which can be utilized to avoid increasing the precision of arithmetic within frequency-domain synthesis programs.

2. Investigate the computer implementation of suboptimal frequency-domain synthesis theory using the methods developed by this research.

3. Determine the usefulness of high-precision software arithmetic to synthesis programs.

4. Continue the investigation of the root representation method with emphasis on improving the addition operation. Develop algorithms to perform spectral factorization using the root representation method.

A SELECTED BIBLIOGRAPHY

[1] Kalman, R. E. "Contributions to the Theory of Optimal Control." Bulletin of the Mexican Mathematical Society, Second Series 5 (1960), pp. 102-119.

[2] "Special Issue on the Linear-Quadratic-Gaussian Control Problem." IEEE Transactions on Automatic Control, Vol. AC-16 (December, 1971).

[3] Wiener, N. Extrapolation, Interpolation, and Smoothing of Stationary Time Series. Cambridge, Massachusetts: MIT Press, 1949.

[4] Newton, G. C., Kaiser, J. F., and Gould, L. A. Analytical Design of Linear Feedback Controls. New York: John Wiley and Sons, Inc., 1957.

[5] Amara, R. C. "The Linear Least Squares Synthesis of Multivariable Control Systems." Transactions of the AIEE, Vol. 42, Part 2 (May, 1959), pp. 115-120.

[6] Hsieh, H. C., and Leondes, C. T. "Techniques for the Optimum Synthesis of Multipole Control Systems with Random Processes as Inputs." IRE Transactions on Automatic Control Systems, Vol. AC-4 (December, 1959), pp. 212-231.

[7] Davis, M. C. "Factoring the Spectral Matrix." IEEE Transactions on Automatic Control, Vol. AC-8 (October, 1963), pp. 296-305.

[8] Bongiorno, J. J. "Minimum Sensitivity Design of Linear Multivariable Feedback Control Systems by Matrix Spectral Factorization." IEEE Transactions on Automatic Control, Vol. AC-14 (December, 1969), pp. 665-673.

[9] Freeman, H. "A Synthesis Method for Multipole Control Systems." Transactions of the AIEE, Vol. 76, Part 2 (March, 1957), pp. 28-31.

[10] Freeman, H. "Stability and Physical Realizability Considerations in the Synthesis of Multipole Control Systems." Transactions of the AIEE, Vol. 35, Part 2 (March, 1958), pp. 1-5.

[11] Chen, C. T. "Representations of Linear Time-Invarient Composite Systems." IEEE Transactions on Automatic Control, Vol. AC-13, No. 3 (June, 1968), pp. 227-283.

[12] Chen, C. T. "Stability of Linear Multivariable Feedback Systems." Proceedings of the IEE, Vol. 56 (May, 1968), pp. 821-828.

[13] Ragazzini, J. R., and Franklin, G. F. Sampled Data Control Systems. New York: McGraw-Hill, 1958.

[14] Bigelow, S. C. "The Design of Analog Computer Compensated Control Systems." Transactions of the AIEE, Vol. 77 (November, 1958), pp. 409-415.

[15] Weston, J. E., and Bongiorno, J. J. "Extension of Analytical Design Techniques to Multivariable Feedback Control Systems." IEEE Transactions on Automatic Control, Vol. AC-17, No. 5 (October, 1972), pp. 613-620.

[16] Fallside, F., and Seraji, H. "Design of Optimal Systems by a Frequency-Domain Technique." Proceedings of the IEE, Vol. 117, No. 10 (October, 1970), pp. 2017-2024.

[17] Youla, D. C., Bongiorno, J. J., and Lu, C. N. "Single-Loop Feedback Stabilization of Linear Multivariable Dynamical Plants." Automatica, Vol. 10 (1974), pp. 159-173.

[18] Lanning, J. H., and Battin, R. H. Random Processes in Automatic Control. New York: McGraw-Hill, 1956.

[19] Horowitz, I. M. Synthesis of Feedback Systems. New York: Academic Press, 1963.

[20] Kwakernaak, H. K., and Sivan, R. Linear Optimal Control Systems. New York: Wiley-Interscience, 1972.

[21] Rosenbrock. H. H. "Design of Multivariable Control Systems Using the Inverse Nyquist Array." Proceedings of the IEE, Vol. 116 (1969), pp. 1929-1986.

[22] Belletrutti, J., and MacFarlane, A. G. J. "Characteristic Loci Techniques in Multivariable Control System Design." Proceedings of the IEE, Vol. 118 (1971), pp. 1291-1297.

[23] MacFarlane, A. G. J. "Use of Characteristic Transfer Functions in the Design of Multivariable Control Systems." Proceedings of the 2nd IFAC Conference on Multivariable Systems Theory, Paper No. 1.3.4, Dusseldorf (1971).

[24] Porter, B., and Crowley, R. Modal Control Theory and Applications. New York: Harper and Row, 1972.

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

[26] MacFarlane, A. G. J. "A Survey of Some Recent Results in Linear Multivariable Feedback Theory." Automatica, Vol. 8 (1972), pp. 455-492.

[27] Horowitz, I. M., and Shaked, U. "Superiority of Transfer Function Over State-Variable Methods in Linear Time-Invariant Feedback Systems Design." IEEE Transactions on Automatic Control, Vol. AC-20, No. 1 (February, 1975).

[28] Youla, D. C., Bongiorno, J. J., and Jabr, H. A. "Modern Wiener-Hopf Design of Optimal Controllers Part I: The Single-Input-Output Case." IEEE Transactions on Automatic Control, Vol. AC-21, No. 1 (February, 1976), pp. 3-13.

[29] Youla, D. C., Bongiorno, J. J., and Jabr, H. A. "Modern Wiener-Hopf Design of Optimal Controllers Part II: The Multivariable Case." IEEE Transactions on Automatic Control, Vol. AC-21, No. 3 (June, 1976), pp. 319-338.

[30] MacFarlane, A. G. J. "Return-Difference Matrix Properties for Optimal Stationary Kalman-Bucy Filter." Proceedings of the IEE, Vol. 118, No. 2 (1971), pp. 373-376.

[31] Barrett, J. F. "Construction of Wiener Filters Using the Return-Difference." International Journal of Control, Vol. 26 (1977), pp. 797-803.

[32] Shaked, U. "A General Transfer Function Approach to the Steady-State Linear Quadratic Gaussian Stochastic Control Problems." International Journal of Control, Vol. 24 (1976), pp. 771-800.

[33] Grimbal, M. J. "Design of Stochastic Optimal Feedback Control Systems." Proceedings of the IEE, Vol. 125, No. 11 (1978), pp. 1275-1284.

[34] Bongiorno, J. J., and Youla, D. C. "On the Design of Single-Loop Single-Input-Output Feedback Control Systems in the Complex-Frequency Domain." IEEE Transactions on Automatic Control, Vol. AC-22, No. 3 (June, 1977), pp. 416-423.

[35] Hearn, A. C. REDUCE 2 User's Manual. 2nd Ed. Salt Lake City, Utah: University of Utah, UCP-19, March, 1973.

[36] Aho, A. V., Hopcroft, J. E., and Ullman, J. D. The Design and Analysis of Computer Algorithms. Reading, Massachusetts: Addison-Wesley, 1974.

[37] Knuth, D. E. The Art of Computer Programming, Vol. 2, Semi Numerical Algorithms. Reading, Massachusetts: Addison-Wesley, 1971.

[38] McClellan, M. T. "The Exact Solution of Systems of Linear Equations with Polynomial Coefficients." Journal of the Association for Computing Machinery, Vol. 20 (1973), pp. 563-588.

[39] Horowitz, E., and Sahni, S. "On Computing the Exact Determinant of Matrices with Polynomial Enteries." Journal of the Association for Computing Machinery, Vol. 22 (1975), pp. 38-50.

[40] Gentleman, W. M., and Johnson, S. C. "Analysis of Algorithms, A Case Study: Determinants of Matrices with Polynomial Entries." Association for Computing Machinery Transactions on Mathematical Software, Vol. 2 (1976), pp. 232-241.

[41] Rao, T. M., Krishnamurthy, E. V., and Subramanian, K. "Finite-Segment p-adic Number Systems with Applications to Exact Computation." Proceedings of the Indian Academy of Science, Vol. 81, No. 2 (1975), pp. 58-79.

[42] Krishnamurthy, E. V. "Exact Inversion of a Rational Polynomial Matrix Using Finite Field Transforms." SIAM Journal of Applied Mathematics, Vol. 35, No. 3 (November, 1978), pp. 453-464.

[43] Ramachandran, V. "Exact Reduction of a Polynomial Matrix to the Smith Normal Form." IEEE Transactions on Automatic Control, Vol. AC-24, No. 4 (August, 1979), pp. 638-641.

[44] Youla, D. C. "On the Factorization of Rational Matrices." IRE Transactions on Information Theory, Vol. 7 (July, 1961), pp. 172-189.

[45] Tuel, W. G. "Computer Algorithm for Spectral Factorization of Rational Matrices." IBM Journal of Research and Development, Vol. 12 (1968), pp. 163-170.

[46] Anderson, B. D., Hitz, K. L., and Diem, N. D. "Recursive Algorithm for Spectral Factorization." IEEE Transactions on Circuits and Systems, Vol. CAS-21, No. 6 (November, 1974), pp.

[47] Davis, M. C. "Factoring the Spectral Matrix." IEEE Transactions on Automatic Control, Vol. AC-8, No. 5 (October, 1963), pp. 296-305.

[48] Grimble, M. J. "Factorization Procedure for a Class of Rational Matrices." International Journal of Control, Vol. 28, No. 1 (1978), pp. 105-111.

[49] Downs, T. "On the Inversion of a Matrix of Rational Functions." Linear Algebra Applications, Vol. 4 (1971), pp. 1-10.

[50] Munko, M., and Zakian, V. "Inversion of Rational Polynomial Matrices." Electronic Letters, Vol. 6 (1970), pp. 629-630.

[51] Pace, I. S., and Barnett, S. "Efficient Algorithms for Linear System Calculations Part I--Smith Form and Common Divisor of Polynomial Matrices." International Journal of Systems Science, Vol. 5, No. 5 (1974), pp. 403-411.

[52] Barnett, S. "Some Topics in Algebraic Systems Theory: A Survey." Recent Mathematical Developments in Control. Ed. D. J. Bell. New York: Academic Press, 1973, pp. 323-344.

[53] Collins, G. E. "PM a System for Polynomial Manipulation." Communications of the Association for Computing Machinery, Vol. 9, No. 8 (August, 1966), pp. 578-589.

[54] System 1360 Scientific Subroutine Package, Version III, Programmer's Manual. White Plains, New York: IBM Corporation, Technical Publications Department, No. H20-0205-3, 1969.

[55] Pace, I. S., and Barnett, S. "Comparison of Algorithms for Calculation of G.C.D. of Polynomials." International Journal of Systems Science, Vol. 4, No. 2 (1973), pp. 211-226.

[56] Matthew, G. K. "An Alternative to Euclid's Algorithm." Transactions of the ASME, Vol. 101 (October, 1979), pp. 582-586.

[57] Sankaran, B. "A New Computer Technique of Root Locus Analysis." (Unpub. M.S. thesis, Oklahoma State University, 1979).

[58] User Information for the FRQRSP Frequency Response Program. Stillwater, Oklahoma: Oklahoma State University, Report No. ER-75-R-109-012, January, 1978.

[59] Rosenbrock. H. H. State-Space and Multivariable Theory. New York: Wiley-Interscience, 1970.

[60] Sterbenz, P. H. Floating Point Computation. New Jersey: Prentice-Hall, 1974.

[61] IBM OS FORTRAN IV (H Extended) Compiler Programmer's Guide. 3rd Ed. San Jose, California: IBM Corporation, No. SC8-6852-2, 1974.

[62] Brent, R. P. "A Fortran Multiple Precision Arithmetic Package." Association for Computing Machinery Transactions on Mathmatical Software, Vol. 4 (1978), pp. 57-70.

[63] Gantmacher, F. R. The Theory of Matrices, Volume One. New York: Chelsea, 1977.

[64] Jabr, H. A. Modern Analytical Design of Optimal Multivariable Control Systems. Ph.D. dissertation, Polytechnic Institute of New York, Farmingdale, 1975.

APPENDIX A

OPTIMAL CONTROLLER DESIGN THEORY

This appendix summarizes the main results of Youla, Bongiorno, and Jabr [29] and serves only as a reference. Conditions for the existence of an optimal controller C(s) are presented along with sufficient assumptions on the model, as indicated in Figure 1. The procedure for determining the optimal controller is also outlined. The definitions, theorems, and lemmas presented here were obtained directly from Reference [29], and the proofs have been omitted but may be found in the References.

Definitions, Conditions, and Assumptions

Definition 1

The plant P(s) and feedback compensator F(s) form an admissible pair if each is individually free of unstable hidden modes and[1]

$$\psi^{+}_{FP}(s) = \psi^{+}_{F}(s)\ \psi^{+}_{P}(s). \tag{A.1}$$

(The monic polynomials $\psi^{+}(s)$ and $\psi^{-}(s)$ absorb all the zeros of $\psi(s)$ in

---

[1] Let the distinct finite poles of A(s) be denoted by s_i and their associated McMillian degrees by δ_i. The monic polynomial

$$\psi_A(s) = \sum_{i=1}^{u} (s - s_i)^{\delta_i}$$

is the characteristic denominator of A(s). C^{+} denotes the closed right-half of the s-plane and C^{-} denotes the open left-hand of the s-plane.

C^+ and C^-, respectively, and, up to a multiplicative constant, $\psi(s) = \psi^+(s)\ \psi^-(s)$.)

Lemma 1

If the plant P(s), the feedback compensator F(s), and the C(s) are free of unstable hidden modes, the closed-loop of Figure 1 is asymptotically stable if and only if

$$\phi(s) = \frac{\psi_P(s)\ \psi_C(s)\ \psi_F(s)}{\det S(s)} \tag{A.2}$$

is a strict Hurwitz polynomial.

Lemma 2

There exists a controller stabilizing the given plant and feedback compensator in the closed-loop configuration of Figure 1 if and only if the pair P(s), F(s) is admissible.

Lemma 3

Let P(s), F(s) form an admissible pair. Let

$$F(s)\ P(s) = A^{-1}(s)\ B(s) = B_1(s)\ A_1^{-1}(s) \tag{A.3}$$

where the pairs A(s), B(s) and $B_1(s)$, $A_1(s)$ form any left-right coprime polynomial decomposition of F(s) P(s). Select polynomial matrices X(s) and Y(s) such that

$$A(s)\ X(s) + B(s)\ Y(s) = I_n\ . \tag{A.4}$$

Then, (1) the closed-loop of Figure 1 is asymptotically stable if and only if

$$R(s) = (Y(s) + A_1(s)\ K(s))\ A(s), \tag{A.5}$$

where K(s) is any m x n real rational matrix analytic in C^+ and which satisfies the constraint

$$\det (X(s) - B_1(s)\ K(s)) \neq 0 . \tag{A.6}$$

(2) The stabilizing controller associated with a particular choice of admissible K(s) possesses the transfer matrix

$$C(s) = (Y(s) + A_1(s)\ K(s))\ (X(s) - B_1(s)\ K(s))^{-1} . \tag{A.7}$$

If C(s) is defined in this manner, $\phi(s)$ from Equation (A.2) will be strict Hurwitz.

Assumption 1

The plant and feedback compensator form an admissible pair, the feedforward compensator is asymptotically stable, and the transfer matrices P(s), F(s), and L(s) are prescribed in advance.

Assumption 2

$P_o(s)$, $F_o(s)$, $L_o(s)$, $Q(s) = P_s^T(-s)\ P_s(s)$, and spectral densities $G_u(s)$, $G_d(s)$, $G_m(s)$, $G_\ell(s)$ are given. If $P_o(s)$, $F_o(s)$, or $L_o(s)$ represents a physical block, they must be stable. However, if they are merely part of the paper modeling, it is possible to relax stability requirement. The input signal, load disturbance, and measurement noises are stochastically independent.

Assumption 3

Let $P(s) = A_2^{-1}(s)\ B_2(s)$ be any left-coprime factorization of P(s) and let

$$G_{m\ell}(s) = F_o(s)\ G_m(s)\ F_o^T(-s) + L_o(s)\ G_\ell(s)\ L_o^T(-s). \tag{A.8}$$

The matrices $Q(s) = P_s^T(-s)\ P_s(s)$, $F(s)$, $(F(s) - I_n)\ P(s)$, $A_2(s)\ G_u(s)\ A_2^T(-s)$, $A_2(s)\ (P_o(s)\ G_d(s)\ P_o^T(-s))\ A_2^T(-s)$, $L(s)\ G_d(s)\ L^T(-s)$, and $G_{m\ell}(s)$ are analytic on the finite $s = j\omega$ axis.

Assumption 4

Let k be any positive constant,

$$G(s) = G_u(s) + P_d(s)\ G_d(s)\ P_d^T(-s) + G_{m\ell}(s) \tag{A.9}$$

and

$$P_d(s) = F(s)\ P_o(s) + L(s)\ . \tag{A.10}$$

The matrices $A(s)\ G(s)\ A^T(-s)$ and $A_1^T(-s)\ (P^T(-s)\ P(s) + kQ(s))\ A_1(s)$ are nonsingular on the finite $s = j\omega$ axis.

Assumption 5

The data satisfy the order relations[2]

$$G_u(s) \leq O(1/s^2) \tag{A.11a}$$

$$P_o(s)\ G_d(s)\ P_o^T(-s) \leq O(1/s^2) \tag{A.11b}$$

$$G_d(s) \approx s^{-2i}I \tag{A.11c}$$

$$P(s) = O(s^{\nu}) \tag{A.11d}$$

$$O(P) + O(F) \leq \mu \tag{A.11e}$$

---

[2] $A(s) \leq O(s^r)$ means no entry in $A(s)$ grows faster than s^r as $s \to \infty$. The order of $A(s)$ equals r, i.e., $A(s) = O(s^r)$ if (1) $A(s) \leq O(s^r)$, and (2) at least one entry grows exactly like s^r. For $A(s)$ square, $A(s) \approx s^r I$ abbreviates

$$\lim_{s \to \infty} s^{-r} A(s) = A_\infty \quad (A_\infty \text{ constant nonsingular matrix})$$

$A(s) \approx s^r I$ implies $A(s) = O(s^r)$ but not conversely.

and

$$(P^T(-s)\ P(s) + kQ(s))\ G(s) \approx s^2 I_m, \tag{A.11f}$$

where $i \leq l\ \mu \geq \max(\nu - 1, -1)$.

The Optimal Controller

It can be shown that under Assumptions 1 through 5 of the previous section, the optimal K(s), which satisfies Equation (A.6) and makes E finite, can be found in the following manner.

Theorem 1

1. Construct two square real rational matrices $\Lambda(s)$, $\Omega(s)$ analytic together with their inverses in C^+ such that[3]

$$A_l^*(P^* Q_t P - kQ)\ A_l = \Lambda^* \Lambda \tag{A.12}$$

and

$$AGA^* = \Omega\ \Omega^*. \tag{A.13}$$

2. Let

$$= A_l^*\ P^*\ Q_t(G_u + P_o G_d P_d^*)\ A^* \tag{A.14}$$

and choose any two real polynomial matrices X(s), Y(s) such that

$$A(s)\ X(s) + B(s)\ Y(s) = I_n. \tag{A.15}$$

3. The transfer matrix of the optimal controller is given by

$$C = (Y + A_k K)\ (X - B_l K)^{-1} \tag{A.16}$$

---

[3]The indeterminate s will be dropped from subsequent representations where there will be no confusion. $A^*(s) = A^T(-s)$, $A^{-*}(s) = (A^*(s))^{-1}$.

where[4]

$$K = \Lambda^{-1}(\{\Lambda^{-*}\ \Gamma\Lambda^{-*}\}_{+} + \{\Lambda A_1^{-1} Y\Omega\}_{-})\Omega^{-1} - A_1^{-1}\ Y, \tag{A.17}$$

or alternatively

$$C = H_o(A^{-1}\Omega - FPH_o)^{-1}, \tag{A.18}$$

$$H_o = A_1\ \Lambda^{-1}(\{\Lambda^{-*}\ \Gamma\Omega^{-*}\}_{+} + \{\Lambda A_1^{-1} Y\Omega\}_{-})\ . \tag{A.19}$$

The (nonhidden) poles of the optimally compensated loop are precisely the zeros of the strict Hurwitz polynomial

$$\Theta(s) = \frac{\psi_F^{-}(s)\ \psi_P^{-}(s)}{\psi_{FP}(s)} \tag{A.20}$$

plus the finite poles of K(s), each counted according to its McMillian degree.

Corollary 1

Suppose F(s) P(s) is analytic in C^{+}. Then

$$C = H_o(\Omega_r - FPH_o)^{-1} \tag{A.21}$$

where

$$H_o = \Lambda^{-1}\{\Lambda_r^{-*}\ \Gamma\Omega_r^{-*}\}_{+} \tag{A.22}$$

$$(P^{*}Q_tP + kQ) = \Lambda_r^{*}\ \Lambda_r \tag{A.23}$$

$$G = \Omega_r\ \Omega_r^{*} \tag{A.24}$$

---

[4] In the partial fraction expansion $\{\cdot\}_{\infty} + \{\cdot\}_{+} + \{\cdot\}_{-}$ of any rational matrix, $\{\cdot\}_{\infty}$ is the part associated with the pole at infinity and $\{\cdot\}_{+}$, $\{\cdot\}_{-}$ the parts associated with all the finite poles in C^{-} and C^{+}, respectively.

$$\Gamma_r = P^* Q_t (G_u + P_o G_d P_d^*) \tag{A.25}$$

and $\Lambda(s)_r$, $\Omega(s)_r$ are square, real rational matrices analytic together with their inverses in C^+.

Corollary 2

Let

$$a = \Lambda^{-*} \Gamma \Lambda^{-*} \tag{A.26}$$

$$b = \Lambda A_1^{-1} Y \Omega \tag{A.27}$$

$$c = \{a - b\}_- \tag{A.28}$$

and

$$\rho = Q_t (G_u + P_o G_d P_o^*) - a^* a + c^* c \tag{A.29}$$

then the minimum cost E_{min} is given by

$$2\pi j \, E_{min} = \mathrm{Tr} \int_{-j\infty}^{j\infty} \rho(s) ds. \tag{A.30}$$

If P(s) F(s) is analytic in C^+ (stable case), then

$$\rho = Q_t (G_u + P_o G_d P_o^*) - \{a\}_+^* \{a\}_+ . \tag{A.31}$$

Corollary 3

Let P(s) be square and analytic together with its inverse in C^+, let F = I (unity feedback), let k = 0 (no saturation constraint), and assume feedforward compensation is not employed. Then, if G and $Q_t (G_u + P_o G_d P_d^*)$ are diagonal matrices, the optimal controller C(s) satisfies the noninteraction condition

$$P(s) \; C(s) = \text{diagonal matrix.} \tag{A.32}$$

APPENDIX B

PROTOTYPE PROGRAM STRUCTURE

An important achievement resulting from this study was the development of a prototype computer program for frequency-domain synthesis of optimal controllers. This appendix describes the logical structure of the program and its features. The program was coded in FORTRAN on an IBM 370 computer; however, the information presented here can be used to develop similar programs in FORTRAN on different machines.

The process of controller design consists of three parts. Part one is the model preparation; part two is the actual synthesis; and part three is the analysis. Model preparation consists of the process of generating the necessary data for the synthesis using the generalized model representation theory presented in Chapter III. The definition of synthesis and analysis is obvious. Rather than include all three parts in one large program, it was more efficient to develop each part into a separate program. The reason for this was that the model preparation needed only to be performed one time, while the synthesis program would usually be run several times for trade-off studies. The analysis was a separate program due mainly to the fact that a program already existed for frequency-domain analysis [58]. This program had analysis capabilities far beyond any that could be efficiently included directly in the synthesis program.

Figure 20 shows the data flow through the entire controller design process. For the model preparation program, two options were made avail-

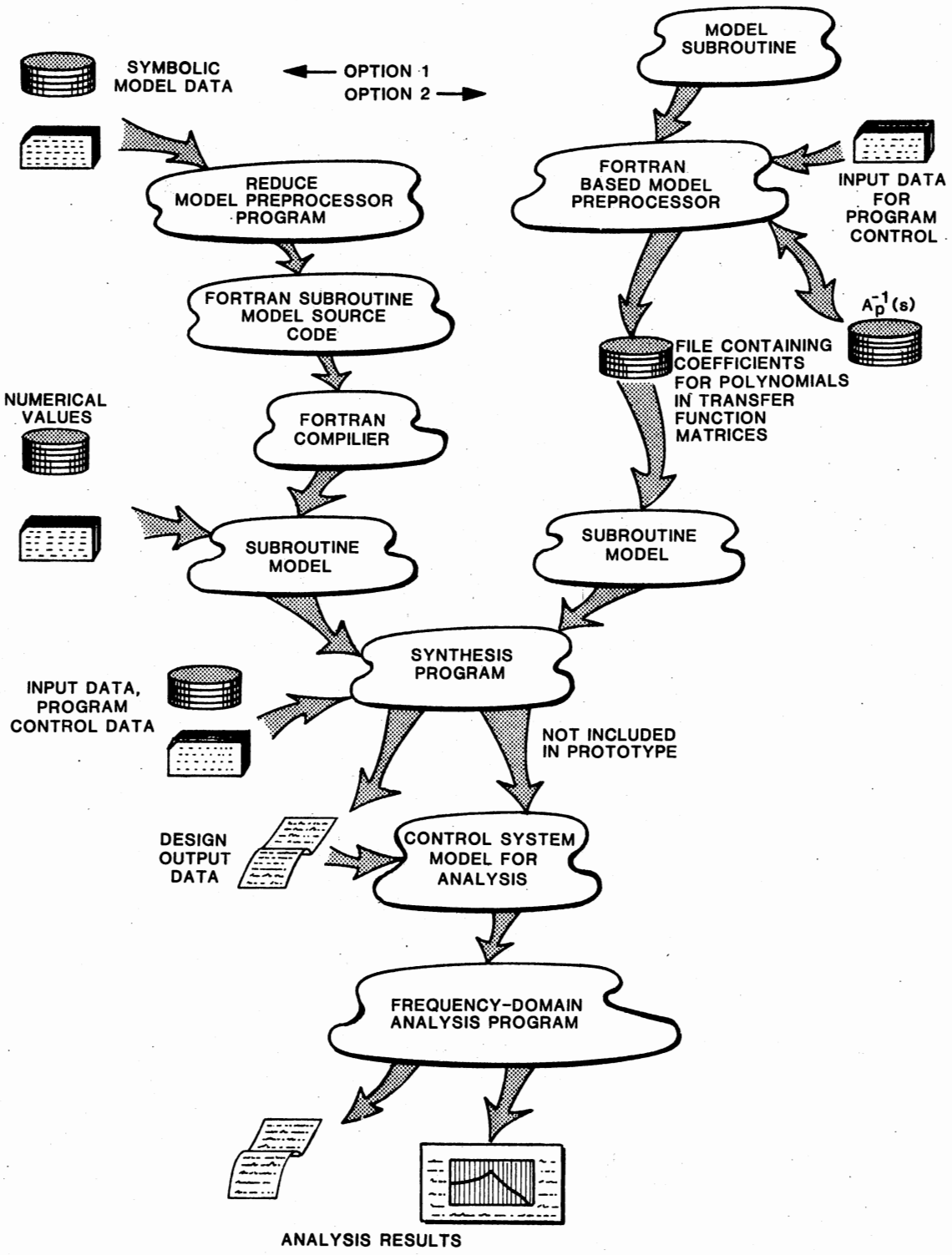

Figure 20. Data Flow Through the Controller Synthesis System

able. The first option consisted of a REDUCE program to which symbolic information describing the matrices $A_p(s)$ and $B_p(s)$ (see Chapter III) was input. Additional information was also input which described the configuration of the plant, feedback measurement system, etc. The REDUCE program inverted the $A_p(s)$ matrix and performed the necessary matrix multiplies to produce the P(s), F(s)P(s), $P_o(s)$, $F_o(s)$, $P_s(s)$, L(s), and $L_o(s)$ plant matrices. These matrices were written in the form of a FORTRAN subroutine (Subroutine MODEL) by REDUCE which could be called by the synthesis program to obtain the required transfer function matrices. The synthesis program required a data file which contained the numerical values for the symbols used to originally define the model. Each matrix was defined in the proper form for the synthesis program, that is, each matrix consisted of a matrix of polynomials and an associated scalar divisor polynomial.

It was also determined that the REDUCE program could be made to calculate additional intermediate matrices needed for the synthesis program. Referring to Appendix A, these additional matrices would be the G(s) matrix of Equation (A.9), the $P_d(s)$ matrix of Equation (A.10), the $P^T(-s)Q_tP(s) + kQ(s)$ matrix of Equation (A.12), and the $P^T(s)Q_tG_u(s) + P_o(s)G_d(s)P^T{}_d(-s)$ part of matrix $\Gamma(s)$ in Equation (A.14). In order for the REDUCE program to generate these matrices, the spectral density matrices $G_u(s)$, $G_d(s)$, $G_m(s)$, and $G_\ell(s)$ had to be defined along with the $A_p(s)$ and $B_p(s)$ matrices.

The second option shown in Figure 20 for the model preparation preprocessor was the use of a FORTRAN-based program. The plant model was coded in the form of a subroutine, which was called by the model preparation program to define the $A_p(s)$ and $B_p(s)$ matrices. Instead of the

polynomials being represented symbolically, the FORTRAN program required that they be represented with actual numeric values for their coefficients. Once the $A_p(s)$ matrix was inverted, it was stored in a disk file. Storing the inverted $A_p(s)$ matrix allows the model preprocessor to retrieve it during subsequent runs for cases where only the plant configuration is changed, thereby avoiding the same inversion over and over. Implementation of this feature in the REDUCE program proved highly inefficient since more computer execution time was required to read the stored $A_p^{-1}(s)$ matrix than was required to do the inversion.

Once the FORTRAN version of the preprocessor inverted the $A_p(s)$ matrix, it would read the configuration data designating desired input-output relationships and would then write a data file which contained the polynomial coefficients of the $P(s)$, $P_o(s)$, $F(s)P(s)$, etc. transfer function matrices. The synthesis program obtained these matrices by calling a special subroutine (Subroutine MODEL) which read the data file to define the desired transfer function matrix.

The selection as to which preprocessor option is best can only be made as later research develops the prototype design system into a more production-oriented system. As for the prototype used in this research, the FORTRAN option was implemented mainly due to the fact that it was more economical to use. The FORTRAN version was also used to study problems centered around rational polynomial matrix inversion.

The logical structure of the FORTRAN model preprocessor and the synthesis program is basically the same. The following discussion on the program structure applies to both the model preprocessor and the synthesis program. Since the analysis program is already well documented (see Reference [58]) it will not be discussed here.

The synthesis program structure is divided into six levels. These levels, from highest to lowest, are named: 1) main level, 2) executive level, 3) general computations level, 4) special matrix operations level, 5) basic matrix arithmetic level, and 6) basic arithmetic and memory management level. Each level (except the main level) consists of a set of subprograms which perform operations at the specified level. In general, two rules apply to the routines. Rule one is the routines in one level may only call routines in levels which are lower or it may call routines in the same level. No routines in one level may call higher level routines. The second rule is that no routine in one level may perform any operation that is available at a lower level. These rules make the overall program very flexible and easily modified.

Polynomial coefficients are stored in contiguous extended precision words of memory ordered from lowest to highest order coefficient. An integer number is stored with each set of polynomial coefficients to specify the number of coefficients. Polynomial matrices are stored as three-dimensional arrays such that the first dimension refers to individual coefficients, the second dimension refers to the rows of the matrix, and the third dimension refers to the columns of the matrix. This scheme keeps the coefficients of any one polynomial together in contiguous storage locations. A two-dimensional, integer matrix is used to store values defining the number of coefficients for each polynomial of the matrix. Scalar polynomial coefficients are stored in one-dimensional arrays with a single integer variable defining the number of coefficients in the array.

Storage for the coefficient arrays is allocated dynamically during program execution by the memory management systems. Dynamic array allo-

cations allow the synthesis program to automatically adjust the size of coefficient arrays during execution to keep the amount of memory required at a minimum. The total memory available for a particular problem is set by the user before the program executes. The program user adjusts the size of an unlabeled common block in the main program (main level). The main program is comprised only of the necessary COMMON statements and a CALL statement which starts the synthesis executive program. By making the executive program a subroutine, only the main program has to be recompiled when the total amount of available storage is changed. The executive subroutine needs never to be recompiled.

Most coefficient arrays are allocated at the executive level. The executive program determines the size requirements for the various coefficient arrays and calls a special subprogram of the memory management system. The subprogram returns a suitable starting location for the array in blank common. When the executive makes calls to routines at lower levels, the starting memory address in common storage of any coefficient array is passed as an argument. The receiving routine refers to the matrix as a three dimensional matrix. The following transaction Illustrates the process:

```
      SUBROUTINE CONTROL
      COMMON / / COMBUF(3000)
         .
         .
         .
         .
      ISTARTA = MEMMAN(IS*N*M)
         .
         .
         .
```

```
CALL DECOMP(COMBUF(ISARTA),IS,N,M ...
   .
   .
   .
END
SUBROUTINE DECOMP(A,ISA,N,M, ...)
DIMENSION A(IS,N,M)
   .
   .
   .
   .
```

Subroutine CONTROL requests the starting location of an array for an nxm polynomial matrix which will contain polynomials with no greater than IS coefficients each. CONTROL then makes calls to various lower level routines as shown. The lower level routine can now easily refer to the individual polynomials in the matrix.

The basic arithmetic level consists of routines which add, subtract, multiply, divide, and compute the GCD of polynomials. The special addition/subtraction routine is also at this level. By keeping basic arithmetic of the same level as the memory management system, the memory management scheme can be modified without having to make coding changes at a higher level. As long as all higher level routines use the basic arithmetic level routines for any necessary polynomial operations, then any memory management scheme can be implemented. Since the basic arithmetic routines are at the same level as the memory management system, they will have to be changed as memory management changes are made.

The executive level controls the sequence in which the controller is computed. The executive calls routines in the specific computation level to compute the controller in the following order:

1. Subroutine MODEL is called to define the transfer function matrices and spectral density matrices needed for the design.

2. The coprime decompositions of Equations (A.3) and (A.4) are computed.

3. The matrix of Equation (A.12) is computed (but not factored). The computation is performed by first evaluating $A_1^* P^*$ then PA_1. This helps insure common factor cancelation between the A_1 and P matrices. The results of the above evaluations are multiplied with Q_t to obtain the term $A_1^* P^* Q_t P A_1$. The other term of the equation is evaluated in a similar manner to obtain $kA_1^* P_s^* P_s A_1$. These two matrix terms are then added to form the final result.

4. The matrix computed in Step 3 is spectrally factored to obtain $\Lambda(s)$. $\Lambda^{-1}(s)$ is computed in this step.

5. The matrix of Equation (A.13) is computed by first computing each of the following matrix terms:

$$AF_o G_m F_o^* A^*$$

$$AL_o G_\ell L_o^* A^*$$

$$AG_u A^*$$

$$A[FP_o+L][FP_o+L]^* A^*.$$

These terms are then added together to form the required matrix.

6. The spectral factorization is performed to obtain matrix $\Omega(s)$. $\Omega^{-1}(s)$ is also computed in this step.

7. Matrix $\Gamma(s)$ of Equation (A.14) is computed in a manner similar to that used in Step 4.

8. Matrix $H_o(s)$ is computed according to Equation (A.19).

9. The controller is computed according to Equation (A.18).

Each specific computation routine makes extensive use of lower level

routines to do the required scalar and matrix polynomial operations. The specific computation routines write all their intermediate results to the program output listing so that the synthesis process can be monitored by the user. Special routines are provided at the basic arithmetic level to output both rational polynomial matrices and polynomial matrices.

2

VITA

John Edward Perrault, Jr.

Candidate for the Degree of

Doctor of Philosophy

Thesis: COMPUTER IMPLEMENTATION OF OPTIMAL MULTIVARIABLE CONTROLLER DESIGN IN THE FREQUENCY DOMAIN

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born in Tulsa, Oklahoma, March 10, 1953, son of John E. and Barbara C. Perrault.

Education: Graduated from Bishop Kelley High School, Tulsa, Oklahoma, in June, 1971; received the Bachelor of Science in Mechanical Engineering degree from the University of Tulsa, Tulsa, Oklahoma, in June, 1975; received the Master of Science degree from Oklahoma State University, Stillwater, Oklahoma, in May, 1977; completed the requirements for the Doctor of Philosophy degree at Oklahoma State University in May, 1981.

Professional Experience: Staff Engineer, Marvel Photo Company, Tulsa, Oklahoma, 1971-75; Graduate Research Assistant, School of Mechanical and Aerospace Engineering, Oklahoma State University, September, 1975-March, 1981; joined the technical staff of Applied Technology Associates, Albuquerque, New Mexico, in March, 1981.