COMPUTER-AIDED DESIGN OF FEEDBACK CONTROL SYSTEMS WITH PLANT PARAMETER VARIATIONS

BY

HEYDON ZANE LEWIS

This research was sponsored by the
National Aeronautics and Space Administration
under research grant NGR 06-003-083

Department of Electrical Engineering
University of Colorado
Boulder, Colorado

June 30, 1970

COMPUTER-AIDED DESIGN OF FEEDBACK CONTROL SYSTEMS WITH PLANT PARAMETER VARIATIONS

by

Heydon Zane Lewis

This research was sponsored by the
National Aeronautics and Space Administration
under research grant NGR 06-003-083

Heydon Zane Lewis

Isaac M. Horowitz
Research Supervisor

Department of Electrical Engineering
University of Colorado
Boulder, Colorado

June 30, 1970

TABLE OF CONTENTS

COMPUTER-AIDED DESIGN OF FEEDBACK CONTROL AND SYSTEMS WITH PLANT PARAMETER VARIATIONS

Abstract--This research is devoted to two applications of computer methods to feedback control system design. In the first application, gradient optimization methods are used for the automatic design of a fixed plant system, to satisfy time domain specifications. Given a fixed system configuration, the program manipulates open-loop compensation parameters. A simulation routine provides values of overshoot, rise time, and high frequency transmission, from which the gradient vector is obtained for minimization of the high frequency loop transmission. The time domain specifications appear as inequality constraints. The work done indicates this to be a possible, though not as yet practical, method of design. Practical problems of step-size selection and constraint interaction have not been completely solved for a wide class of problems.

The second application extends recent methods for the design of adaptive control systems with variable plant parameters. A significant improvement is the removal of the requirement of a dominant second-order response, heretofore used in one class of current design methods. Again, the computer simulation provides the relationship between open-loop parameter values and step response data. The results are presented in the plant-pole parameter subspace. Curves of constant step

response, rise time and overshoot are obtained and these define a region over which the parameters may vary with acceptable response. This is followed by manual perturbation of the parameters, with the human designer evaluating the resulting changes in constant performance curves.

The above approach has readily permitted the insertion of a transmission zero into the closed-loop system transfer function, either fixed or as a drifting plant zero.

ACKNOWLEDGMENTS

I would first like to express my thanks to Dr. Isaac M. Horowitz for his direction of this research and to Dr. Harry F. Jordan for his helpful suggestions in the computer methods. My graduate work has been made possible by support from the Department of Electrical Engineering, University of Colorado, and from the National Aeronautics Space Administration under Contract NGR 06-003-083.

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.0 Objectives

The basic objective of this investigation is to explore ways to improve feedback control design by means of digital computer techniques. Current design methods are studied at the conceptual level to determine whether reformulation for computer usage is necessary or desirable. Promising methods are examined at the practical level in order to identify advantages and disadvantages of using the computer.

The main area of interest is the linear control system whose plant has slowly varying parameters. A primary objective is to remove restrictions currently imposed by the human designer.

1.1 Computer Advantages to be Exploited

The digital computer cannot think and cannot improve on the intelligence and insight of the person creating its programs. The usefulness of the computer lies in its ability to make tedious, repetitious calculations quickly, filter and condense the raw data to meaningful information, and present this in an organized fashion. This allows the human more time to think and to try many combinations of design tradeoffs.

The computer can store and manipulate quantities of data, usually without error. It can make specific value decisions based on this data, in a repeatable

fashion, without bias from external factors.

This investigation will not consider numerical analysis questions in any depth. However, it is pertinent to mention why special methods of analysis have been developed for numerical work and what implications this has for the control system problem. Consider the problem of finding the value of a determinant. A human will visually note locations of zeros and perform operations on rows and columns to create more zeros. Eventually he will break it down into minor determinants. The human does this to avoid tedious arithmetic operations. But the computer can do the latter more easily than it can keep track of the process of reduction into minor determinants. Thus a simple but repetitious procedure is often more suitable for computer work.

A specific numerical problem encountered in this investigation is that of obtaining the step response of a control system. The system is described by a set of simultaneous differential equations which can be reduced to a transfer function. A human would probably find the inverse transform from a suitable table. Residue calculations require the evaluation of real and imaginary par. of complex functions. These are algebraic operations which the computer does not perform easily. On the other hand, a time history of the step response can be obtained by successive extrapolation of the derivatives. This involves only arithmetic

operations and is much better suited for computer solution.

The point being made is that many of the control system design procedures currently in use were designed for human solution. One must carefully consider if these methods are also suitable for computer solution. To successfully use the computer to assist in design, one must capitalize on its numerical advantages and not force it to perform in the image of the human.

1.2 Definition of Problems

Problem 1

The first problem is the adaptation of gradient techniques to automatic design of a control system with fixed plant parameters with the time domain performance specifications of overshoot and rise time treated as inequality constraints. The objective is to achieve the latter at a minimum value of the open loop transmission at a high frequency (in order to minimize sensor noise effect).

Problem 2

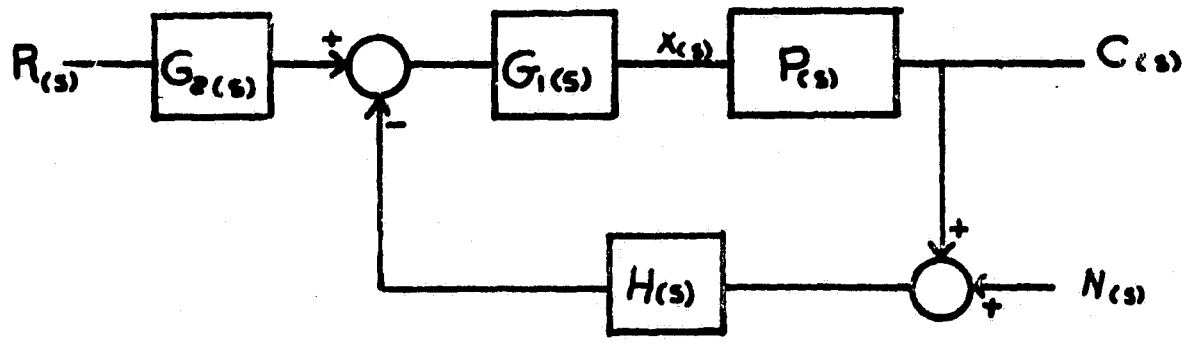
The second area of investigation covers design techniques for control systems whose plant parameters vary slowly over a wide range. Several recent[3,4] publications have presented workable methods which are, however, restricted to dominant second-order response functions. Computer evaluation of the step response removes this restriction. The design is made directly

to time response specifications. The design procedure involves a perturbation process supervised by the human designer and data presentation consistent with the method. It is applicable to systems with zeros in the closed loop response, either in a pre-filter outside the loop or in the open loop transmission. In the latter case, the zero may also be a variable plant parameter.

1.3 Definition of Control System Nomenclature

The block diagram of the control system is shown in Figure 1.1. In some cases $G_2(s)$ or $H(s)$ may be unity. The open loop transmission $L(s)$, the closed loop transmission $T(s)$, and the noise transmission $M(s)$ are also defined. Reference will be made to the open loop transmission at a high frequency (1000 rps) denoted GFAR. This frequency is well beyond all the corner frequencies of the loop transmission.

The time response parameters of overshoot, rise time, and settling time are shown in Figure 1.2. When $T(s)$ is third order with a real pole approximately equal to the real part of the complex pair, then the situation shown in Figure 1.3 arises. The derivative of $c_{(t)}$ becomes zero before $c_{(t)}$ crosses the final value line. This type of response is defined as having a "low overshoot." A slight change in the roots of $T(s)$ causes the response to change from curve A to curve B. However, there is a discontinuous change in the rise time



1. Open Loop Transmission: $L(s) = G_1(s)\ P(s)\ H(s)$

2. Closed Loop Transmission: $T(s) = \frac{C(s)}{R(s)}$

$$T(s) = \frac{G_2(s)\ G_1(s)\ P(s)}{1+G_1(s)\ P(s)\ H(s)} = \frac{G_2(s)\ G_1(s)\ P(s)}{1+L(s)}$$

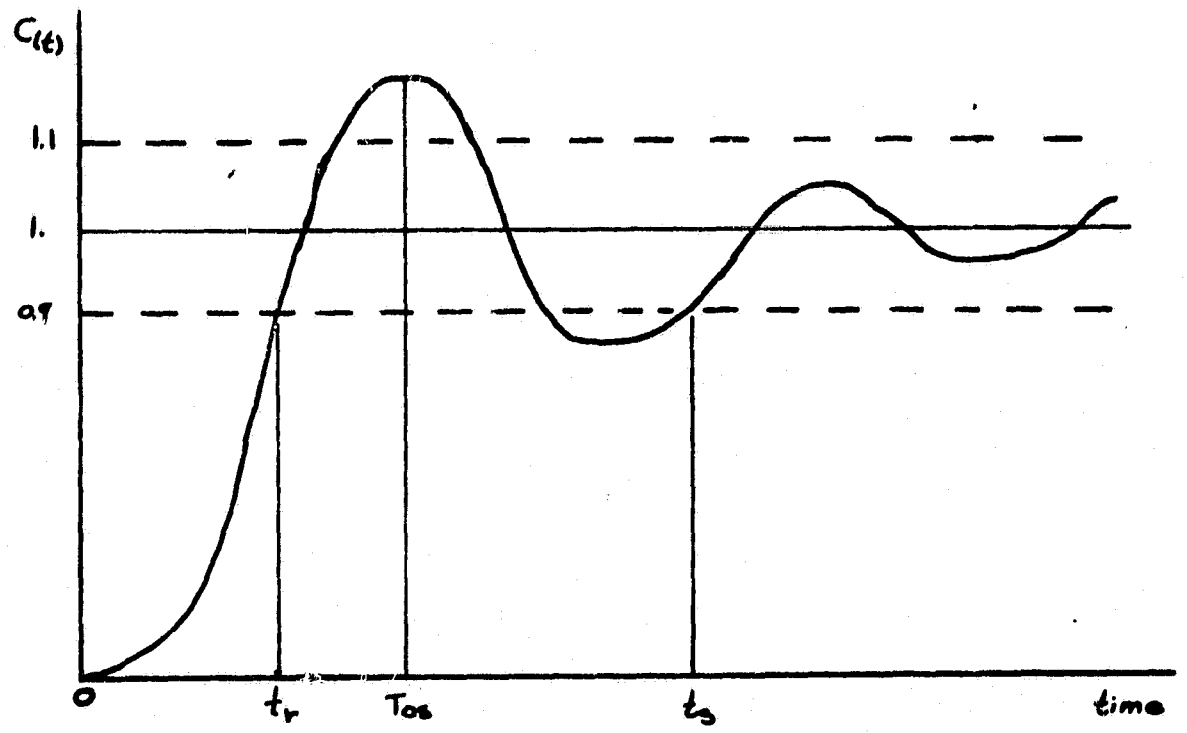
3. Noise Transmission to Plant Input: $M(s) = \frac{X(s)}{N(s)}$

$$M(s) = \frac{H(s)\ G_1(s)}{1+G_1(s)\ P(s)\ H(s)} = \frac{H(s)\ G_1(s)}{1+L(s)}$$

4. High Frequency Transmission: $GFAR = |L(s)|_{s = j1000}$

Figure 1.1

Control System Configuration and
Transmission Definitions



Overshoot = $(C(T_{os}) - 1) * 100.$

Rise time t_r = time to reach 90% of final value

Settling Time t_s = time required to enter a band ±10% of final value and remain inside

Figure 1.2

Definition of Time Domain Performance Specifications

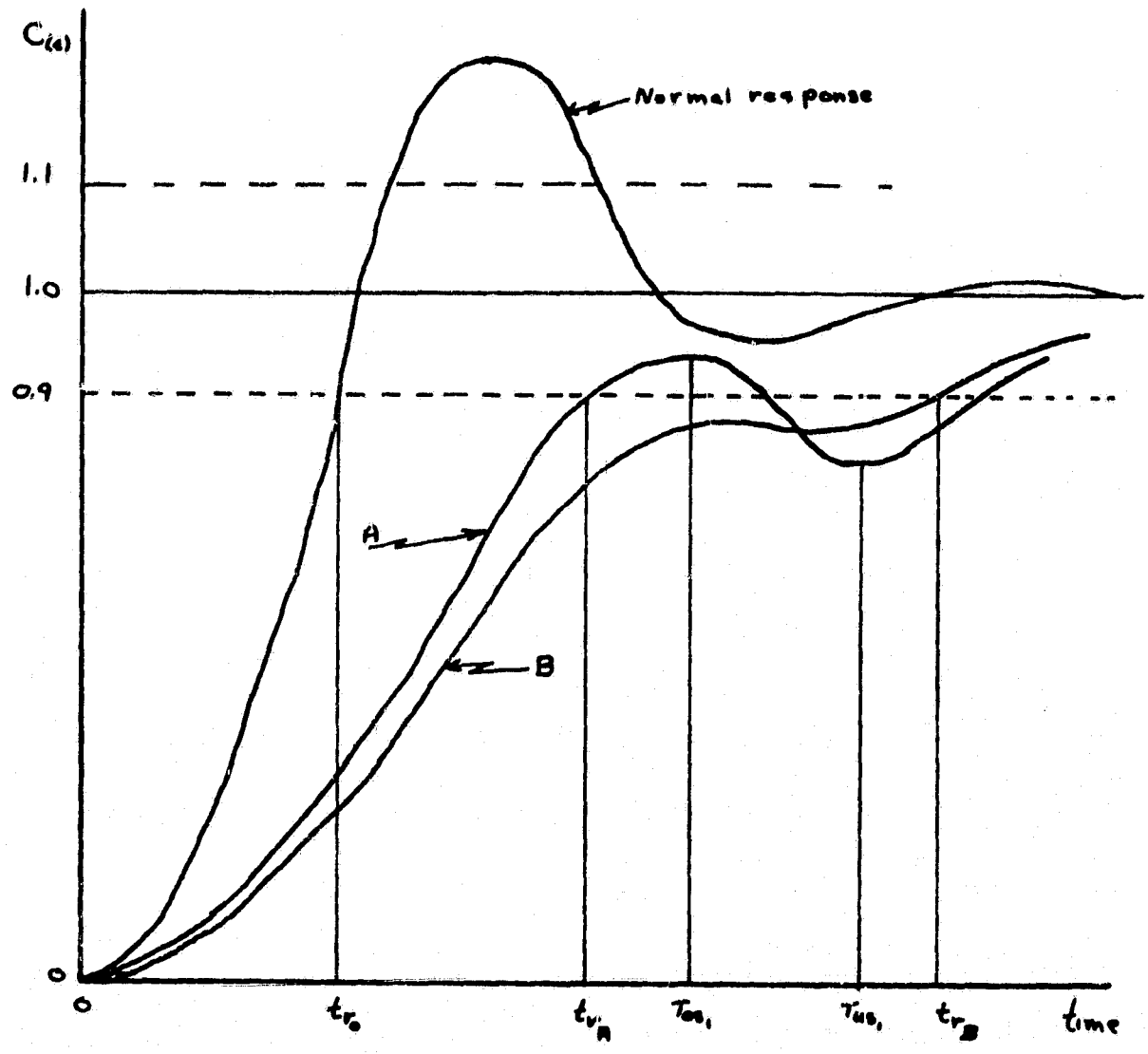


Figure 1.3

Definition of System Response with "Low Overshoot"

from tr_A to tr_B. Low overshoot may also occur when a high frequency sinusoid of small amplitude is added to the normal sinusoid present in an underdamped response.

CHAPTER 2

COMPUTER SIMULATION TECHNIQUES FOR CONTROL SYSTEM TIME RESPONSE

2.0 Introduction

This chapter describes the procedures for finding the overshoot, rise time, and settling time of the system step response. Some earlier synthesis methods have used analytic expressions for overshoot, etc., in terms of the system parameters. This restricted the system order to third or less. An important objective of this project was to extend this order to at least the fourth. In addition, many of the previous methods used only the closed loop roots to characterize the system. It is desirable that the open loop poles, zeros, and gain also be allowed.

A package of subroutines called RUN is used to provide the computer simulation of the system. It accepts an array of parameter values. These parameters are: open-loop poles, zeros, and gain. RUN generates the step response and returns values for overshoot, rise time, and settling time. It also provides data from the frequency response of the open loop transmission L(s). Other results such as integral squared error can be provided, if desired.

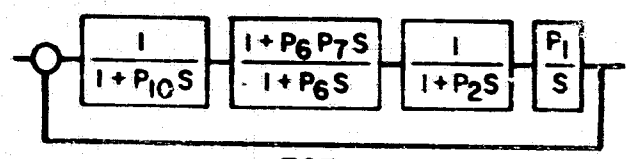
RUN was originally designed for a hybrid computer and later changed to an all digital simulation. Only the latter is described.

2.1 The Input Parameter Space

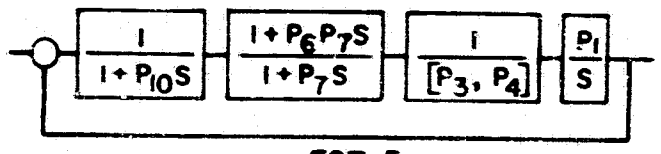
The input parameters to RUN are open-loop poles and zeros and gain. These open-loop parameters were chosen even though the closed-loop parameters are fewer in number. The reason is that after the design is effected in the closed-loop domain, the corresponding open-loop values must be found for design implementation. Since it is necessary to examine the effect of open-loop plant parameter variations, it is desirable to have direct access to these variables.

Choice of the open-loop parameter space does introduce complications. A specific compensation configuration must be chosen before proceeding with the design. There are a number of combinations of types of plant poles and zeros, types of compensation poles and zeros, and the locations of the compensation poles and zeros in the control loop. Figure 2.1 shows a number of the configurations used in this study.

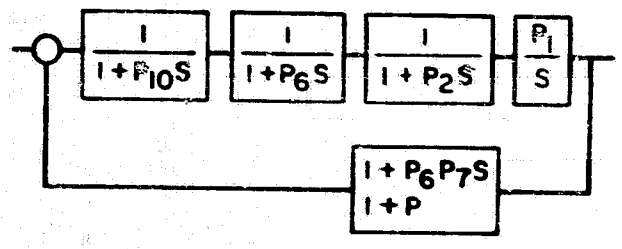
A pole written as $\frac{1}{s+P}$ is defined as being in root locus form and a pole written as $\frac{1}{1+\gamma s}$ is defined as being in Bode form. Most of the possible configurations were written using the Bode form. This allows several real poles (or zeros) to be included. Those which may not be needed are eliminated by setting the approximate γ to zero. This reduces the number of required configurations. A routine called FRESP2 calculates GFAR previously defined.



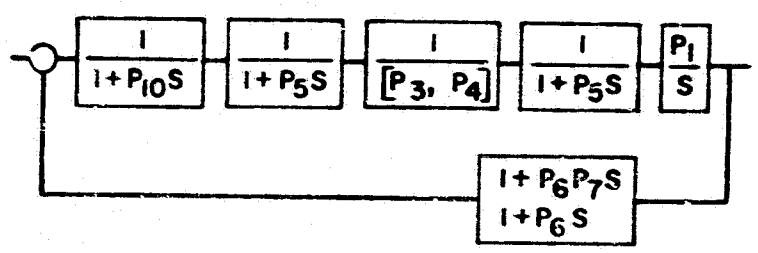
FCT 1



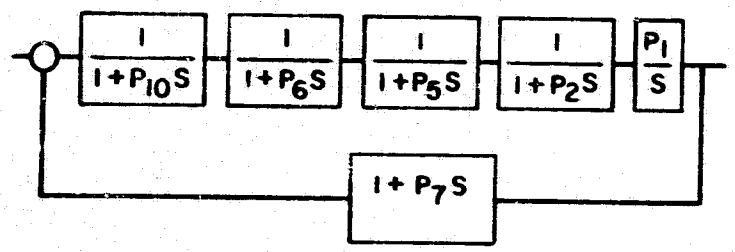
FCT 5



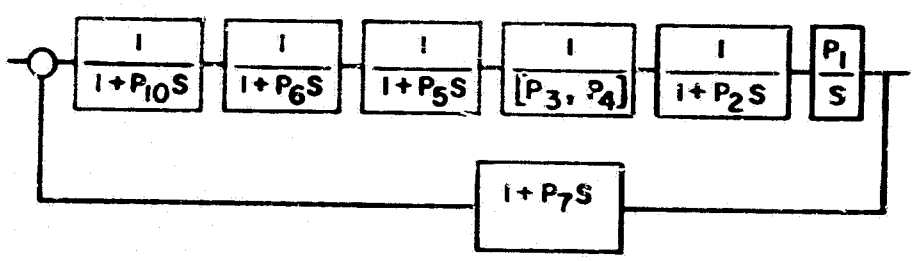
FCT 11



FCT 9

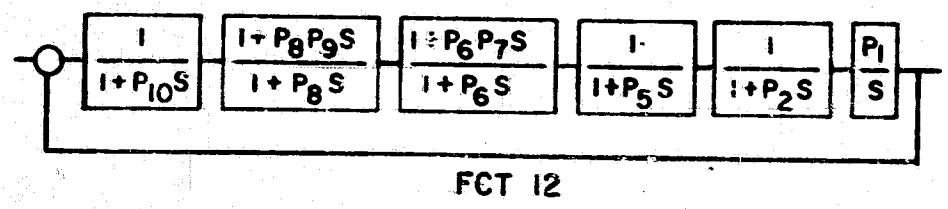


FCT 10



FCT 7

REAL AND COMPLEX POLE PLANT
REAL COMPENSATION



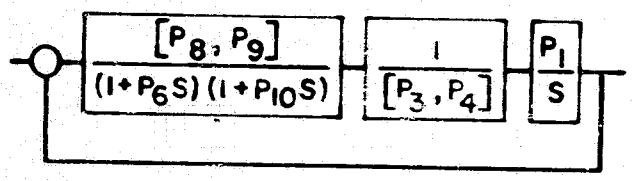
FCT 12

REAL PLANT - REAL COMPENSATION

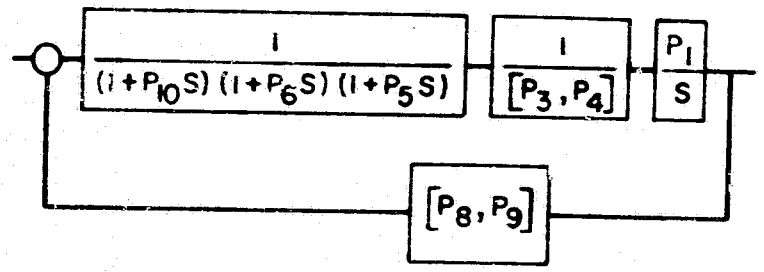
NOTE:

NOMENCLATURE $[\alpha\ \beta]$ DENOTES A SECOND ORDER POLYNOMIAL OF THE FORM $S^2+2ds+d^2+\beta^2/d^2+\beta^2$

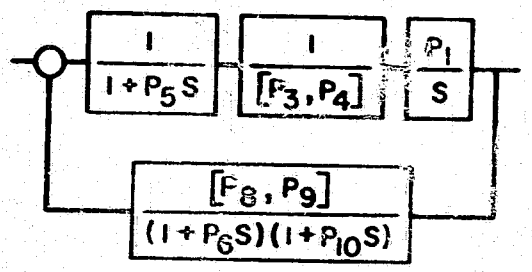
Figure 2.1a



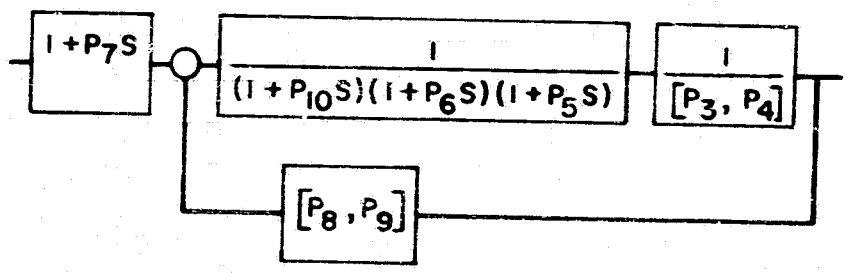
FCT 3



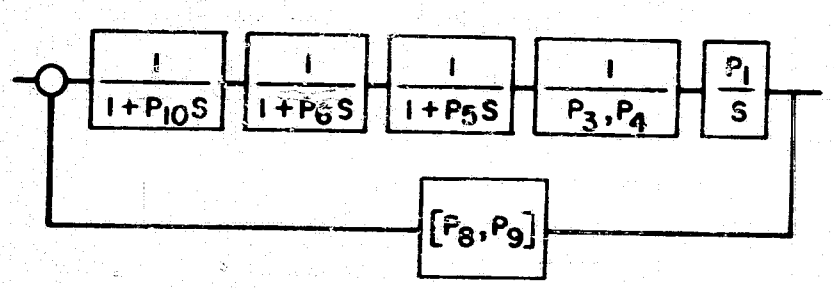
FCT 13



FCT 6

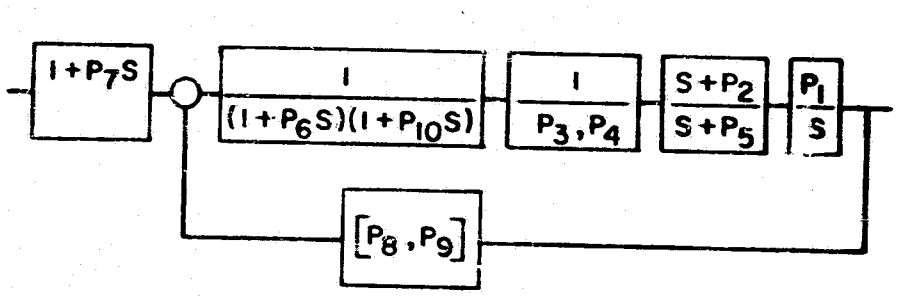


FCT 18



FCT 8

REAL AND COMPLEX PLANT



FCT 20

REAL AND COMPLEX PLANT
COMPLEX POLES IN ROOT LOCUS FORM

COMPLEX ZEROS - REAL POLE COMPENSATION

Figure 2.1b

The bandwidth is obtained by the subroutine FRIT using an interval halving technique. FRIT evaluates $L_{(j\omega)}$ at two values of ω (.01 and 100.) using FRESP2. It checks to see that $|L_{(j\omega)}|$ equal to .707 is included in this interval. It extends the interval if necessary. When the interval is established, it is halved twenty times and the value of ω returned as the bandwidth.

2.2 Time Response Simulation

The QD017 program described later uses poles and zeros of a transfer function and generates the time history via partial fraction expansion and inverse LaPlace transformation. It forms the closed-loop transfer function from the open-loop poles, zeros, and gain. This method has the advantage that many poles and zeros may be used in the open-loop system. Those poles that become "far-off" in the closed loop have small residues and therefore do not degrade the accuracy of the time response or affect running time. This is not the case in a numerical solution of the differential equations. However, the QD017 program is large and requires too much computing time for use as the inner loop of iterative calculations. It was decided not to attempt a modification.

After deciding on the numerical solution, there arose the question whether or not to use a simulation language. Several programs[7,21] were available. However, they either had no capability for linking to Fortran subroutines and for being called by other

programs, or there was too much overhead invested in flexibility which was not required for this problem. Simulation languages as such were ruled out.

More questions arose regarding the method of numerical solution. First, one must choose between solving the differential equations directly or converting them to equivalent difference equations by z-transforms. The latter[8,9] is reputed to be quite fast. Details of this method were not available, so the direct solution approach was taken.

The first method tried used the Hamming variable-step predictor-corrector numerical solution procedure[25,10]. The results for overshoot, rise time, etc., were compared with results from the analog computer and the QD017 program. The check was quite satisfactory. However, a single time response required four seconds of computation time to give the same time history produced by the analog in two seconds. The difficulty was traced to the small increment used in the independent variable. This was necessary to have a small quantization size for rise time, settle time. With such a small step-size, the variable step adjustment was never used and a simpler numerical solution algorithm should be chosen.

A second-order Runge-Kutta with fixed step-size and no error checks was selected. The results again checked quite satisfactorily. A series of modifications

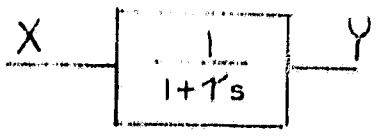
to the programming, described later, brought the computing time required for one time history down to 0.2 seconds. This is for a FCT1 configuration with five differential equations to be solved.

2.3 Generation of System Differential Equations

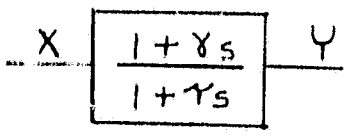
The system differential equation is generated from the transfer function. Standard blocks consisting of a real pole, a real pole-zero pair, a complex pole pair, and a complex zero pair with two real poles, were defined.

The general procedure is to reverse the process by which one obtains a transfer function from the differential equation. For instance, with the real pole block with transfer function, $\frac{X_{out}}{X_{in}} = \frac{1}{1+\gamma s}$, cross multiplication gives $(1+\gamma s)X_{out} = X_{in}$ which becomes $\gamma \dot{X}_{out} = X_{in} - X_{out}$. This procedure is used for all cases except the complex zero and the relations are given in Figure 2.2. When this procedure is applied to the complex zero block, a requirement for the second derivative of the input arises. This is not available generally. For this case, the standard procedure for simulation of transfer functions on an analog computer was used to develop a computer diagram. The differential equations were written from this diagram. The complete block diagram with equations and check solutions for FCT6 is shown in Figure 2.4.

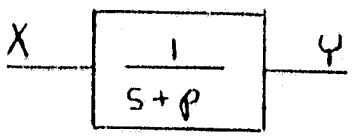
Note that the inclusion of a zero requires the



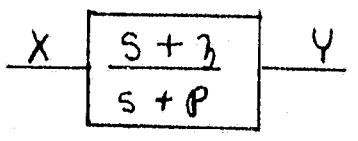
$$\dot{y} = \frac{1}{\tau}(x-y)$$



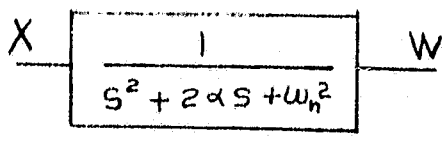
$$\dot{y} = \frac{1}{\tau}(x-y) + \gamma\dot{x}$$



$$\dot{y} = x - \beta y$$

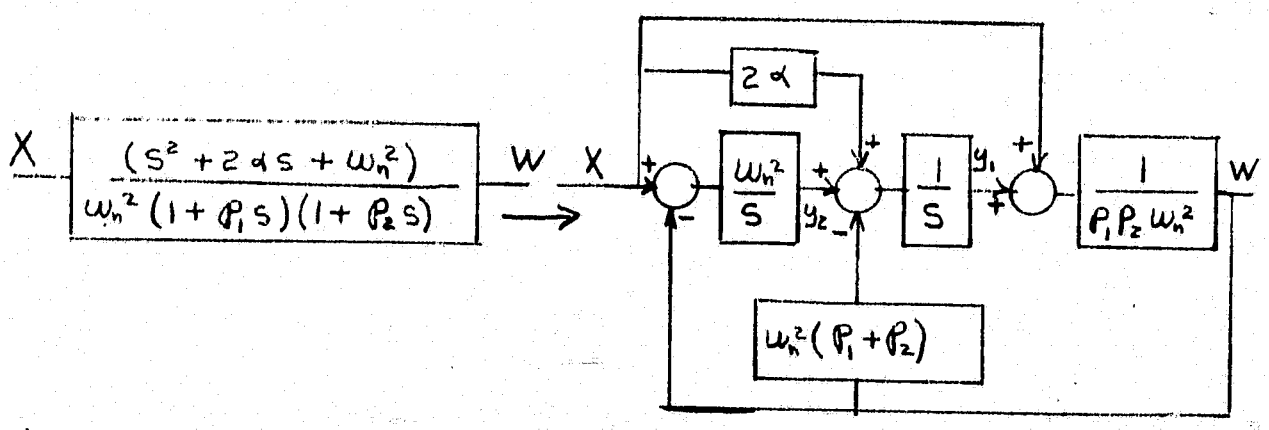


$$\dot{y} = \zeta x + \dot{x} - \beta y$$



$$y_1 = w \qquad y_2 = \dot{w}$$

$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = x - 2\alpha y_2 - \omega_n^2 y_1$$



$$w = \frac{(x+y_1)}{\beta_1\beta_2\omega_n^2}$$

$$\dot{y}_1 = y_2 + 2\alpha x - \omega_n^2(\beta_1+\beta_2)w$$

$$\dot{y}_2 = \omega_n^2(x-w)$$

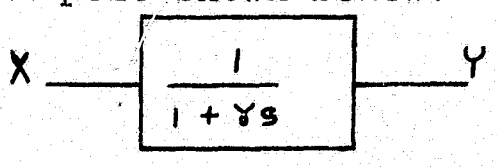
Figure 2.2
Differential Equations for Standard Transfer Functions

derivative of the input state variable. This can generally be made available by proper ordering of blocks. In writing the equations for the derivatives in Fortran, one must be careful to order the equations so as to use state information at the same point in time.

The question has been raised why the procedure for deriving the derivative equations cannot be automated and reduced to a matrix formulation. At this level, an algorithm could be produced to let the computer generate its own equations. There are, however, several problems. First, as just cited, when zeros are present, the ordering of equations is important to prevent time skew of state information. It is simple for a human to rearrange the equations but difficult to reduce to an algorithm. Furthermore, checkout of an automatic procedure would be difficult. The time skew may make the answers only slightly incorrect and difficult to separate from ordinary numerical errors. No workable schemes were found for automatically generating equations from the poles and zeros or expanding equations to add a new pole or zero.

2.4 Modification to Derivative Defining Equation

A typical element of a control system is a first-order lag or real pole shown below.



A control system may have several of these. It is inconvenient to define a different configuration each time a new one is added, so one would like to define a configuration with as many as normally desired. The unwanted poles may be eliminated by setting the proper parameter (γ in the above) to zero. However, the differential equation for this block is $\dot{y} = \frac{1}{\gamma}[x-y]$. It will not do to set γ equal to zero. In fact, one does not even want γ to be in the same order of magnitude as the step-size of the numerical solution routine.

A solution was found by modifying the defining equation for the derivative. Consider this equation before passing to the limit.

$$\frac{y(t+\Delta t) - y(t)}{\Delta t} = \frac{1}{\gamma}\left[x(t) - y(t)\right] \qquad (2.4.1)$$

or

$$\gamma\, y(t+\Delta t) = \gamma\, y(t) - (\Delta t)\, y(t) + (\Delta t)\, x(t) \qquad (2.4.2)$$

Now consider the equation

$$\frac{y(t+\Delta t) - y(t)}{\Delta t} = \frac{1}{\gamma}\left[x(t) - \left(\frac{y(t) + y(t+\Delta t)}{2}\right)\right] \qquad (2.4.3)$$

or

$$\gamma\, y(t+\Delta t) = \gamma\, y(t) - \frac{\Delta t}{2}\, y(t) - \frac{\Delta t}{2}\, y(t+\Delta t) + \Delta t\, x(t) \qquad (2.4.4)$$

In this equation, the derivative is calculated from the average of y at t and at (t+Δt), instead of just y at t. Putting equation (2.5.4) back into the form of the derivative definition gives

$$\frac{y(t+\Delta t) - y(t)}{\Delta t} = \frac{1}{\left[\gamma + \frac{\Delta t}{2}\right]}\left[x(t) - y(t)\right] \qquad (2.4.5)$$

The derivative equations used by the numerical solution routine are written in this form.

The physical interpretation of this problem is quite simple. When $1/\gamma$ becomes large, this means the difference between x and y will be small. Since $1/\gamma$ represents the pole location, this agrees with intuition. When this occurs in analog computer programming, the solution is to replace the differential equation with the algebraic equation x = y. It would be difficult to make this substitution in the equations as the program operates. It will be seen in the gradient program that poles may be driven quite far out. This modification causes the effective value of the pole to become asymptotic to $\frac{2}{\Delta t}$ as the pole moves out to infinity.

The results of the time response with this modification and without it were compared with results using the QD017 program. There is a slight difference, particularly in overshoot. But the accuracy is well within the normal knowledge of system parameters. The small inaccuracy is a very small price to pay for the convenience of the modification. One must be careful to note that this was used with a fixed step-size integration routine.

2.5 Logic for Selection of Overshoot, Rise Time

The numerical procedure described so far takes the input parameters of open-loop poles, zeros, and gain, plus a definition of the block diagram configuration and returns a complete time history of the step response. The complete time history is not really needed;

only values for overshoot, rise time, and settle time are desired. In the course of using the program, additional values are returned to compensate for the fact that one does not actually see the response. For instance, a divergent system could have a reasonable value for rise time and overshoot. Additional logic was incorporated into the program to watch for this.

The desired results must be stated in a form suitable for the computer. Overshoot is defined as occuring the first time when the present value of the output $c_{(t)}$ is less than its preceding value, and $c_{(t)}$ is greater than the final value. Rise time is defined as occurring when $c_{(t)}$ becomes greater than 90% of final value. Settling time t_s is defined as the smallest value of time such that $0.90c_f \leq c_{(t)} \leq 1.10c_f$.

Additional results were returned to indicate when a response has any undesirable characteristic. The maximum value for the entire time history was saved. If this maximum was greater than the value at first overshoot, the response was denoted as divergent and a flag set. The situation called "low overshoot," defined in Figure 1.3, required several readings to describe it. The values denoted as C_{max1} and C_{min1} defined the amount of the low overshoot wiggle. The per cent low overshoot was defined as $\frac{C_{max1} - C_{min1}}{C_{max1}} \times 100$. The instants of occurrence of C_{max1} and C_{min1} (t_{os1}, t_{us1}, respectively) were also saved.

The flow chart for the subroutine RUN is given in Figure 2.3. The three flags J1, J2, and J3 are used to control rise time selection, first overshoot selection, and low overshoot selection, respectively. X is the output variable of the system. The routine SETUP is described later and calculates some constants. The logic rejects wiggles occurring when $c_{(t)}$ is less than 0.1, as not significant.

2.6 Results and Test Cases

As each of the new configurations was defined and differential equations written for it, the accuracy was checked by comparing the results against two independent programs. The equations were checked with MIMIC using variable step-size integration. The values of overshoot, etc., for the check case values of poles, etc., were determined by the QD017 program using inverse LaPlace transformation. All of the various configurations checked out satisfactorily. Results for configurations FCT6 and FCT20 are given as representative in Figures 2.4 and 2.5. Note that FCT6 uses the Bode representation for poles, zeros while FCT20 uses a mixture of root locus and Bode representations.

2.7 Numerical Stability Considerations

While it is somewhat dangerous to use a low-order numerical solution routine with no error tests, no particular problems were noted. In the course of running some of the test cases, certain ones were found

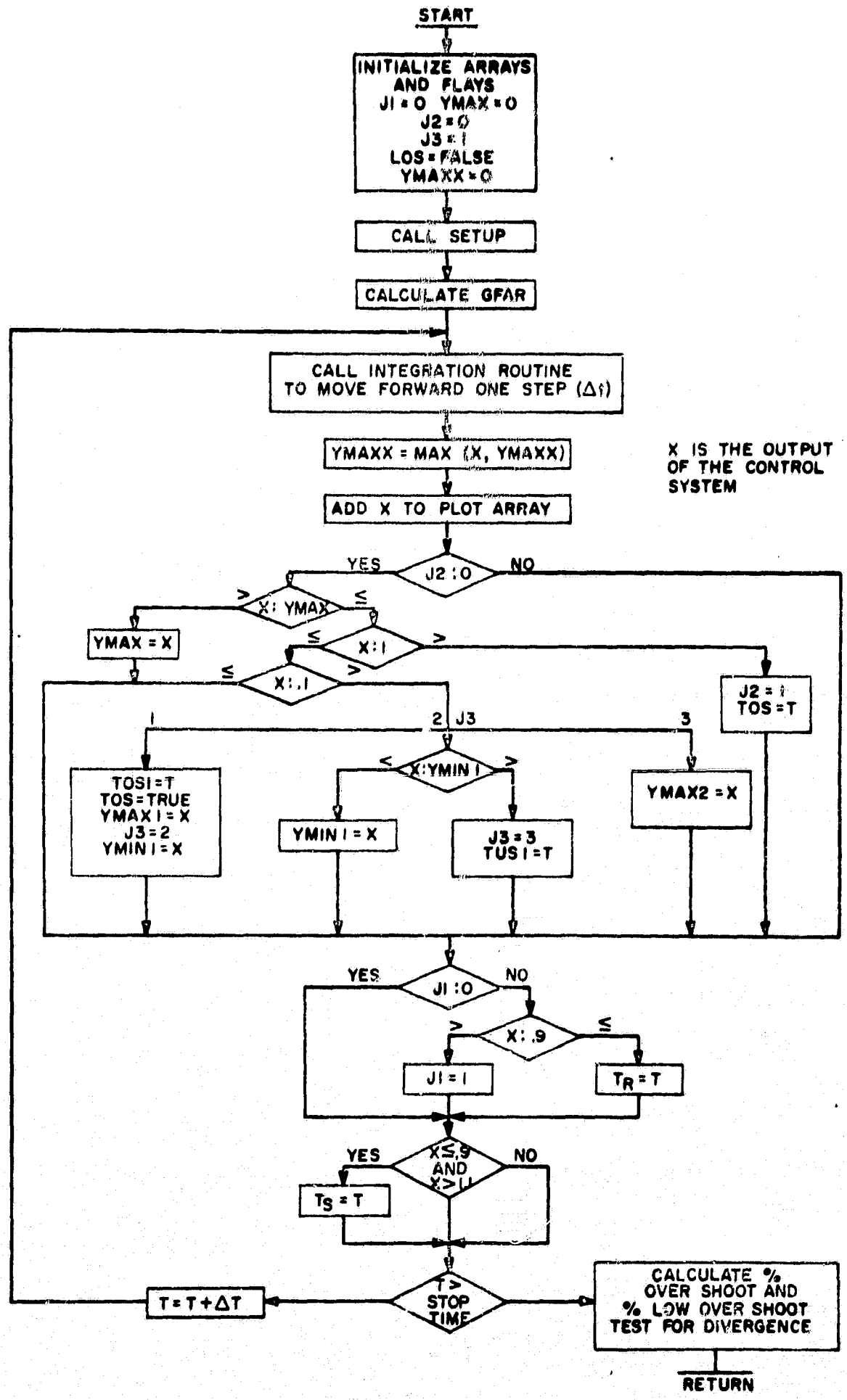
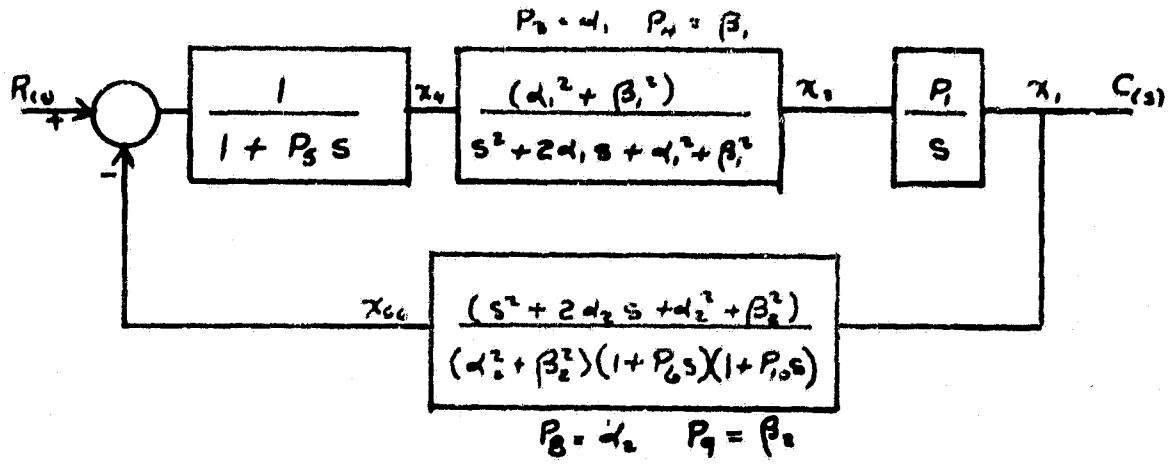

START
INITIALIZE ARRAYS AND FLAYS
J1 = 0 YMAX = 0
J2 = 0
J3 = 1
LOS = FALSE
YMAXX = 0
CALL SETUP
CALCULATE GFAR
CALL INTEGRATION ROUTINE TO MOVE FORWARD ONE STEP (Δt)
YMAXX = MAX (X, YMAXX)
ADD X TO PLOT ARRAY
X IS THE OUTPUT OF THE CONTROL SYSTEM
YES
J2 : 0
NO
>
X : YMAX
≤
YMAX = X
≤
X : 1
>
≤
X : .1
>
J2 = 1
TOS = T
1
2 J3
3
TOSI = T
TOS = TRUE
YMAX I = X
J3 = 2
YMIN I = X
<
X : YMIN I
>
YMIN I = X
J3 = 3
TUS I = T
YMAX2 = X
YES
J1 : 0
NO
>
X : .9
≤
J1 = 1
TR = T
YES
X ≤ .9 AND X > .1
NO
TS = T
T >
STOP TIME
T = T + ΔT
CALCULATE % OVER SHOOT AND % LOW OVER SHOOT
TEST FOR DIVERGENCE
RETURN

Figure 2.3



$$\dot{x}_1 = P_1 x_2$$
$$\dot{x}_2 = x_3$$
$$\dot{x}_3 = x_4 - 2 P_3 x_3 - \omega_{22} x_2$$
$$\dot{x}_4 = (1 - x_{66} - x_4)/P_5$$
$$\dot{x}_5 = \omega_{12}(x_1 - x_{66})$$
$$\dot{x}_6 = x_5 + 2 P_8 x_1 - \omega_{12}(P_6 + P_{10}) x_{66}$$
$$x_{66} = (x_6 + x_1)/(P_6 P_{10} \omega_{12})$$
$$\omega_{12} = P_8^2 + P_9^2$$
$$\omega_{22} = P_3^2 + P_4^2$$

Test Case:

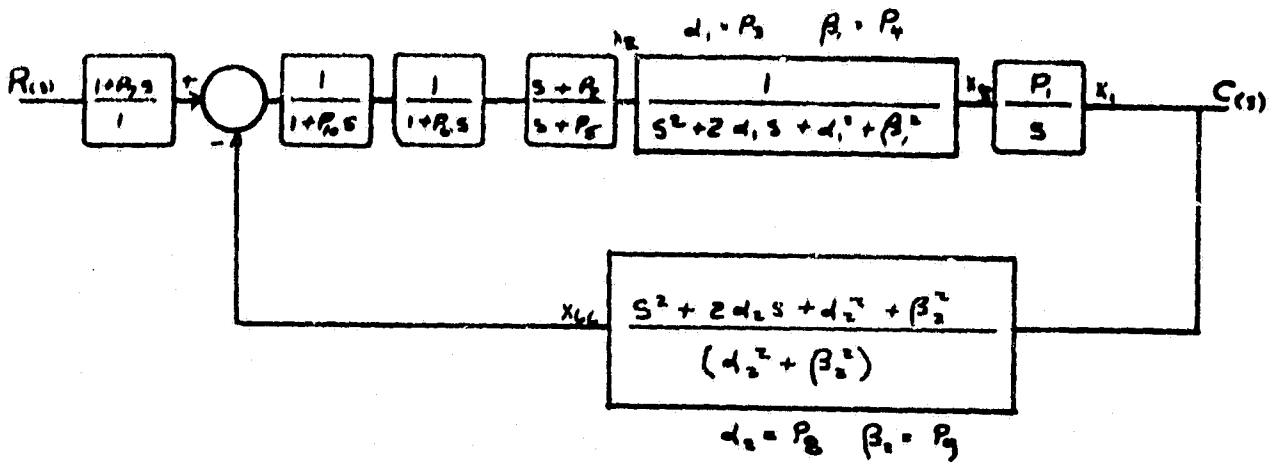
$P_1 = 1$, $P_3 = .2$, $P_4 = .98$, $P_5 = .5$, $P_6 = .1$, $P_8 = .707$

$P_o = .707$, $P_{10} = .05$

Program	Overshoot	Rise Time	GFAR	Remarks
QD017	9.27%	2.868 sec.	4×10^{-10}	
MIMIC	9.23%	2.86 sec.	--	
SWEEP	9.605%	2.86 sec.	4×10^{-10}	
SWEEP	9.39%	2.84 sec.	4×10^{-10}	With derivative modification

Figure 2.4

FCT 6 Definitions and Test Case



$$\dot{x}_1 = P_1 x_2$$

$$\dot{x}_2 = P_2 x_3 + \dot{x}_3 - P_5 x_2$$

$$\dot{x}_3 = x_4$$

$$\dot{x}_4 = -2P_3x_4 - \omega_{22} x_3 + (x_7 - x_{66})$$

$$\dot{x}_5 = \omega_{12}(x_1 - x_{66})$$

$$\dot{x}_6 = x_5 + 2 P_8 x_1 - \omega_{12} (P_6 + P_{10}) x_{66}$$

$$\dot{x}_7 = (x_8 - x_7)/P_6$$

$$\dot{x}_8 = (1. - x_8)/P_{10}$$

$$x_9 = C(t) = x_1 + P_7 \dot{x}_1$$

$$\omega_{12} = P_8^2 + P_9^2$$

$$\omega_{22} = P_3^2 + P_4^2$$

Test Case:

$P_1 = 1156$, $P_2 = 3$, $P_3 = 5$, $P_4 = 1$, $P_5 = 2$, $P_6 = .033$,

$P_7 = .1$, $P_8 = 2.5$, $P_9 = 3.56$, $P_{10} = .005$

Program	Overshoot	Rise Time
QD017	14.836%	.43 sec.
SWEEP	14.07%	.434 sec.

Figure 2.5

FCT 20 Definitions and Test Case

to require a smaller step-size than others.

The most obvious source of trouble is in the assignment of open-loop gain. In all of the block diagrams, there is an equation of the form, $\dot{X}_1 = P_1 X_2$ where P_1 is the open-loop gain. In the configurations used in the parameter variation designs, P_1 can be quite large. This emphasizes any errors in calculation of X_2. In an analog simulation, this would show up as an integrator with an excessively large input gain.

In digital simulation, one is not supposed to have to worry about scaling because the floating point arithmetic takes care of that. However, the fixed step numerical procedure assumes that the errors in all variables are evenly weighted. A large gain into $\dot{X}_1$ violates that. The solution was to break the P_1 into two parts, $(P_1)^{1/2}$. The two parts were assigned to two different equations so that the derivatives would be more nearly equal.

2.8 Timing Considerations

RUN is called to generate the time history many times in the course of the iterative programs. Thus it is important to reduce its running time as much as possible. In these programs, there are loops within loops, within other loops. The inside or tightest loops should be located and examined.

Within the gradient program described later, we

have the main cycle for determining a gradient vector, a step-size, and moving to the next point. There may be fifty to seventy-five such cycles in a typical run. Next, at the gradient level, for n variable parameters there will be n+1 calls to RUN, where n is typically four to six. The step-size determination will typically have five calls to RUN. Thus there are perhaps 500 to 750 calls to RUN in a typical program.

RUN has two loops within it. The first is the FRIT loop which calls FRESP2 twenty times or about 10,000 times per program. Within FRESP2, the execution time was reduced by avoiding calculation of subscripts in arrays. Certain constants are calculated once in SETUP before entering the loop. Since the value of bandwidth produced by this loop was not used in some programs, this loop was removed.

The second loop is the call to the numerical solution routine MIMIN. With a step-size of .02 seconds and a stop time of 5 seconds, MIMIN is called 250 times by RUN or 125,000 times in all. In some cases a step-size of .002 seconds was needed along with a stop time of 2 seconds, requiring 1000 calls to MIMIN. Within MIMIN there are four calls to an FCT routine to calculate derivatives. This routine may be executed between 500,000 and 2,000,000 times per program depending on step-size, etc. Thus the coding which must certainly be efficient is the integration loop of RUN going

through MIMIN to the FCT.

Several techniques were employed to reduce execution time. All coefficients which did not change during the time history were calculated in SETUP before entering the loop. All variables and constants needed were passed via COMMON to eliminate having to set registers with variable locations unnecessarily. Subscripted variables were used only where DO loops were necessary to avoid calculations of subscripts that were fixed. A double subscripted array originally used in MIMIN was changed to three single subscripted arrays. These arrays are involved in a loop executed NEQ times where NEQ is the number of differential equations.

The portion of RUN which examines the response to pick off overshoot, etc., must be executed with each call to MIMIN. No special techniques were used here and no loops are involved.

The problem of reducing running time is a very real one. A typical run with four variable parameters and fifty-three iterations required 130 seconds to run on a CDC 6400. This is 2.6 seconds per iteration cycle or roughly 0.25 seconds per call to RUN. Other programs with five parameters and twice as many differential equations will use 256 seconds for only twenty-five iterations. These programs are quite compute bound, and do very few input-output operations.

Before attempting to extend the design programs

described later to practical situations, a more efficient method of generating the time response is needed. Certainly a hybrid computer could provide this, as the time per solution remains the same as the order of the system increases.

2.9 QD017 Control System Analysis Program

The QD017 program is a modification of a program designed at the Martin-Marietta Company Denver facility for analysis of the flight control portion of the Titan missile system. The control system must fit the block diagram of Figure 2.6. The blocks A, B, C, D, E, F contain transfer functions which may be specified in either polynomial form or root form and the latter may be either in root locus or Bode form. The program then multiplies out the block transfer functions to form desired open-loop or closed-loop transfer functions corresponding to the eight cases given on Figure 2.6. Here the poles and zeros of the resulting transfer function are presented in root locus and Bode form. At this point either a frequency response or a transient response, or both, may be requested. The transient response calculates residues, forms the partial fraction expansion, and calculates a time response. This latter procedure is better than the solution of the differential equations in two senses. Poles far removed from the principal dynamics have small residues and never affect the calculation. These poles have short time

Case	Transmission	Case	Transmission
1	X_1/Θ_{in}	5	$C \cdot D$
2	X_2/Θ_{in}	6	$B \cdot C \cdot E$
3	X_3/Θ_{in}	7	$A \cdot B \cdot C \cdot F$
4	Θ_{out}/Θ_{in}	8	$CD + BCE + ABCF$

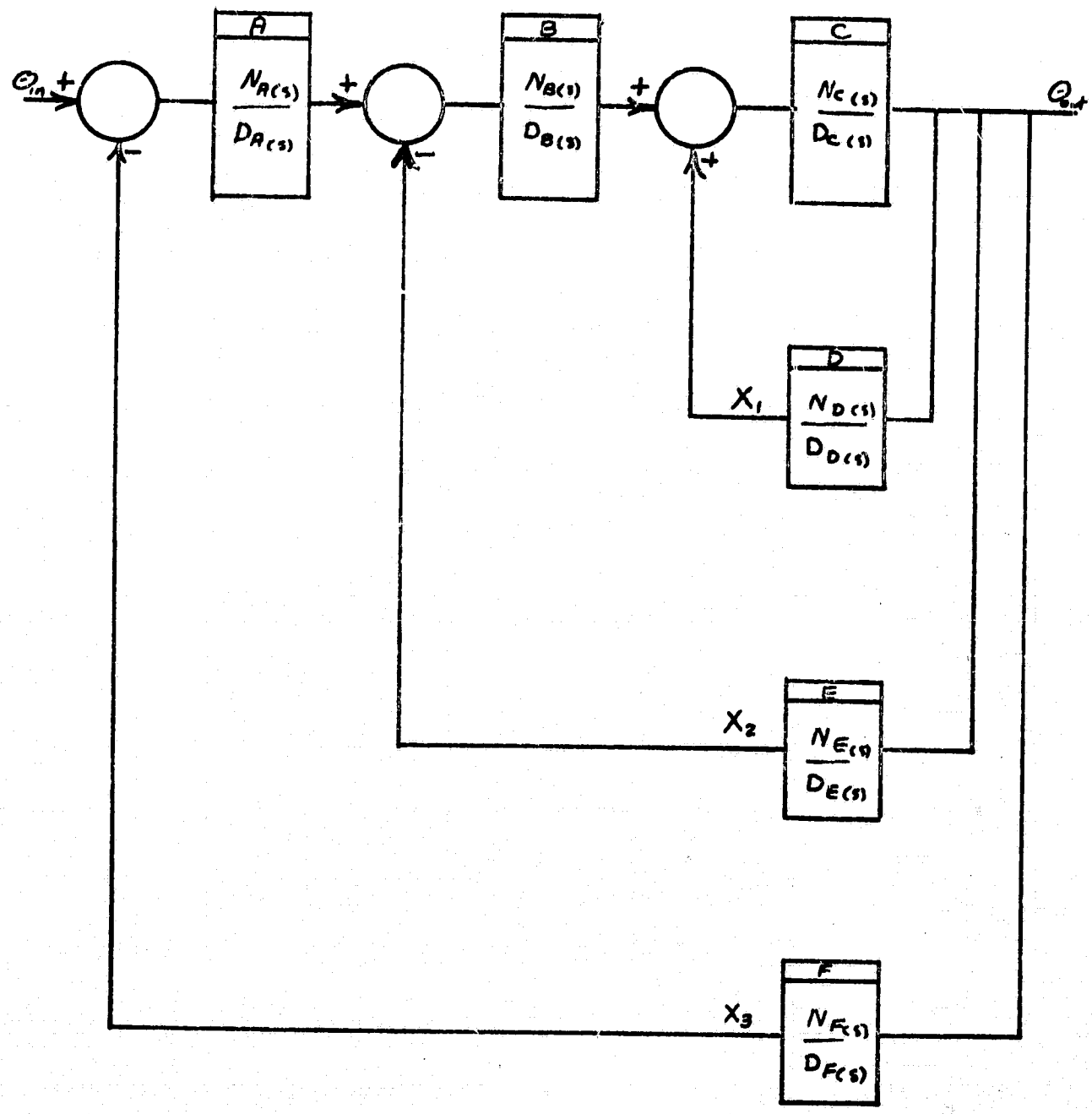


Figure 2.6

QD 017 Block Diagram

constants and go to zero quickly. These poles continue to affect the numerical solution of the differential equations as long as the output is changing. The modification to lessen the impact of the far-away pole was described earlier.

This program, as described, requires about 10 seconds to compute a check case for one of the FCT cases. However, by reducing the allowable block diagram to a two-degree-of-freedom one, by using pre-specified cases, by calculating the frequency response at only one value of ω, and removing all input-output, this program could provide an alternate method of supplying the values needed by RUN.

As will be discussed elsewhere, a problem was encountered with systems having a small first overshoot but were unstable. This program provides the closed loop roots. When such a situation is detected, the gradient program could minimize the real parts of all roots in the right-half plane to obtain a stable system and then proceed to minimize the overshoot until a satisfactory response was obtained.

However, the real power of this program lies in the fact that it can handle systems of arbitrary size (out to 100th order polynomials) automatically and with good accuracy. In the design of a control system for a plant with large parameter variations, one establishes poles and zeros to give a desired transient response.

Then alternating poles and zeros are placed in the higher frequency region, in order to reduce $|L(j\omega)|$ as fast as possible. While these poles and zeros do not affect the response greatly, they cannot be ignored. One must select a pole which approximates the many far-off poles. One reason they are placed far out is that design procedures to handle them closer in do not exist.

This program was modified so that it would first provide microfilm plots since these are of better quality than the printer plots. It was then modified to work on the CDC 282 display console system.

After the program is loaded, it reads some control cards to see what it is supposed to do. It may take the data describing the system and root values from the keyboard. It will then display frequency or transient responses as directed by keyboard responses. The frequency responses may be either gain-phase plots or Bode plots. Thus the designer can add the far-off poles and zeros and see immediately what the results will be, without worrying about correcting asymptotic plots. The transient response plot can be displayed in order to be certain that the performance has not been degraded.

CHAPTER 3

CONCEPTUAL BACKGROUND OF AUTOMATIC DESIGN PROGRAM

3.0 Introduction

This chapter describes a method for automatic design of a feedback control system to achieve specified step response. The procedure used is a steepest-descent gradient search in the parameter space to minimize the open-loop high frequency transmission while maintaining an acceptable time domain response. No approximations are necessary to relate time performance to stability margins, etc. Since this procedure differs from previously published methods, a brief historical review is in order to place the method in proper perspective.

3.1 Review of Known Work

The basic procedures for minimization of functions has been available for some time. The calculus of variations treated these problems as far back as 1696 with the brachistochrone problem. Procedures for minimizing functions of a single variable have been studied and sophisticated analytical and numerical methods developed[12,13]. Procedures for minimizing (or maximizing) functions of several variables represent the next level of complexity. These are described crudely for the two-dimensional case as hill-climbing problems where the hill may have a variety of sharp ridges, false summits, saddles, etc. Numerical procedures are available for problems in which the function can be evaluated but its

derivatives are not conveniently available analytically.

There was a period in the development of control system design when it was fashionable to simulate the system on an analog computer and manually "twiddle" potentiometer settings until a satisfactory response was attained. Once a satisfactory response was attained, the designer had no idea how it compared to other possible acceptable designs. This human designer was manually performing a multi-parameter optimization. The use of a large digital computer program can automate the logic decisions of the human designer, making them more precise and repeatable.

The use of computers in automated design is relatively new. Network synthesis and process control have been perhaps the most active areas. In both of these the computer has been used for actual design and for model development and output prediction. The latter is not actual design, but the techniques employed are quite similar.

Temes and Callahan[21] surveyed the results in network synthesis. The design procedure usually involves fitting a model response to a single desired response in the frequency or time domain. Some procedures operate directly with the network equations while others define a set of adjoint equations to refine the iterative procedure, as in Rohrer[17].

The procedure of fitting a model response curve to a

desired curve usually involves minimizing the sum of squared errors at several data points. The variable parameter values are manipulated to achieve this minimum error.

In the process control area, model development has been used effectively. Model parameters are varied until the model response fits experimental results. With a good model, results of experiments may be predicted before the experiment is tried. Process control also utilizes supervisory computers to determine optimum set points for a reaction. Perturbation and gradient methods are employed to find the best combination. These optimizations are performed periodically on-line to account for outside influences.

Another example from the aerospace control discipline is a program[6] to select gain and filter pole locations in the autopilot of a large flexible launch vehicle. The vehicle is open-loop unstable and the frequencies of structural bending modes are quite close to the desired crossover frequency. These autopilots often involve five feedback loops with different transfer functions due to sensor differences. The large number of available compensation variables makes human optimization quite difficult.

References describing results in the areas mentioned have noted the difficulties of optimization in multi-dimensional spaces. The functions to be minimized

often have false minima, saddle points, sharp ridges or valleys, slowly turning steep-sided yet gently descending valleys, etc. Quite elaborate methods have been devised to cope with some of these specific problems[21].

3.2 Control System Design Problem

In the normal control system design problems, the desired result is not usually specified as a single acceptable response plot. The step response specifications are more likely to be (say) [Figure 1.2]:

1) overshoot $\leq$ 15%
2) 0.5 $\leq$ rise time $\leq$ 1 sec.
3) settling time $\leq$ 2 seconds.

There are many designs which satisfy the above requirements. A good criterion for choosing among these is how well the system attenuates high frequency sensor noise. This criterion makes sense only for a single degree-of-freedom structure.

3.3 Outline of Automatic Control System Design Procedure

It is desirable that the program start with fairly arbitrary initial parameter choices, which may not satisfy the performance requirements and may even be unstable. The first phase of operation is to manipulate parameters to at least satisfy the time response constraints. The program may be able to operate on only one unacceptable response value at a time. That response would be made acceptable before working with another one. Thus there is a choice between assigning a hierarchy of

responses (e.g., overshoot is more important than rise time), or a form of each taking turns. This is important since different boundaries may have opposite effects on parameter variations. That is, increasing gain improves rise time but degrades overshoot.

When the program has succeeded in finding a parameter combination which satisfies the time performance, it turns to minimizing the cost functional. This functional could be one or a combination of several things. Open-loop bandwidth, or somewhat equivalently the value of L(s) at a high frequency, is a typical choice.

The program then enters the third phase; namely, the location of the constrained minimum. The method of steepest descent may be compared to seeking the top of a hill in a pea soup fog. One sticks a foot out in several directions, decides which goes uphill, and proceeds that way for several steps; tests again, etc. Also, there are fences on the hillside at unknown locations, so it does no good to keep track of one's position. The fences can be located only by bumping into them. One then turns sideways, sees which way is up, and moves along the fence. But the fence can be followed only by constantly hitting it, as a blind man taps his cane along the curb. Since there are several performance specifications, there will be several fences; i.e., one may encounter a second fence while following the first. One is also allowed only a limited number of lengths of

steps. Thus, one may try to step over a fence and have to go back and try a smaller step length. It is also conceivable that the contour of the fence and hill could be such that one must leave a boundary and resume normal hill climbing when the boundary no longer interferes with progress toward the optimum solution.

There is another class of boundaries which also must be considered. The performance boundaries are output restrictions and can be located only by testing. However, the parameter space itself may also have direct limits imposed. For instance, for a satisfactory velocity constant, K_v, there is a minimum value of gain. For realistic components in the compensation networks, the pole in a lag compensation network may have a minimum value. Or the separation of pole and zero in a lead network may have a maximum value. These constraints are inequality or nonlinear constraints, yet they are easier to handle since they are defined directly on the input or parameter space.

The final question is when to stop the search. As one approaches the top of the hill, the slope approaches zero. Mathematically, this is stated that the norm of the gradient becomes small and this may be tested for a stop condition. However, if the actual minimum is outside the constraints it will spend its time moving along the boundaries until told to stop. In this case, a local constrained minimum exists when one is hitting the

boundary headon. Going to either side does not improve the cost function. As will be seen later, the dot product of the cost function gradient and the normal to the boundary (its gradient) is used to follow the boundary. When this dot product approaches ± 1, the program has found a local constrained minimum. The possibility of encountering a second constraint before reaching this condition does exist.

When a minimum is located, the question is always asked, "Is this the minimum for the entire space or just a local minimum"? This is usually resolved by selecting several different starting points and checking that the end result is the same. This may be done by the computer with a random number selection or by the human designer. One must not be too critical of the results at this point. There may be many designs which are equivalent. For instance, when using lag compensation, one could move the pole-zero dipole to a lower frequency without appreciably affecting either the frequency response at high frequencies or the transient response. The important parameters must be identified by referring to classical design techniques.

3.4 Step-Size Selection

The approach just described has two parts: The first is to decide which way to proceed and the second is how far to go. The first phase is accomplished by evaluating the gradient of the criterion function. This gives the local direction of steepest slope. This procedure is

relatively straightforward.

The second phase of step-size selection is not as simple, and is the most critical phase of a gradient search. A poor procedure will not just cause poor searching; it may lead to failure.

One can play safe with very small steps leading to only slight changes in gradient direction with consequent inefficient use of computer time. But if too large a step is taken, the gradient direction changes considerably, which can lead to much wandering around. Clearly, there is an optimum between these two extremes. Figure 3.1 shows examples of the above.

If the functional surface is a hemisphere, one gradient calculation would suffice. Each gradient vector passes through the minimum of the function. One need only determine the proper step-size. However, most practical problems do not have spherical functional surfaces. It is noted that the optimum step-size varies along the path.

Surfaces in five and higher dimensions become difficult to visualize. The computer program cannot visualize anything; its selection must be based on numerical evaluations. Even with a perfect algorithm, the computer program can never find exactly the correct step size to the top of the ridge as indicated in the optimum curve of Figure 3.1. Quantization of numbers will prevent this. So as ridges become steeper, there increases the tendency for ridge hopping with little progress toward the maximum.

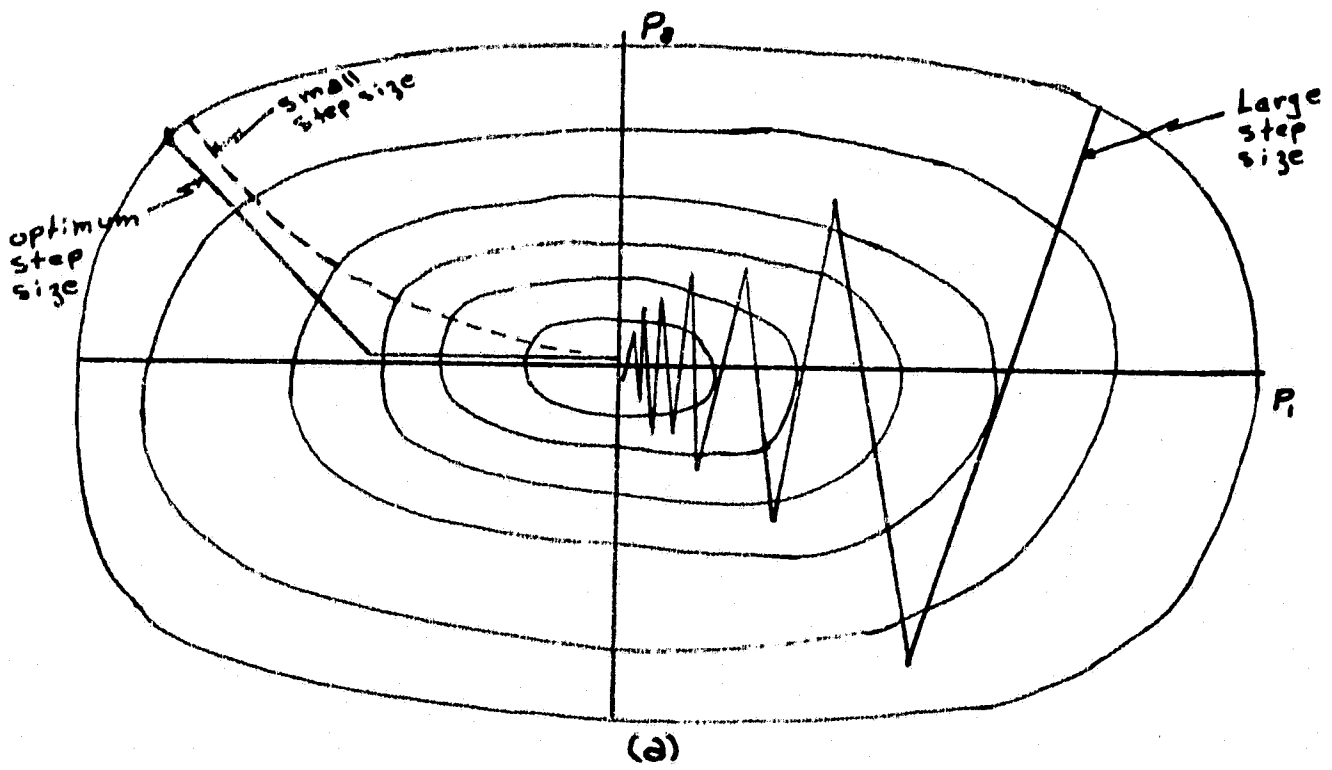


Figure 3.1

Example of Step Size Selection in Gradient Search

In the control system design procedure outlined here, there are at least four function surfaces: overshoot, rise time, settle time, and high frequency response. It is unlikely that any functional transformation suitable for one (such as changing ellipses into circles) will help another.

3.5 Alternative Procedures

The calculation of all the gradients, selection of step-sizes, location and following of boundaries, etc. can be very time consuming. Two alternate procedures exist. The first moves through the parameter space in a random walk, remembering the location of the best value of the cost function which did not violate any boundaries. The program tries a fixed number of points and stops. This requires evaluation of the function at a very large number of points. However, so does the gradient method and the latter has a large amount of additional supporting calculations. Since this procedure is very straightforward, it has definite merit in situations where the system is nonlinear, particularly if discontinuous. It is very difficult for the gradient procedure to cope with a discontinuous boundary or cost function such as with pulse-width modulated control.

The second approach makes use of the sophisticated mathematical methods for location of an extremum in an unconstrained space where the surfaces do not have evenly spaced, circular contours.[29] In this case, the performance boundaries are not considered as separate

entities or constraints, but are brought into the problem as weighted terms added to the cost functional. The weighting is such that inside the boundary there is very little added to the cost. But outside the boundary the cost increases very rapidly. This is still a continuous function rather than inequality constraints. The boundaries are further weighted in proportion to the number of iterations performed. The program proceeds as follows.

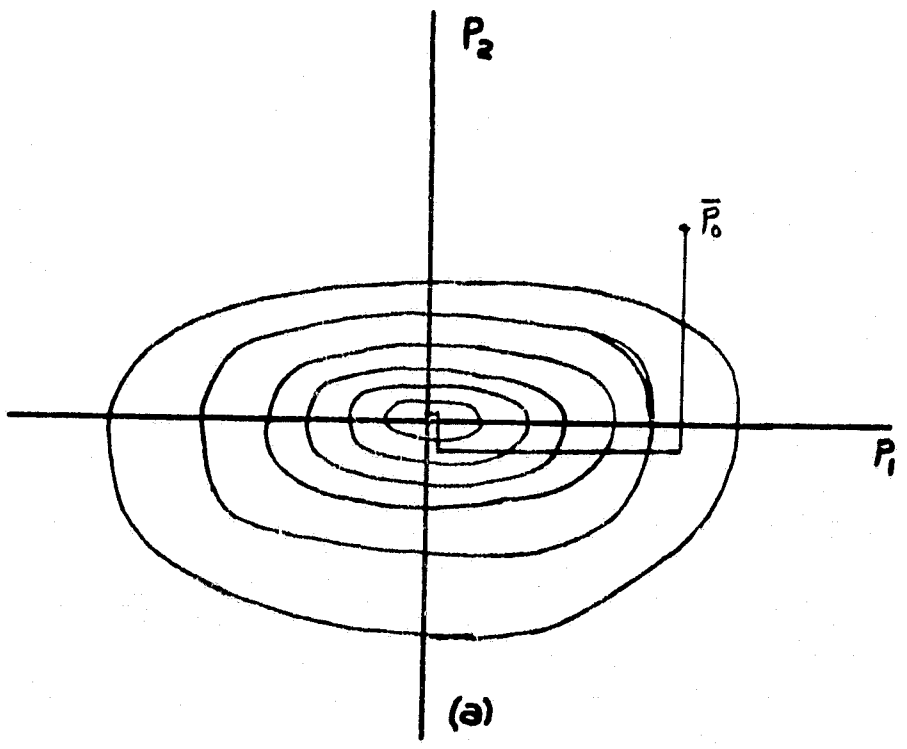
The boundaries receive little weighting, such that the cost function is relatively unconstrained. The program locates a local minimum in this space and remembers its location. The weighting on the performance boundaries is increased. Starting from the preceding local minimum, a new local minimum is located. This procedure continues, increasing the effect of the boundaries each time.

This has the effect of slowly turning up the edges of the surface of the cost function to deny those locations which violate the performance requirements. Thus, each successive minima is closer to the constrained minimum. If the surfaces are reasonably well-behaved, the procedure of locating a minimum in the almost unconstrained space and working gradually into a constrained space can eliminate problems of being diverted to poor local minima by action of the boundaries in the method first described. If the actual constrained minimum is significantly far away from the first minimum located, even this method will have much work to do to move from one to the other.

Since this method relies on the gradual introduction of the boundaries and on not moving far in any given step, it can improve the situation where one is searching down a wedge between two boundaries. This procedure increases the sharpness of the wedge gradually and the program should be near its final optimum before the sharpness would cause any difficulty. Otherwise, even the sophisticated ridge (or valley) following procedures will become confused as the sharpness of the boundary transition increases.

Within the gradient or hill climbing procedure itself, two alternatives exist. First, one can calculate a complete gradient involving partial derivatives with respect to all the parameters. Then a step is taken in this direction. This method is known as the steepest descent method. It involves n+1 passes through the system simulation routine for each gradient and a few more to select the step size, which can consume quite a bit of computer time.

In the other approach, one works with only one parameter at a time, taking each in turn. The movement is parallel to an axis until a minimum is detected along that line. Then a turn is made and the same movement is made parallel to the second axis. And so on. This method is known as the rotation method. It requires many fewer passes through the simulation subroutine. These two procedures are shown pictorially in Figure 3.2.



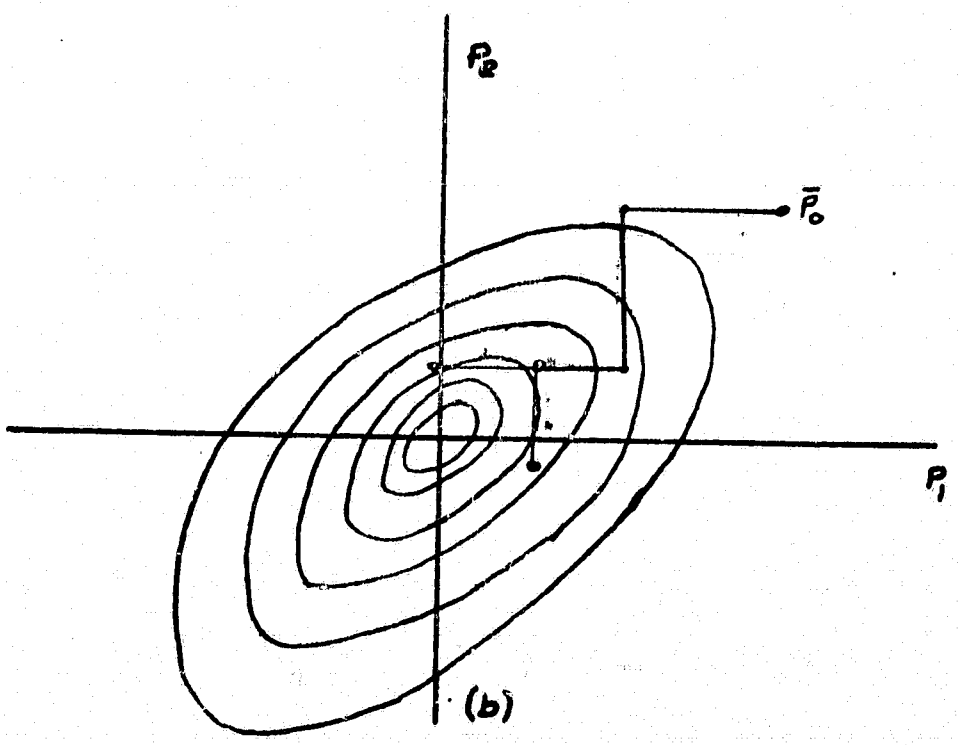


Figure 3.2
Examples of One-at-a-Time Parameter Search

However, the latter procedure is quite sensitive to ridges in the surface. From Figure 3.2a, it is seen that the search proceeds directly to the desired solution. However, Figure 3.2b shows the same ridge, only not parallel to a parameter axis. Once the ridge is attained, the search hangs up. The program takes a step along one axis and it goes downhill. It takes a step along the next axis and it goes downhill. So it concludes it has reached the summit and stops. This does not imply that the steepest descent method is much better. It can waste a lot of time zig-zagging across the top of the ridge without ever getting actually on top. But at least it keeps going.

These examples are presented to emphasize the importance of proper selection of parameter definitions. The references state that this situation calls for a linear transformation to rotate the ridge, flatten it out, and make the contours more nearly circular. But most of the references conveniently avoid describing how to determine the proper transformation. It is easier to avoid the ridge problems in the problem definition phase than to cure them later.

A look at classical control system design procedures warns of several possible problems. In lag compensation, the ratio of pole to zero is the important parameter. The absolute location of the dipole is not important so long as it is sufficiently below the crossover frequency. One would expect the program to proceed quickly along the

ratio parameter into a valley which would slope rather gently along the absolute location parameter. If on the other hand actual pole-zero values are the parameters, it places the valley diagonally across the plane. This means the program may become involved with zig-zagging back and forth across the valley unless it is fortunate enough to actually hit the bottom. This same choice occurs between standard δ, ω_n or α, β specification for complex poles and zeros.

3.6 Parameter Space Selection

So far, the discussion has left open one very important question which must be answered before actual work on a program can begin. The question is as much philosophical as technical. We have spoken of manipulating a vector of parameters to bring the time response within specification and then to minimize some other function of these parameters. But just what is the relationship between the parameter space and the response spaces? In the control system problem, it is preferable to work with open loop poles, zeros, etc., or closed loop poles and zeros?

The problem is more easily solved by means of closed loop parameters. The required order of system to fit a given set of specifications is not large. The ranges of parameters will not be large. However, once a satisfactory closed loop response is obtained, the parameter space must be transferred to the open loop space to actually specify

the compensation poles and zeros. Since the plant undoubtedly has fixed poles and zeros, the designer must have these appear in the open-loop transfer function or he must cancel them and replace them with poles and zeros elsewhere. The latter represents additional complexity of the compensation network and always raises the question of what happens if the cancellation is not complete. Truxal[34] has described a procedure for synthesis of the open compensation as a R-C network including the plant poles.

If the optimization procedure is not sensitive to the choice one might as well work directly in the open-loop space. The plant poles and zeros are included exactly and held fixed. Constraints on open-loop parameters such as minimum gain are easier to apply. On the other hand, the ranges of parameters will be greater. However, all poles and zeros are included in the simulation and one need not place artificial restrictions on parameters to make them "far-off." The exact system response is obtained, not that of some reduced "dominant" system.

CHAPTER 4

DESCRIPTION OF AUTOMATED DESIGN PROGRAM ALGORITHMS

4.0 Introduction

This chapter describes the algorithms and some features of the coding of the automatic design program. This program can be broken into five major sections: control gradient, step-size selection, constraint or boundary follower, and finally, the control system response simulation section. The last is performed by the subroutine RUN described in Chapter 2. The remaining sections are described in this chapter. The flow chart of the overall program is shown in Figure 4.1. Detailed flow charts are given for each section.

The iterative design philosophy can be summarized as follows:

Step 1. Start at $\bar{P}_{(o)} = \bar{P}_o$, $GFAR(\bar{P}_o) = F_o$

Step 2. Evaluate the gradient ∇F_o at $\bar{P}_o$

Step 3. Compute a discrete parameter change to be made in the gradient direction as

$$\Delta P = K \nabla F_o$$

Step 4. Calculate the new parameter values by the equation $\bar{P}_{i+1} = \bar{P}_i + \Delta P$

This basic cycle is repeated until a satisfactory result is obtained. It is modified to include the effects of the inequality constraints of performance specifications.

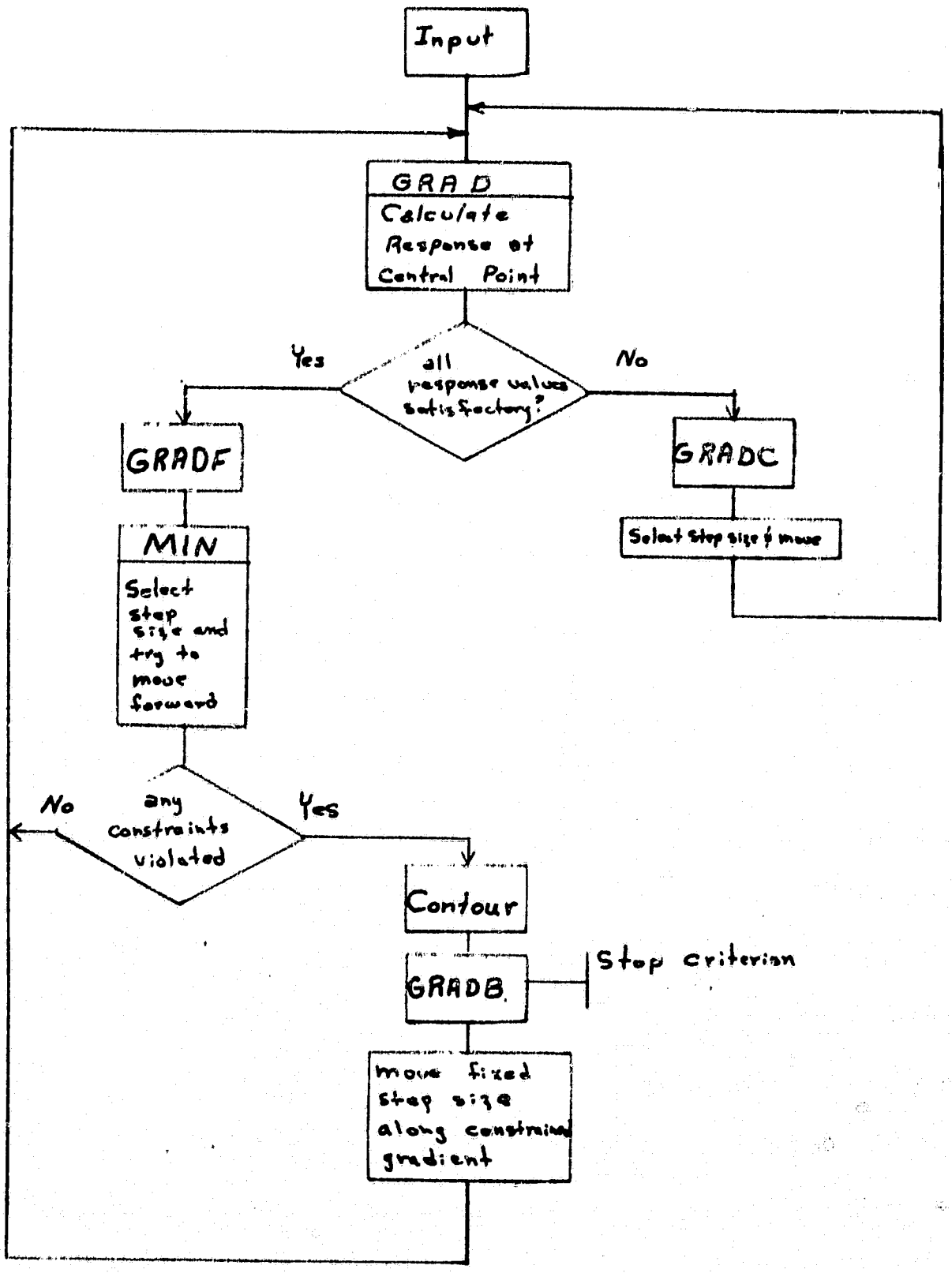


Figure 4.1

Flow Chart of Steepest Descent Design Program

4.1 Main Program

The main program first reads input data, sets initial values for parameters, counters, etc., and sets up initial calls to the microfilm plotting package. There is a return to the main program once each iteration cycle, so that output may be generated and various stop criteria evaluated.

4.2 Gradient Calculations and Routines

The gradient of $F_{(\overline{P})}$ is defined as

$$\nabla F_{(P)} = \left(\frac{\partial F}{\partial \rho_1}, \frac{\partial F}{\partial \rho_2}, \frac{\partial F}{\partial \rho_3}, \ldots \frac{\partial F}{\partial \rho_n} \right) \qquad (4.2.1)$$

The partial derivatives are found by a discrete perturbation process, using the definition of the derivative before passing to the limit.

$$\left.\frac{\partial F}{\partial \rho_i}\right|_{P_o} \cong \frac{F(\rho_1, \rho_2, \rho_3, \ldots \rho_i + \Delta\rho_i, \ldots, \rho_n) - F(\rho_n)}{\Delta \rho_i} = \frac{\Delta F_i}{\Delta \rho_i} \qquad (4.2.2)$$

The gradient is evaluated at $\overline{P}_o$, the central point of the perturbation. Each parameter is perturbed in turn by ΔP_i and the resulting ΔF_i noted, as given by a call to RUN. Normally ΔP_i is 10% of P_i. However, because of quantization imposed on returned values of ΔF_i a minimum value for ΔP_i of .05 is used. This quantization is particularly evident in the results for rise time and settling time.

The gradient calculation section is composed of one control routine GRAD, three different gradient routines GRADF, GRADC, and GRADB, and two accessory routines. The flow chart is shown in Figure 4.2. The control routine

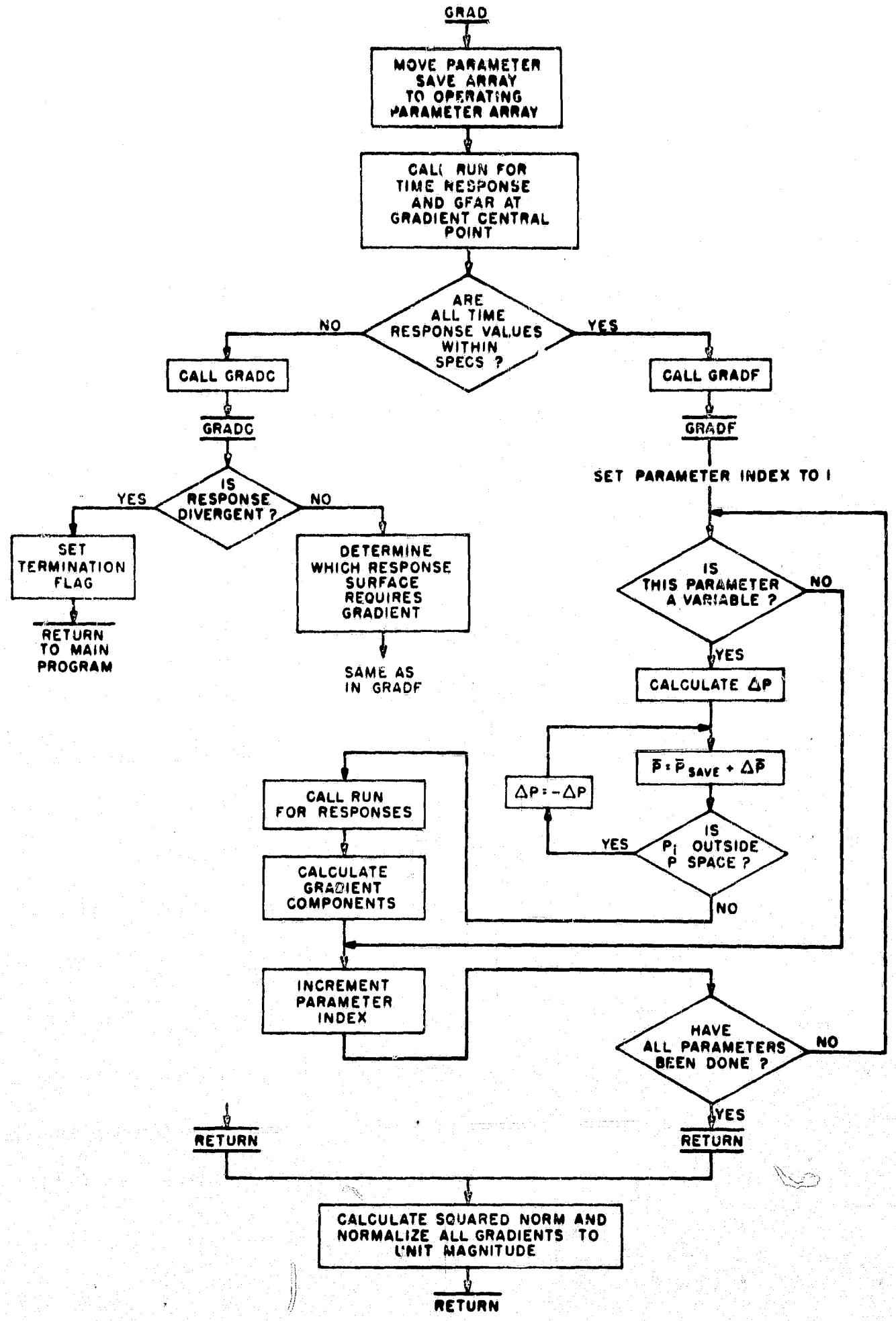

GRAD
MOVE PARAMETER SAVE ARRAY TO OPERATING PARAMETER ARRAY
CALL RUN FOR TIME RESPONSE AND GFAR AT GRADIENT CENTRAL POINT
ARE ALL TIME RESPONSE VALUES WITHIN SPECS ?
NO
YES
CALL GRADC
GRADC
IS RESPONSE DIVERGENT ?
YES
NO
SET TERMINATION FLAG
RETURN TO MAIN PROGRAM
DETERMINE WHICH RESPONSE SURFACE REQUIRES GRADIENT
SAME AS IN GRADF
CALL GRADF
GRADF
SET PARAMETER INDEX TO I
IS THIS PARAMETER A VARIABLE ?
NO
YES
CALCULATE ΔP
P̄ = P̄ SAVE + ΔP̄
ΔP = -ΔP
IS Pi OUTSIDE P SPACE ?
YES
NO
CALL RUN FOR RESPONSES
CALCULATE GRADIENT COMPONENTS
INCREMENT PARAMETER INDEX
HAVE ALL PARAMETERS BEEN DONE ?
NO
YES
RETURN
RETURN
CALCULATE SQUARED NORM AND NORMALIZE ALL GRADIENTS TO UNIT MAGNITUDE
RETURN

Figure 4.2

starts the gradient procedure by evaluating the function at the central point. The program operates such that it minimizes time response functions, until all satisfy the design specifications. It then minimizes the high frequency transmission, GFAR. The control routine must examine the time response at the central point to determine if overshoot, etc. are satisfactory. A flag is returned by RUN indicating which, if any, response specification is exceeded. RUN also warns if a divergent response or a response with "low overshoot" occurred. A microfilm plot of the response can be made and the design terminated if desired.

If the time response is unsatisfactory, the control routine will call GRADC to calculate a gradient, using the unsatisfactory response as the function. This is a gradient of the response surface, not a normal to a response surface of a particular value, and it will direct the program to minimize the response to zero. If the time response is satisfactory, the control routine will call GRADF to obtain a gradient using GFAR as the function.

Since both the time response and GFAR are provided by RUN, gradient vectors are calculated for overshoot, rise-time, and GFAR by both GRADC and GRADF. Only the appropriate gradient vector is placed in the array, which is used later for minimization. This allows the designer to observe the printout of all the gradient vectors as the program processes through the iteration cycles.

Several special tests are made in the course of the calculations. The components of the gradient are calculated in a loop, over all parameter indices. The program examines an array of flags associated with the parameter array and omits all parameters which are absent or fixed in value.

The limits on the parameter space must be imposed each time a step is taken along the gradient direction. The actual limits are imposed by NEWP described later. However, a subtle problem intimately related to parameter space scaling must be handled at this point. If any parameter is within 2% of either of its limits and its gradient component is such to send it beyond the limit, then that component is set to zero. However, as soon as the sign of the gradient reverses, the component is used again in the normal manner. The reason for invoking the above when the parameter approaches a limit rather than when it crosses it, is as follows. Situations occur when two function surfaces are used alternately, usually overshoot and rise time in the initial phase. An increase in gain improves rise time and degrades overshoot, so these components have opposite signs. When the two surfaces alternate the parameter is forced to its limit by one and pulled off by the other. If this parameter dominates the gradient, an oscillation occurs. The threshold on the parameter space limit tends to trap a parameter near the boundary and prevent this situation. This problem is intimately connected

with scaling and interaction of surfaces, discussed later.

The second accessory program calculates the squared norm of the gradient and normalizes it. The squared norm is

$$|\nabla F|^2 = \sum_{i=1}^{n} (g_i)^2 \qquad (4.2.3)$$

4.3 Step-Size Selection

Since a single gradient calculation requires n+1 calls to the system response simulation routine for n variable parameters, economy in the number of gradients calculated is important. Figure 3.1 has illustrated the problems involved. Step-size selection is selected in this program by the optimum gradient method described by Bekey and McGhee.[15]

The multi-parameter optimization problem requires a suitable change be found for each parameter. The gradient itself gives the relative values of the increments. The choice of the scalar distance along the gradient will be discussed in the context of the problem. It can be said at this point that step-size selection is probably the most important critical factor in this program.

Figure 4.3 illustrates the problem of locating the root of an algebraic function. For a two-dimensional minimization, imagine that the function surface is sliced vertically along the gradient vector. Figure 4.3 represents a side view, with the minimum at the point where the gradient vector crosses a valley and starts up the far

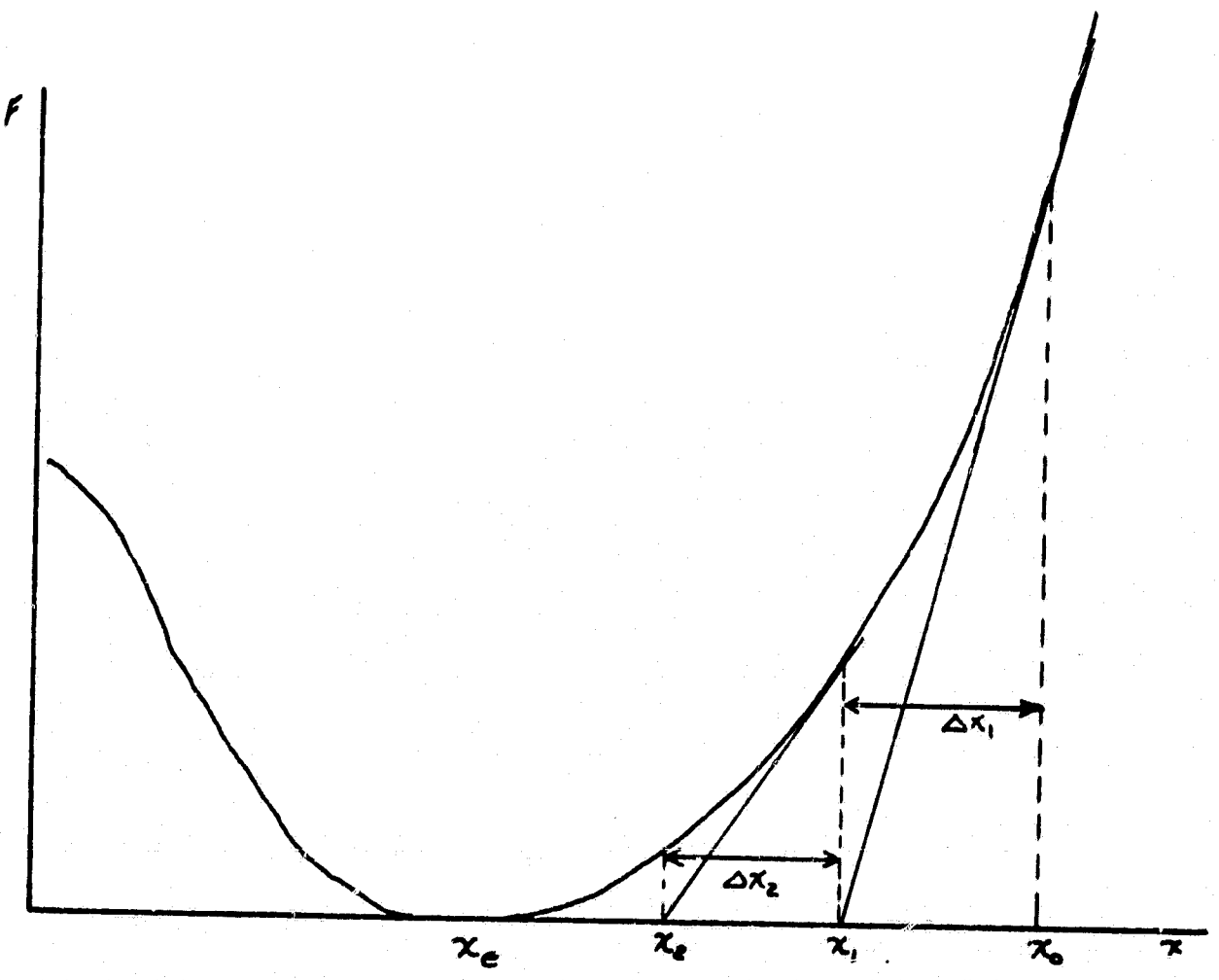


Figure 4.3

Newton-Raphson Step Size Determination

side. Thus the Newton-Raphson method is appropriate. The search begins at x_o seeking the root of the function $F_{(x)}$ at x_o. Consider a geometric approach first. The line which is tangent to the curve at x_o intersects the x-axis at x_1, giving a new approximation for the root. The slope at x_o is

$$F'_{(x_o)} = \frac{F_{(x_o)}}{x_o - x_1} \qquad (4.3.1)$$

solving for x_1,

$$x_1 = x_o - \frac{F_{(x_o)}}{F'_{(x_o)}} = x_o + \Delta x_1 \qquad (4.3.2)$$

It is convenient to write Δx as a function of the slope, i.e. $\Delta x_1 = -KF'_{(x_o)}$, so that

$$K = \frac{F_{(x_o)}}{[F'_{(x_o)}]^2} \qquad (4.3.3)$$

This must be extended to include the n-dimensional slope or gradient vector for $F'_{(x_o)}$, which is done more easily analytically. In the n-space, assume that $F_{(\bar{P})}$ has a convergent Taylor series at $\bar{P}_o$ such that

$$F_{(\bar{P}_o + \Delta\bar{P})} = F_{(P_o)} + \nabla F_{(P_o)}^T \Delta\bar{P} + R(\Delta\bar{P}^2) \qquad (4.3.4)$$

A step proportional to the slope is desired, or

$$\Delta\bar{P} = -K \nabla F_{(P_o)} \qquad (4.3.5)$$

Substituting (4.3.5) into (4.3.4) gives

$$F_{(\bar{P}_o + \Delta\bar{P})} = F_{(P_o)} - K[\nabla F_{(P_o)}^T \nabla F_{(P_o)}] = 0 \qquad (4.3.6)$$

so that

$$K = - \frac{F(\bar{P}_o)}{\nabla F(\bar{P}_o)^T \nabla F(\bar{P}_o)} = - \frac{F(\bar{P}_o)}{|\nabla F(\bar{P}_o)|^2} \qquad (4.3.7)$$

However, the above is based on the function being zero at the root x_e, which is not likely in a minimization problem. Note that as the minimum is approached, the gradient becomes small, tending to zero. If the function $F_{(\bar{P})}$ does not also approach zero, the step size will grow without bound. Hence, the restriction is made that K is used as an initial value for the step size and procedures are included for testing its validity.

The logic of the optimum gradient algorithm is given in the flow chart of Figure 4.4. The original algorithm of Bekey and McGhee has been modified to allow for the time response constraints when minimizing GFAR. The initial step size given by (4.3.7) is used when working with time response gradients to achieve a satisfactory response. The estimate is compared to several maximum allowable step sizes imposed to reduce undesired interactions between two constraint surfaces. Since no minimum is desired there, the complete logic is not needed and the subroutine MIN is used only for minimizing GFAR.

Once a step size has been determined, a subroutine NEWP calculates a new parameter vector from the equation $\bar{P}_{i+1} = \bar{P}_i + (\nabla F)(x_{step})$. Hard limits on the parameter space are imposed here by setting P_i equal to its limit if the

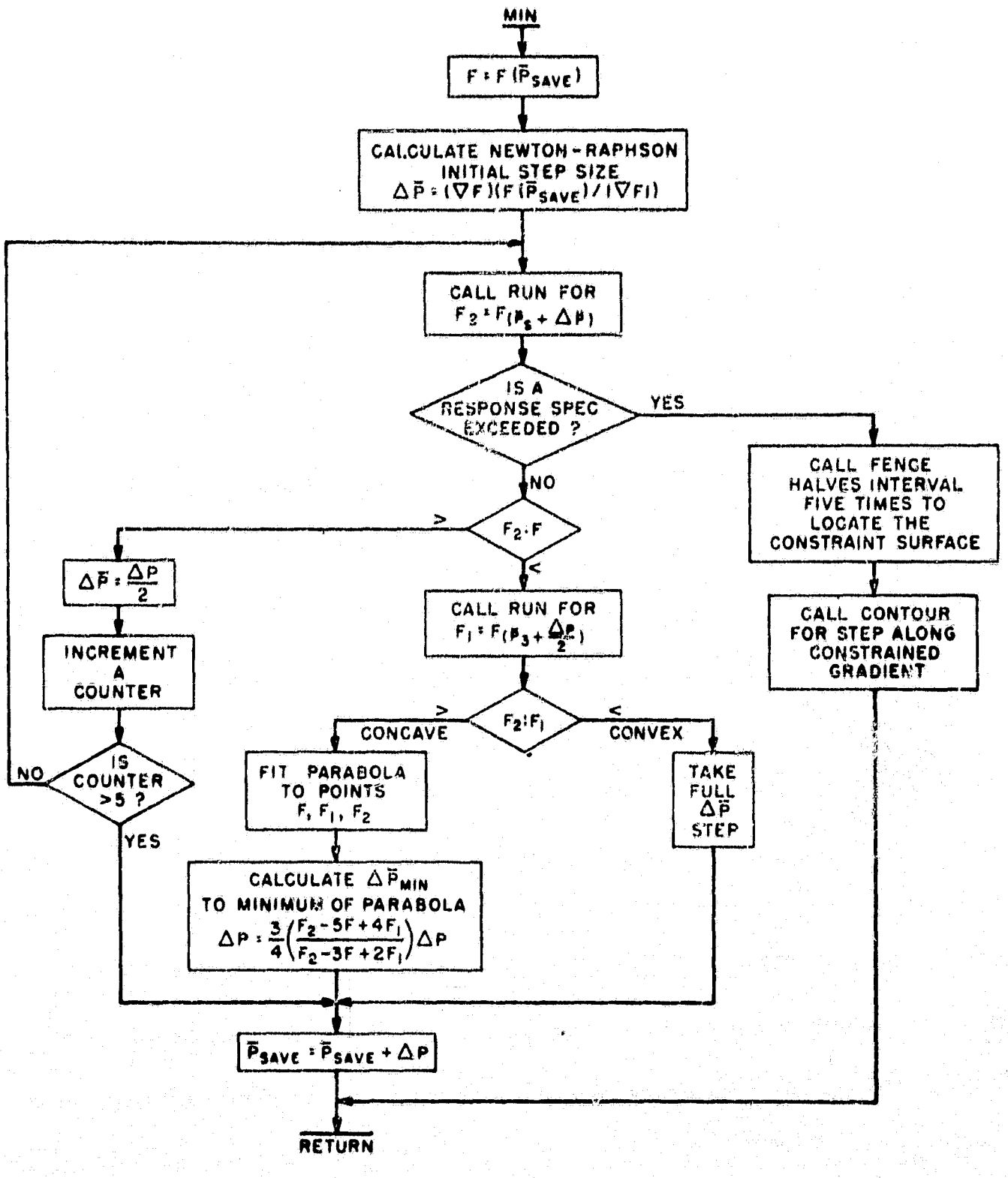

MIN
F = F(P̄SAVE)
CALCULATE NEWTON-RAPHSON INITIAL STEP SIZE
ΔP̄ = (∇F)(F(P̄SAVE)/|∇F|)
CALL RUN FOR F2 = F(Ps + ΔP)
IS A RESPONSE SPEC EXCEEDED ?
YES
NO
CALL FENCE HALVES INTERVAL FIVE TIMES TO LOCATE THE CONSTRAINT SURFACE
CALL CONTOUR FOR STEP ALONG CONSTRAINED GRADIENT
F2 : F
>
<
ΔP̄ = ΔP/2
INCREMENT A COUNTER
CALL RUN FOR F1 = F(Ps + ΔP/2)
F2 : F1
CONCAVE
CONVEX
NO
IS COUNTER >5 ?
YES
FIT PARABOLA TO POINTS F, F1, F2
TAKE FULL ΔP̄ STEP
CALCULATE ΔP̄MIN TO MINIMUM OF PARABOLA
ΔP = 3/4 ((F2 - 5F + 4F1)/(F2 - 3F + 2F1)) ΔP
P̄SAVE = P̄SAVE + ΔP
RETURN

Figure 4.4

calculation places it beyond that value. These limits allow design restrictions to be inserted, such as a minimum gain to satisfy a velocity error specification. They also prevent large steps from producing unreasonable values, such as negative gain. These would lead to unreasonable responses, tending to confuse the program.

The program tests for the following special situation. If the gain is too small, the overshoot will be zero and the rise time is equal to the stop time in RUN, for all perturbations. All components of the gradient and the squared norm would then be zero. A test for zero is made on the norm before division, to prevent premature termination of a series of runs. If the previous situation exists, i.e., rise time equal stop time, the program arbitrarily increases the gain and recycles. This usually brings the parameter values into a portion of the parameter space where the gradient procedure will work properly.

4.4 Boundary Following Procedure

The flow chart of Figure 4.4 shows that when the initial step goes beyond a response constraint surface, the subroutine FENCE is called to locate the constraint. Interval halving is performed five times on the initial step. Next, subroutine CONTOUR is called to move along the constraint surface. The method, whose logic is shown in Figure 4.5, has been adapted from Eveleigh,[11] which was intended for equality constraints rather than

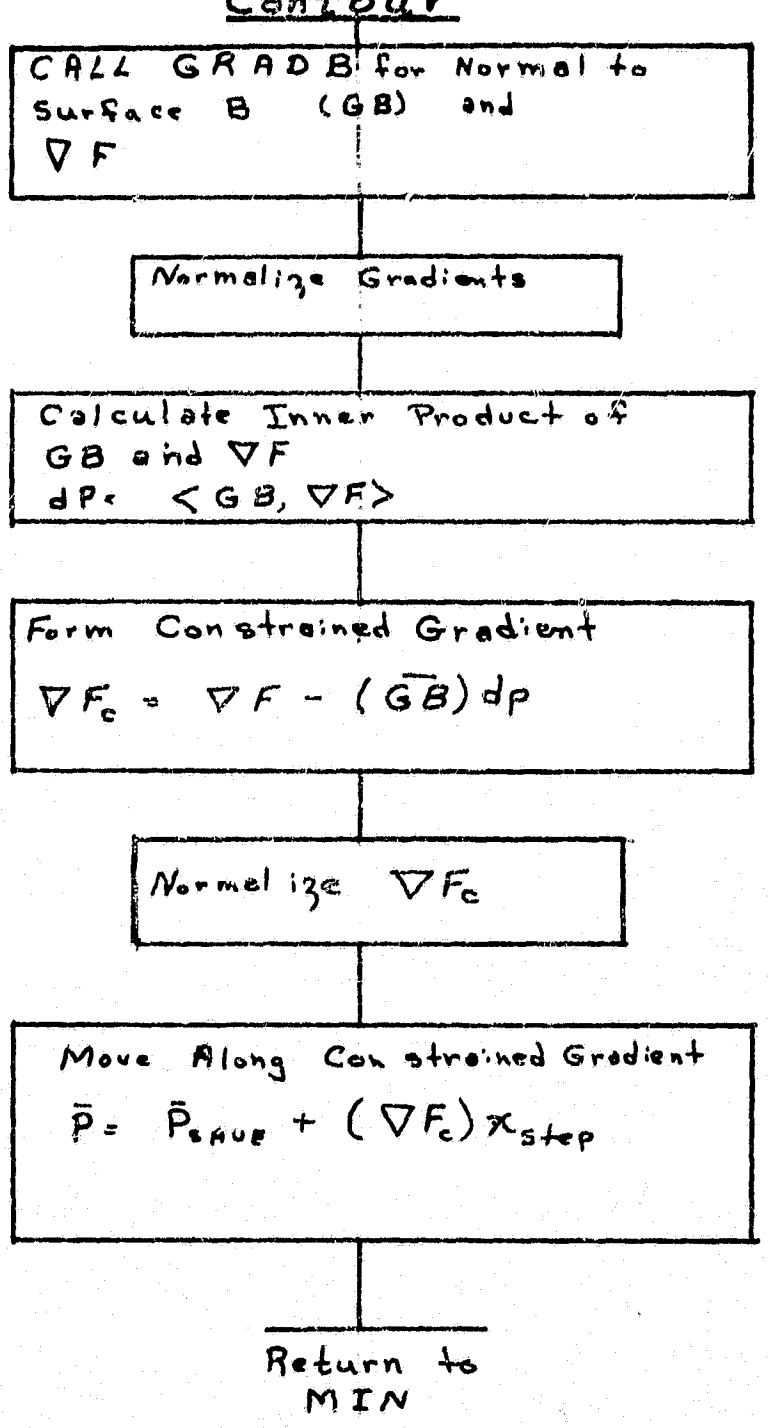


Figure 4.5

Flow Chart of Subroutine

CONTOUR

the inequality constraints existing here.

The point $\bar{P}$ given by the last step size attempt is defined as $\bar{P}_o$ and is assumed to be close to the actual constraint surface. The constraint surface is represented by $B_{(\bar{P})} = 0$. (For instance, OS-20 = 0 for overshoot of 20%). The normal to this surface is the gradient of B at $\bar{P}_o$; $\nabla B_{(\bar{P}_o)}$. $\nabla F_{(\bar{P}_o)}$ is also calculated. Figure 4.6 shows a possible orientation of these vectors. It is desired to move along the surface B in the direction of ∇F. This is represented by the constrained gradient ∇F_c. The gradient ∇F can be resolved into two components ∇F_c and ∇F_n where the latter is normal to B. ∇F_n is found by taking the inner product of ∇F and ∇B denoted dp. Then

$$\nabla F_h = dp(\nabla F) = (\nabla F)(\nabla F \cdot \nabla B) \qquad (4.4.1)$$

For this to be true, ∇F and ∇B must have unit magnitude. Then

$$\nabla F_c = \nabla F - \nabla F_h \qquad (4.4.2)$$

A step of fixed size is taken along ∇F_c. The Newton-Raphson initial estimate is tried, and if found too large, is replaced by fixed small value. This step along ∇F_c normally places the next $\bar{P}_o$ outside a constraint surface. It is possible to calculate additional terms, depending on surface curvature, to prevent this. However, the constraint is not an equality, therefore the program must check after each step along ∇F_c if a path leaving the surface is appropriate. The normal sequence of testing the response at $\bar{P}_o$ and selecting either GRADF or GRADC accomplishes this.

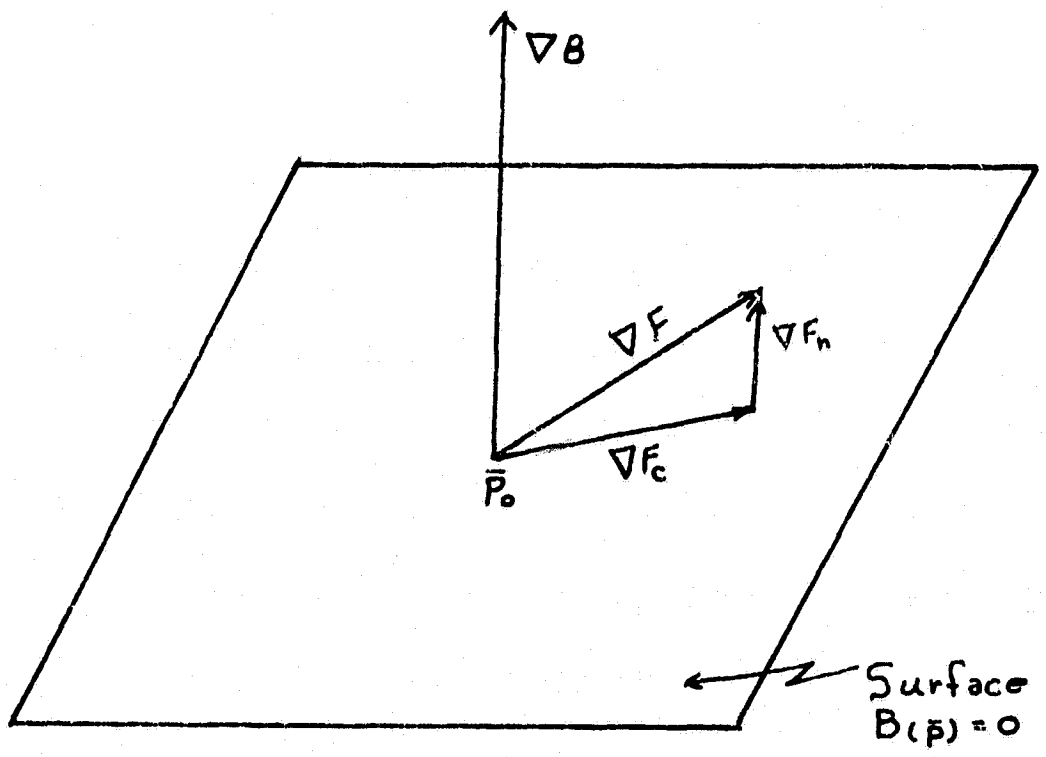


Figure 4.6

Relationship of the Gradient Vector when a Constraint Surface is Encountered

4.5 Termination Criterion

There must be a criterion by which the program can judge when a satisfactory design has been achieved and so to stop. The normal termination criterion for a gradient search is that the norm of the gradient has become sufficiently small. Since it is known that this program cannot reach the unconstrained minimum at which the gradient norm goes to zero, some other criterion must be established. However, the norm of the constrained gradient may be tested. ∇F_c tending to zero is equivalent to the dot-product of ∇B and ∇F tending to unity. This is the first test used. It is not adequate, as the search may encounter a second constraint with the dot product less than the stop value selected. The program will now attempt to search along the valley formed by the intersecting constraint surfaces, to further minimize the high frequency transmission. The program does not utilize information from both surfaces for the search and thus cannot define a stop criterion involving both surfaces. This is an area for further investigation.

4.6 Data Presentation

Two types of data presentation are used. First, printed output of results of gradient perturbation and step-size calculations is provided. It is inconvenient to scan many printed pages looking for trends which exist in oscillating parameters. For this purpose, microfilm plots of parameter values and time response values as a function

of the iteration cycle number are provided.

Second, when a program encounters a situation in which the response is detected as being divergent or having low overshoot, the human designer would like to see a plot of the time history of the response showing how poor the response is. This is also provided on microfilm.

4.7 Program Problem Areas

Two primary problems still exist with the operation of the program as it now exists. They are different although quite related in their cause and probable solution. The first problem involves the search to bring an unacceptable time response within specifications. Step-size selection is very critical as the program can jump back and forth between two boundaries, sometimes oscillating from one side to the other of an acceptable portion of the parameter space but never entering it. Two possible solutions are proposed. When the search is close enough to a satisfactory time response that it alternates in violating first overshoot and then rise time, a gradient combining these results could be formed. This will raise a question of weighting since one second of rise time is not equivalent to one percent of overshoot. Alternately, a sequence of past points in the parameter space can be saved. If parameters are oscillating due to the alternative boundaries, the past values can be used to find a mean value of the parameter and use this to modify the gradient.

The second problem occurs when the program has successfully reached one constraint surface and moved along it encountering a second. At this time, the program simply pulls back from the boundaries slightly and re-minimizes GFAR. This sometimes provides further improvement. Certainly it cannot be called a logical procedure. A constrained gradient combining information about normals to both constraint surfaces and the gradient of GFAR is needed.

4.8 Parameter Space Transformation or Scaling

Several times in the preceding discussion, it has been noted that parameter space scaling was found necessary. References dealing with gradient methods note the problems of poorly shaped contours and suggest a linear transformation as a cure. Thus it was not surprising when the following example appeared and forced the introduction of scaling.

There are two parameters involved. The first is 20, the second is 0.1. The program makes a 10% perturbation and the overshoot changes from 25% to 20% for the first and from 25% to 30% for the second. For the first, the gradient component will be $\Delta F/\Delta P = -5/2 = -2.5$. For the second, the gradient component will be $\Delta F/\Delta P = 5/.01 = 500$. Thus the gradient vector is aligned parallel to the second parameter axis. A typical step-size of .1 is used. Then P_1 will become 20.1 and P_2 will be .2. Thus the program is searching along the line $P_1 = 20$. The program

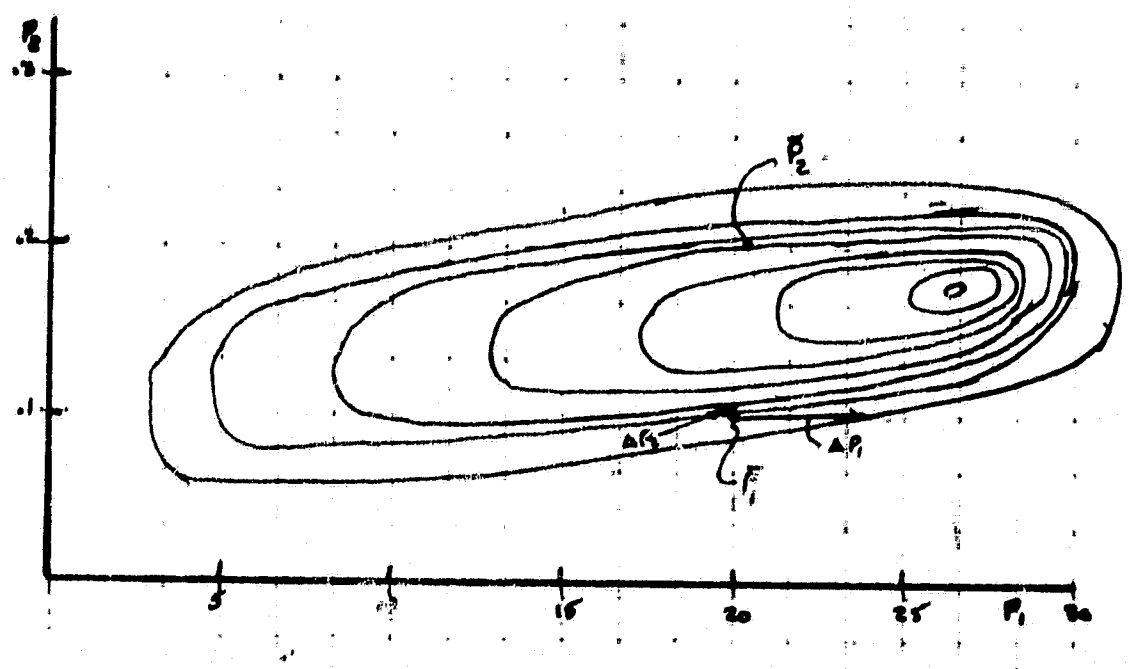
will go to a point along that line where overshoot is minimized and oscillate. But the gradient indicated that a significant improvement could be made by changing P_1, but not in steps of .1.

This situation is illustrated in Figure 4.7(a). We see that a long narrow ridge could exist. The change from $\overline{P}_1$ to $\overline{P}_2$ has jumped across the ridge. The next step will bring $\overline{P}_3$ to a point very near $\overline{P}_1$. There exist sophisticated procedures for putting a point on the ridge and turning the direction of iteration toward the maximum.

However, the gradient program actually does not work with ridges as shown. The zig-zagging occurs because it is working on two different constraint surfaces namely overshoot and rise time, which slope opposite to each other in several axes. The ridge-following procedures do not accept a situation involving two interchanging surfaces. Thus the alternative of parameter space scaling or transformation must be used.

Now consider the change of variables, $P_1' = .05P_1$ and $P_2' = 10P_2$. Then $g_1 = 5/2/20 = 50$ and $g_2 = 5/.01/.1 = 50$. Thus the gradient components are equal, which is proper, since the perturbations had equal results. Next the step-size of .1 is again used. Then $\Delta P_1 = 2$ and $\Delta P_2 = .01$. This means that each parameter takes a reasonable step compared to its absolute value.

The meaning of this transformation is shown in Figure 4.7(b). The contours have been made more circular



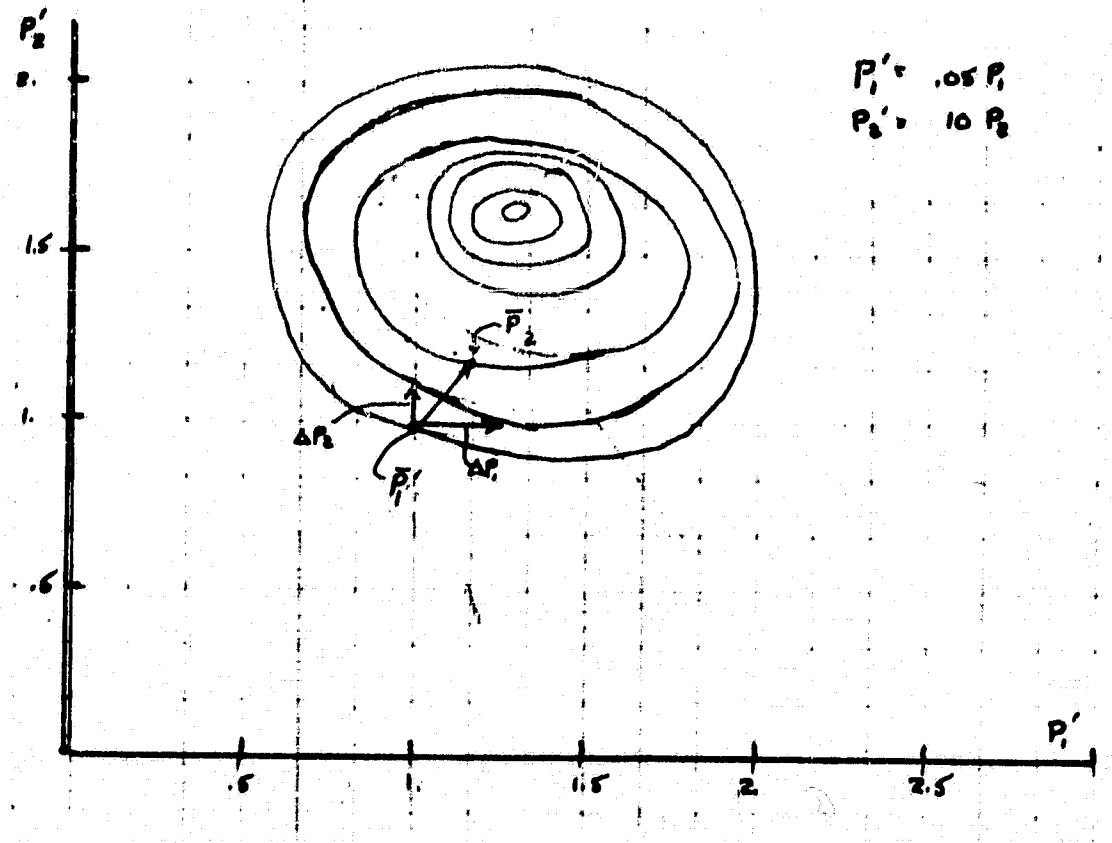


Figure 4.7

Example of Parameter Space Scaling

and more evenly spaced. Now the step from $\bar{P}_1'$ to $\bar{P}_2'$ along the gradient is aimed toward the proper maximum rather than just the ridge.

One might say that this procedure is not necessary; the program should know how to take the step to the ridge top and not go beyond. Further, is it a legitimate operation to alter the contours in the function space? Indeed, the program will follow a different path in reaching the maximum and probably take many less steps in doing so. We seek only the constrained maximum, not a particular route to it.

The scaling is included in the program just as it was demonstrated here. Each gradient component is multiplied by the scale factor before the gradient is normalized. Then, each gradient component is multiplied by the scale factor as each component is updated to the new point of the parameter space.

The modification to equations 4.2.2, 4.2.3, and 4.3.7 given earlier are as follows:

$$g_i = \frac{\Delta F_i}{\Delta P_i} = \frac{\Delta F}{\frac{\Delta P_i'}{P_{s_i}}} = \left(\frac{\Delta F_i}{\Delta P_i}\right) P_{s_i} \qquad (4.8.1)$$

$$\text{or} \quad g_i' = P_{s_i} \cdot g_i \qquad (4.8.2)$$

$$\Delta P_i = \frac{F(\bar{P}_0)\, g_i}{|\nabla F(\bar{P}_0)|^2} \qquad (4.8.3)$$

$$\Delta P_i' = \frac{P_{s_i} \cdot F(\bar{P}_0)\, g_i'}{|\nabla F(\bar{P}_0)|^2} = \frac{P_{s_i} \cdot F(\bar{P}_0) \left(\frac{\Delta F_i}{\Delta P_i}\right) P_{s_i}}{|\nabla F(\bar{P}_0)|^2} \qquad (4.8.4)$$

$$= \Delta P_i \cdot P_{s_i}^2$$

$$|\nabla F_{(\bar{P}_0)}|^2 = \sum_{i=1}^{n} (g_i)^2$$

transformed

$$|\nabla F_{(\bar{P}_0')}|^2 = \sum_{i=1}^{n} (g_i')^2 = \sum_{i=1}^{n} (g_i \cdot P_{0_i})^2 \qquad (4.3.5)$$

Selection of the scale factors is the next problem. Unity can be used until a problem is discovered. The scale factors used in this example simply normalized the results. This can be included in the program, so that the space is normalized automatically which represents a continuously changing scale factor.

In the course of some examples, some initial gains were poorly estimated by an order of magnitude. Thus the scaling, while better than unity, still held back that variable. The variable scale factor would normalize each parameter to unity and allow the scale factor vector to be used for special conditions. The latter is necessary because even with scaling, there are situations, usually involving two boundaries, in which the gradient components are large and of opposite sign. This dominates the gradient and the program just oscillates. Thus, a fixed scale factor would weight this parameter less and let the program move into a region where it would operate more normally. The program could then be restarted with the normal weighting, and a solution obtained. Also, this would allow the program to vary its parameters without

them becoming over or under weighted, as with a single scaling vector.

This problem of scaling can be most irritating as there is little to indicate, prior to wasting some computer time, whether scaling is needed and how to approach it, save simple normalization to unity. Various references on the subject indicate only that a linear transformation is called for, show that one exists, and leave it at that. In this program, the problem is compounded by the fact that four gradients, with respect to overshoot, rise time, GFAR, and a boundary, are in use in every successful program. A scaling which is satisfactory for one surface need not be very good for another.

CHAPTER 5

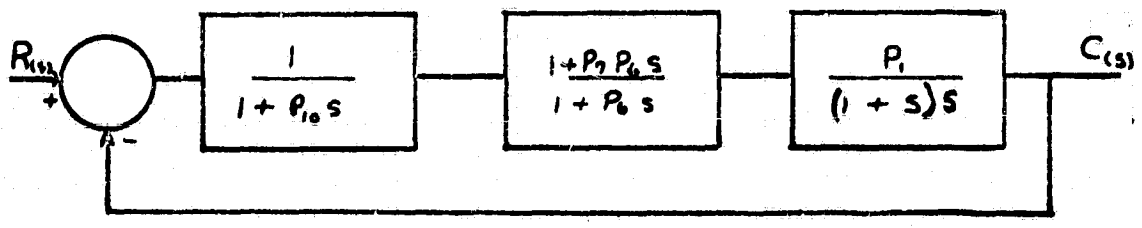
RESULTS OF AUTOMATED DECIGN PROGRAM

5.0 Introduction

This chapter discusses and evaluates the results of the automatic design program. A specific example, from several that were run, is examined. The results are generalized to cover observations from all design examples. The results are evaluated in the sense of determining if successful designs are possible, what new knowledge was obtained, and the future possibilities of the methods.

5.1 Design Example

The configuration is



The plant is a real pole at s = -1.

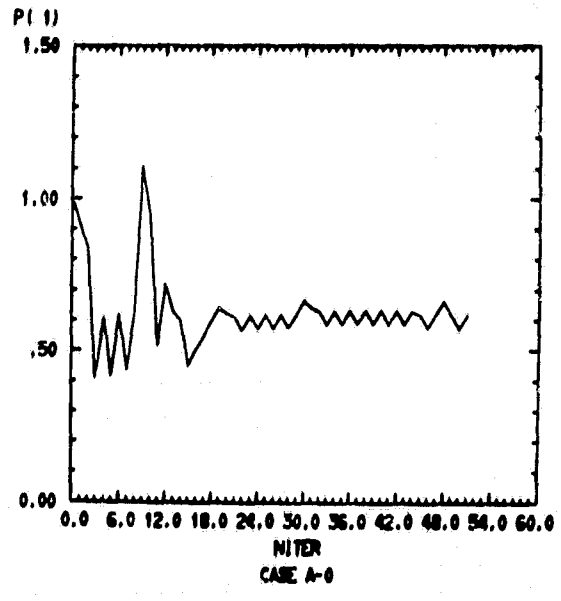
The compensation parameters are P_1, P_6, P_7, P_{10}.

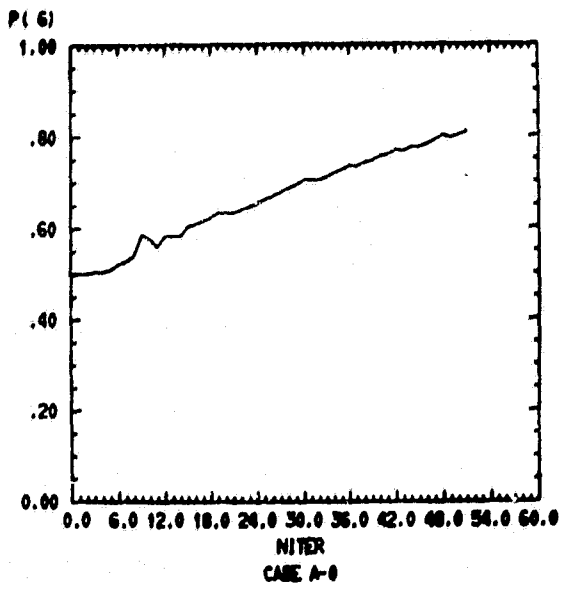
The response required is OS $\leq$ 20%

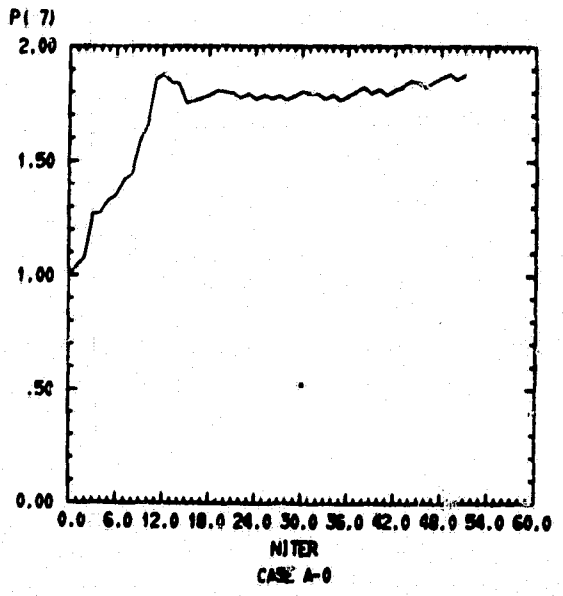
$t_r \leq$ 3 seconds.

A number of combinations of initial values for the compensation were tried. One example is shown in Fig. 5.1. The parameter history for P_1, P_6, P_7, and P_{10} are shown.

The history for overshoot, rise time, and GFAR are shown as XX(1), XX(2), and XX(5), respectively.







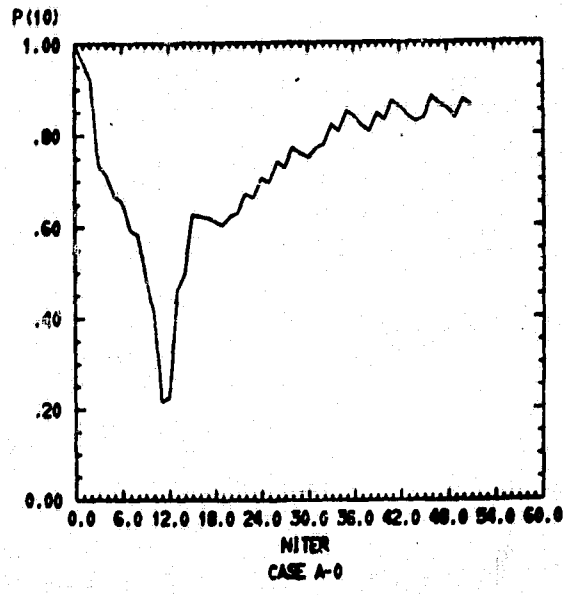
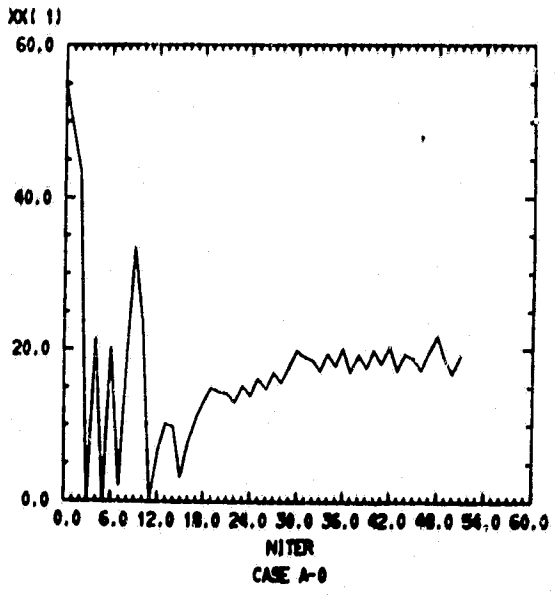


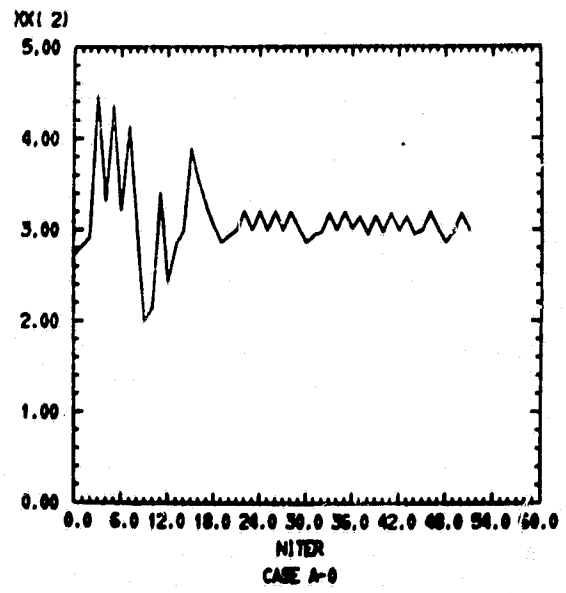
Figure 5.1 a

Parameter Result Plots for a Typical Design

OVERSHOOT



RISETIME



GFAR

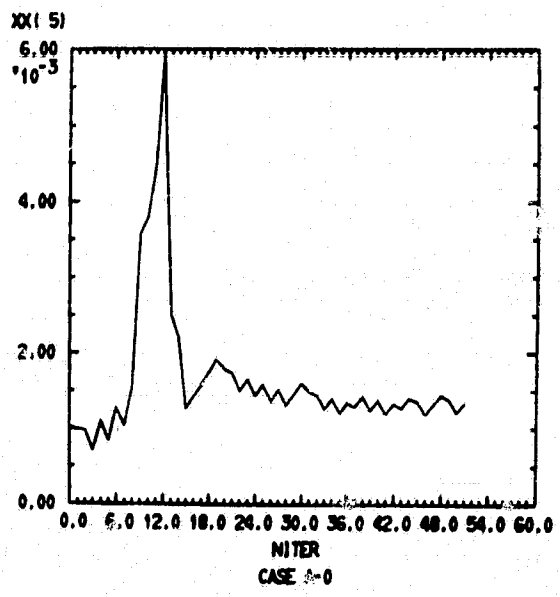


Figure 5.1 b

Time Result Plots for a Typical Design

An acceptable design was achieved at step number 12. GFAR was reduced until the rise time constraint was encountered at step number 14. GFAR was further reduced following this constraint until the overshoot constraint was encountered at step 40. After this, GFAR continued a gradual downward trend.

5.2 General Results

The program was tried with a variety of design problems. Two types of plants were used, one having a fixed real pole and the other having a fixed complex pair. The compensation utilized real poles and zeros which could be placed in either the forward or feedback paths. This provides four combinations of plant and compensation. One further combination which was not used much involved a complex pole pair plant and compensation of complex zeros in the feedback and other real poles.

The results of the program can be summarized as follows.

a) The iterative design program can take a system with a stable response, improve the response until it meets time response specifications of maximum overshoot and rise time, then proceed to minimize the high-frequency transmission until a time response constraint is encountered. It then follows a constrained gradient reduc-

ing further the high-frequency transmission until the other performance limit is reached.

b) When confronted with an initial parameter set whose time response is unstable, the gradient procedure will find a stable parameter set if real lead compensation is adequate. If a conditionally stable system is present, a stable initial design is generally required.

c) It can successfully perform the gradient search using open-loop compensation parameters. It could successfully allow constraints on these parameters. It could take test cases with parameter choices intentionally given poorly, such as zero preceding the pole for lag compensation, and change this to a satisfactory design.

d) When final design values were tested via the QD017 program, the residue of the closest real pole in the closed loop transmission was not negligible. This indicates that design to time domain specifications allowed reduction of high frequency transmission beyond that if a dominant second order response specification had been used for the time response.

e) The program showed that parameter space transformation is generally required for good re-

sults. No conclusions were reached on procedures for determination of the transformation.

f) The program is very sensitive to step-size selection. The function surfaces considered do not have valleys or ridges in the ranges normally considered. However, valleys are created by the intersections of two constraint surfaces with opposing effects. These valleys can lead to jumping back and forth in the same sense as the large step size path on a sharp ridge. Parameter space scaling improved operation in these situations but it remains a problem.

g) The program works reasonably well in following the constrained gradient along a constraint surface. The question of proper step-size along the constrained gradient is not resolved. One would expect the program to operate best by following the boundary closely. This was not borne out by experience. When large steps were taken, the path deviated from the constraint surface considerably. Yet these paths reached a smaller value of GFAR after a given number of iterative cycles than those which followed the boundary closer. There are several interacting

effects in this operation which must be separated to find the true cause and its solution.

h) The program verified that certain compensation parameters do not have a large effect on the design. Conclusions here are risky because a different scaling procedure could change the results. It was found that the separation of a pole-zero dipole was more important than its location, particularly in lag compensation. This was predicted earlier.

The program also indicated that poor scaling can cause designs that are nonsense when examined in a classical control sense. For instance, one scaling procedure caused the pole to come at a lower frequency than the zero when the dipole was placed near the crossover frequency for lead compensation. The high frequency transmission was minimized and the performance specifications satisfied, yet this dipole is backwards.

i) Designs were made with a real zero in the forward path and then in the feedback path. In the first case, the zero appears in $T_{(s)}$, in the latter it does not. For the same initial parameter and the same time response specifications, the case with the real zero in $T_{(s)}$ lead

to a smaller value of GFAR than the case without.

j) Comparing the results of designs for real pole plants and for complex pole plants, it can be said the latter are much more difficult. The phase is changing more rapidly near the cross-over frequency. This causes slight changes in compensation values to result in larger changes in response values, particularly overshoot, than for real pole plants. When the complex plant poles were placed on the imaginary axis, the search was still successful but did much wandering around in maintaining satisfactory time response.

5.3 Evaluation of Results

In evaluating the results, the program has shown that the design of control system in the classical sense for fixed plants can be done automatically by computer search methods. It has shown that the design can be made to specific time response requirements without approximation or conversion to s-plane or frequency domain specifications. Regarding the choice between manipulating open loop parameters versus closed loop parameters, the program successfully operated in the open loop space. Plant parameters were introduced directly and limits on compensation variables were successful.

The design examples showed that minimum noise transmission occurred when both time domain specifications were just satisfied. This indicates that enough variable parameters must be available to fit all non-conflicting specifications. Further the minimum noise transmission occurred with configuration placing the real compensation zero in the forward path.

The final designs have real compensation poles in close enough that the closed loop response is no longer dominant second order. Thus the best designs involved a third order $T_{(s)}$ with a real zero which means four variable parameters. It would be difficult for a human designer to scan through families of curves of responses attempting to select ones which fit his time specifications and minimized some other function.

From the standpoint of numerical methods, the program demonstrates that a gradient search with inequality constraints in the function space and the parameter space is possible. It indicates that problems associated with gradient searches of poorly shaped and interacting surfaces are present in this program and can be quite serious. It generally shows that as the problems become more difficult, the initial guesses must be better.

As a general evaluation, the program shows that it is possible to design control systems in the same sense that frequency response and root locus methods have been

used. It has not replaced the human who must provide good initial parameters, check for conflicting specifications, etc.

For the program to have a real future as a design tool, several problem areas must be solved. These are:

1. The areas of parameter space scaling and step-size selection need improvement to speed up the procedure.
2. Gradients from both the constraint surfaces need to be combined and used when achieving a satisfactory design and when following a boundary. This should reduce the problems of interacting surfaces.
3. A faster method of generating step response results is needed.
4. Interactive console features to allow the designer to monitor the design progress and modify the search would be desirable.

CHAPTER 6

DESIGN PROCEDURES FOR SYSTEMS HAVING PLANT PARAMETER VARIATIONS

6.0 INTRODUCTION

A fundamental property of feedback is its ability to reduce the effects of plant parameter variations on the input-output system response function T(s). This property is almost always cited to justify the use of feedback, but it is rarely used directly, in a quantitative manner, in the design process. Recently, such synthesis procedures, including specifications on parameter variation, have been presented.[1,2] However, these procedures are limited in a very important aspect, in that T(s) is restricted to be dominantly second order. This limits flexibility in design and to the plant types for which the procedure is applicable. Also, even for applicable plants, the dominance requirement results in a restricted form for the loop transmission function and thereupon on the system sensitivity to internal noise.

In this paper, computer methods are first used to generate root loci to aid in the understanding of parameter response interaction. Then a new design procedure is developed in which open loop parameters are directly related to step response specifications. Computer simulation of the system model provides information required for this purpose. A

complete design example, including comparison with previous methods, is presented.

6.2 STATEMENT OF PROBLEM

This chapter considers the same basic problem of [4], namely: 1) There is a single input-output plant with parameters which may lie (or "slowly" vary) within a given region of parameter space; 2) specific bounds on, for instance, the step response are prescribed, such as acceptable range of rise-time, overshoot, and settling time; 3) linear time-invariant compensation is to be used for which the rms effect at the plant input of noise, lumped at the sensor, is to be minimized. The three sections of this definition have historical backgrounds requiring explanation.

First, in the pole-zero approach, the definition of region of parameter variation has to date been restricted to definition of a region in the s-plane, where plant poles and zeros may lie, normally considering only a single complex plant pole, and a gain variation. No attempt is made to correlate pole-zero-gain variations to actual physical parameters producing the variations. Second, the definition of acceptable response, while broadly stated as ranges of rise-time, etc., is usually reduced to a definition of a region in the s-plane in which the dominant complex poles of the system response function T(s), must lie. No zeros in T(s) are allowed and all additional poles are required to be "far-off," i.e., their residues must be small. Third, the design procedure leads to high open-loop gain and second order differentiation

over some frequency range. This increases the possibility that sensor noise may cause saturation of some components of the control system. The definitions of configuration and transmission are as follows:

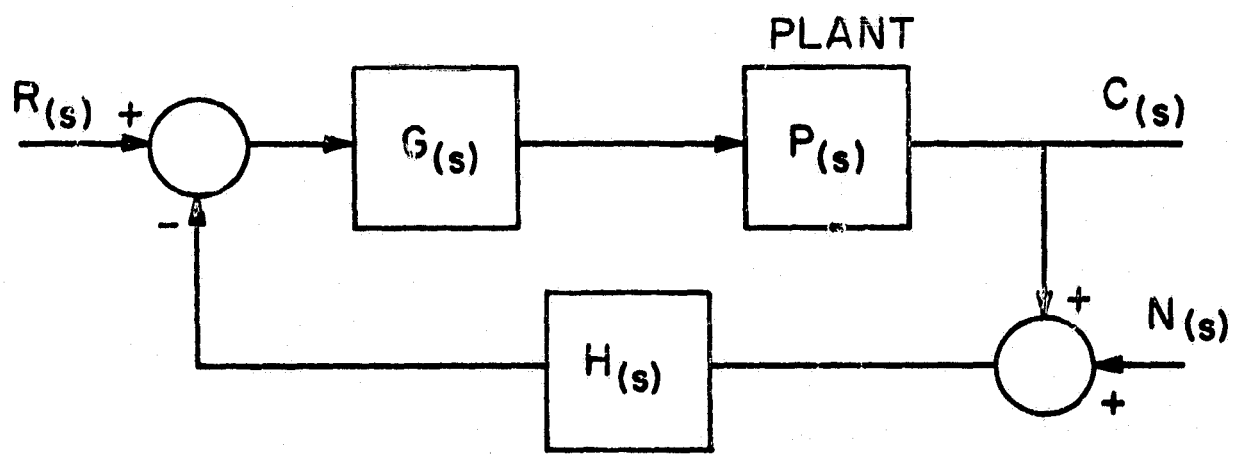


Figure 6.2

A two-degree-of-freedom system structure.

$$L(s) = PGH(s) \text{ and } T(s) = \frac{C}{R}(s) = \frac{GP(s)}{1+GPH(s)} \tag{6.1}$$

For the particular example used in this paper, the plant and compensation transfer functions are defined as follows:

$$G(s) = \frac{KP_1}{s+P_1} \tag{6.2}$$

$$P(s) = \frac{k}{s((s+\alpha_p)^2+\beta_p^2)} \tag{6.3}$$

$$H(s) = ((s+\alpha_z)^2+\beta_z^2) \tag{6.4}$$

The high frequency (noise) transmission will be denoted GFAR and is equal to KP_1 (equation 6.2).

A transfer function in Bode form is defined as $1/1+\tau s$ and one in root locus form is defined as $1/s+a$.

6.3 STUDY OF PROBLEM LEADING TO DESIGN PROCEDURE

The basic approach has been to introduce compensation zeros in the feedback path and to force the closed loop poles closeby, by means of large loop gain.[20] Previous methods[4,3] employ this concept, with mapping of the open-loop parameters into a closed-loop parameter space as the principal design tool. This chapter uses the same concept but, among other differences, works directly in the open-loop parameter space. The objective is to find compensation-gain values which satisfy actual time response (not intermediate dominant s-plane) specifications, while minimizing noise transmission GFAR, which is equal to KP_1. It is emphasized that while step response parameters such as overshoot and rise-time are used here, any other response requirement may be imposed, so long as they may be evaluated by a computer simulation routine. In this chapter a fourth order model is used in order to correlate the results with previous work. But this is not a restriction in the method. The design is initiated by investigating parameter-response interaction and then utilizing the information to develop the significant parameters for the computer-aided design.

Performance Analysis by Root Loci

The basic concept of using high gain to drive wandering poles to complex zeros located near the desired closed loop poles, and the costs involved, have been known for some time.[4] The designer can easily visualize possible root locus patterns which fit this concept. However, when specific numerical problems are investigated one will likely find that the root loci are not simple after all and it is useful to obtain some detailed insight, which is now done.

The design problem including numerical values of [4] is examined. The plant

$$P(s) = \frac{1}{s((s+\alpha_p)^2+\beta_p^2)}$$

may have values of α_p, β_p anywhere within the rectangle ABCD of Figure 6.2. The compensation zeros were determined to be $\alpha_z = 2.0$, $\beta_z = 3.6$ and the gain K = 35.2. The plant gain k is taken as unity. The specifications required that the dominant closed-loop poles lie inside the dotted region of Figure 6.3. This range is approximately that which has been suggested as acceptable in flight control. The real pole P_1 is placed at -30 to make the system fourth order. Six loci are shown for several critical plant pole values.

The loci for all corners and point E go to the compensation zero after breaking away from the real axis. For corners A, B, and C, the loci on the real axis is a simple combination of the pole at the origin

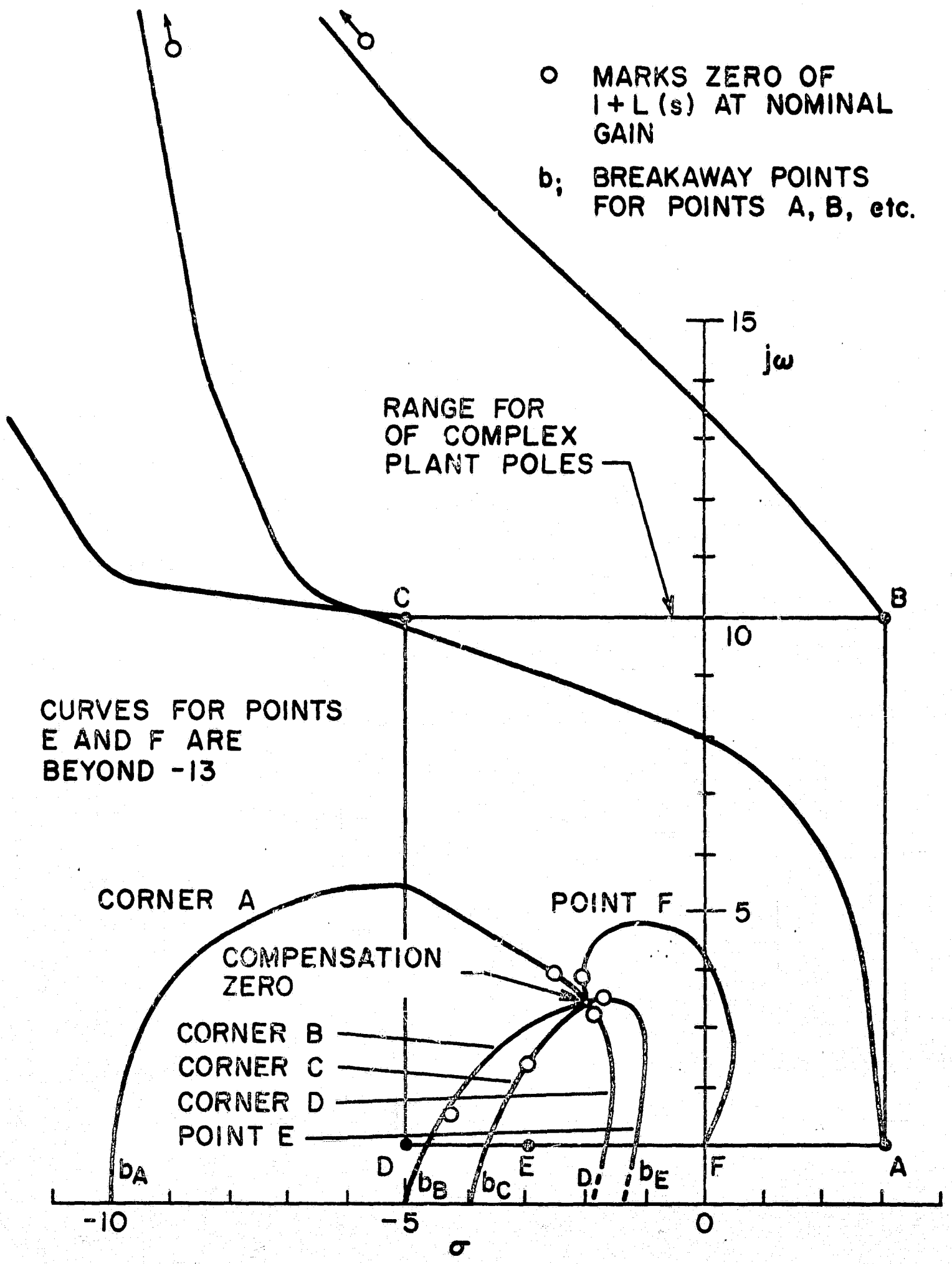
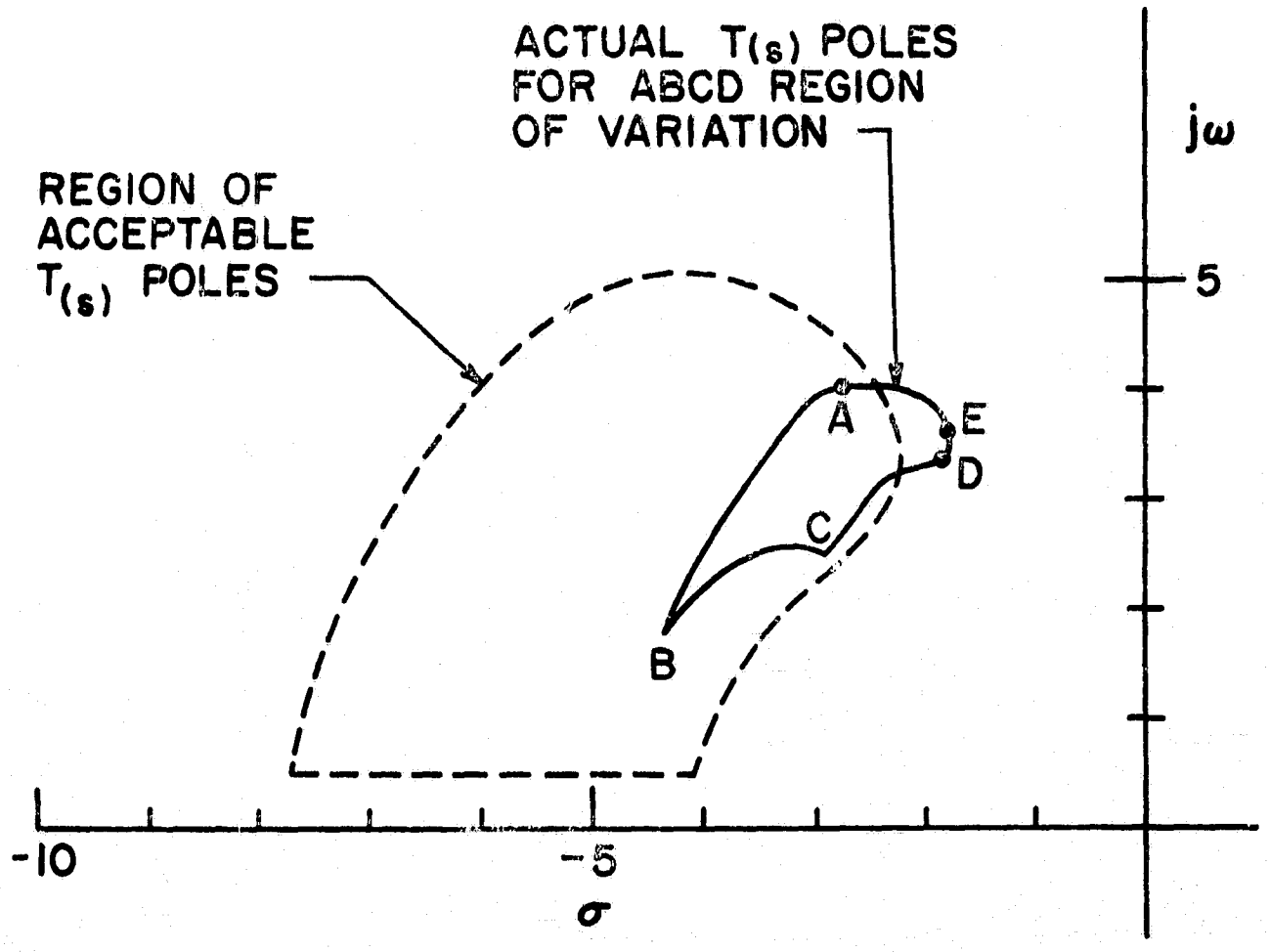


Figure 6.2 ROOT LOCI FOR ORIGINAL DESIGN VALUES



$K = 35.2,\ \alpha_Z = 2.,\ \beta_Z = 3.6,\ P_1 = 30$

ACTUAL RESPONSE VALUES

CORNER	% OVERSHOOT	RISE TIME - SEC	SETTLE TIME - SEC
A	12.8	.482	.96
B	.04	.776	.776
C	1.8	.804	.804
D	18.4	.574	1.28
E	21.3	.517	1.23

Figure 6.3 MAPPING OF REGION OF VARIATION TO CLOSED LOOP $T_{(s)}$ PLANE, DOMINANT ROOTS ONLY.

and P_1 and the plant poles are translated into far-off poles. The plant poles for D and E go to the real axis, split, and combine with the pole at the origin and P_1 to form two complex pairs. Fortunately, this occurs at a gain significantly below that normally needed. For point F, the plant pole goes directly to the zero. Of particular note is that the loci for a particular plant pole reach the compensation zero from the opposite side in almost all cases. The real pole P_1 is significantly involved in all these loci and this emphasizes the need for using at least a fourth order representation for the system.

Since the dominant poles are quite close to the compensation zero at the nominal design gain for points A, D, and E, the location of the compensation zero has a predominant effect on pole location. These points produce the poles with the smallest damping ratio and thus determine maximum overshoot. The other corners, B and C, have dominant poles with much greater damping and likely determine the maximum rise-time. The dominant roots for point A have the combination of largest ω_n and small damping ratio and thus likely determine the minimum rise-time. Changes in zero position basically tend to pull the dominant poles along with the zero, at nominal gain. Moving P_1 in closer to the origin tends to drive the poles further along the loci and force far-off poles toward the imaginary axis.

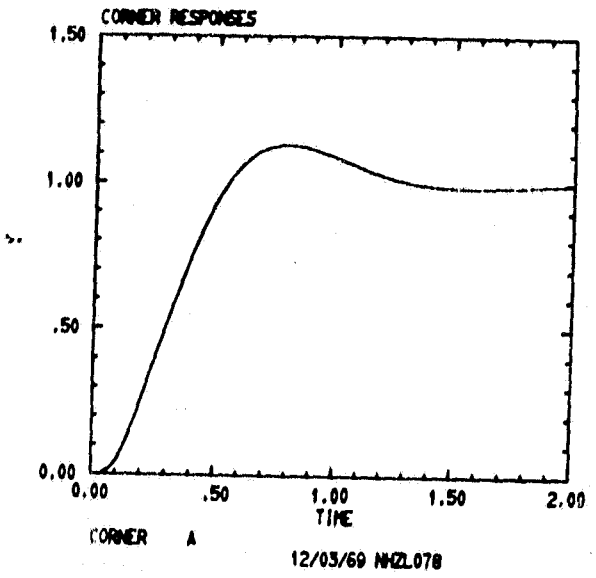
The last point must be considered carefully and it is appropriate to mention a problem area. In [3,4], all plant and compensation poles are condisered in root locus form. Since P_1 is in some sense

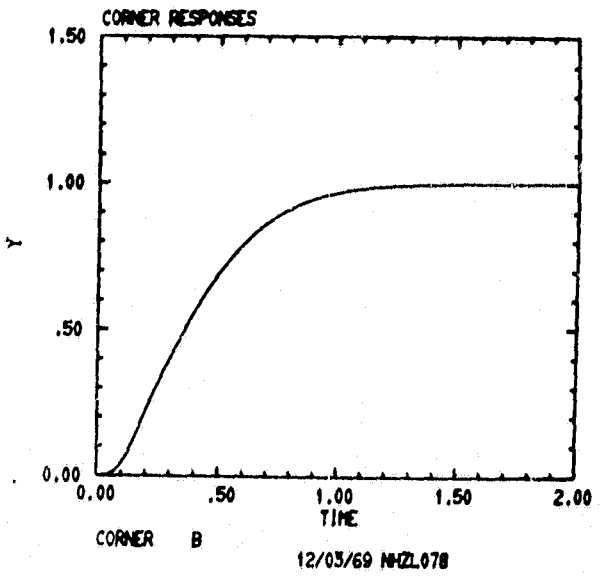
a filter pole, it could also be considered in the Bode form (i.e. as in equation 6.2). In this case, a change in P_1, or rather 1/P, changes the root locus gain also. Furthermore, if the plant complex poles are represented in Bode form, their movement within the ABCD square changes the root locus gain. One must determine if plant pole changes are correlated with plant gain changes and the appropriate form selected.

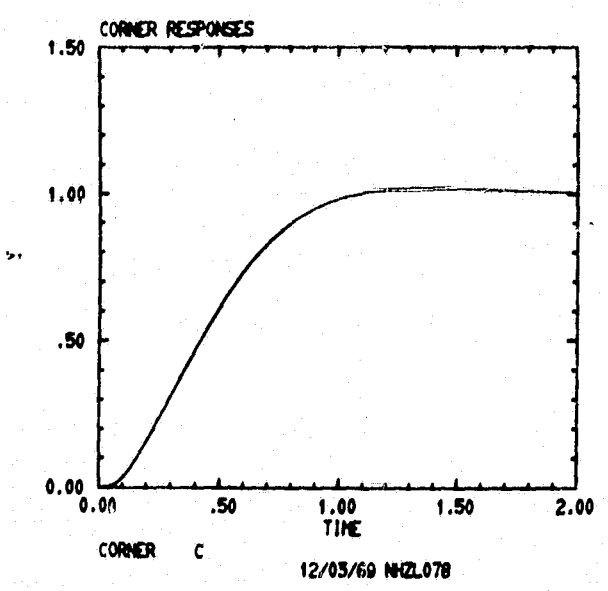
In [1,2], after the design was effected in the X-Y plane, it was desirable to map the boundary of the open-loop ABCD region into the closed-loop s-plane. Compensation is fixed and the mapping shows how well the region of variation fits the allowable region. This is the same as connecting all the nominal gain points on the root loci of Figure 6.2. This mapping is shown in Figure 6.3 for later reference. Figure 6.4 shows the actual step responses for points A, B, C and E, which are all well-behaved. The above information, together with that in the next section, will be used later to guide the design procedure.

Analysis by Constant Performance Contours

A distinctive new feature of this chapter is the design directly to time-response specifications, rather than to intermediate s-plane or frequency response specifications. This is done by computer simulation of a system model to provide the step response time history. Given the system structure along with plant and compensation parameter values, a computer program returns the values for the applicable time-







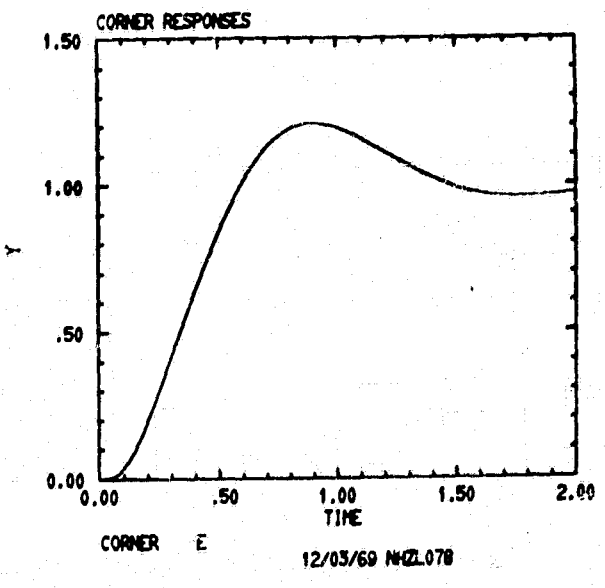


Figure 6.4

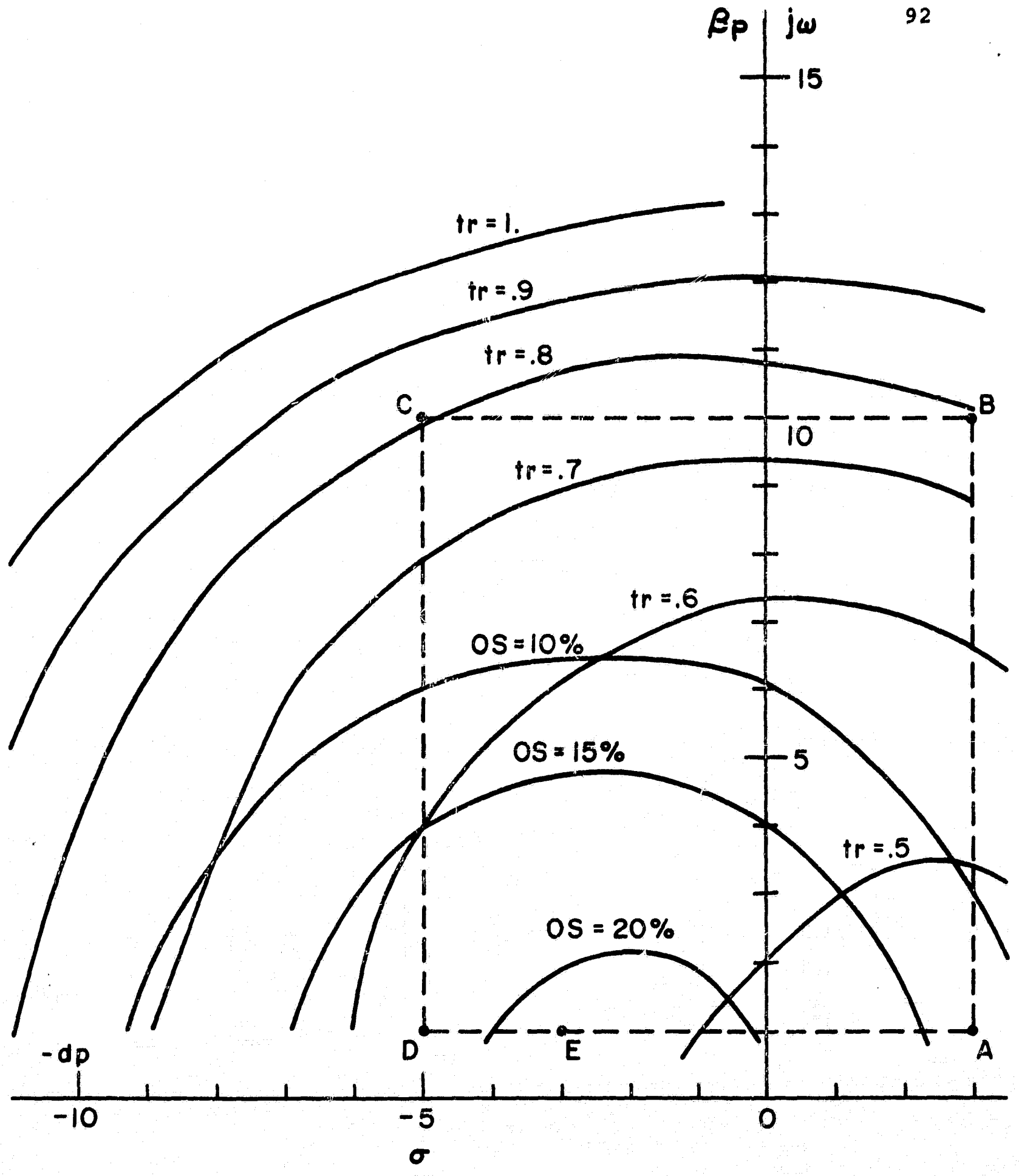
Step Responses for Original Design

response parameters -- in the present case, overshoot, rise-time, and settling time. The computer program is sufficiently flexible to permit change of system order and configuration. It may be mechanized on an analog/hybrid computer simulating transfer functions, or on a digital computer by solving differential equations or by inverse Laplace transformation. The procedure selected is not important to the analysis method as such, but involves economic considerations of computer time costs.

The data presentation consists of a set of curves of constant performance (e.g. rise-time, tr=1 sec) shown in the plane of the variable plant complex poles as in Figure 6.5. Thus, the curve labelled tr = 1 is the locus of complex plant pole values for which the present design step response rise-time is one second, etc. It is seen that the plant variation region ABCD is located inside the curves for OS $\leqq$ 25% and .45 sec $\leqq$ tr $\leqq$ .8 sec. Note that corners B and C correspond to maximum rise-time, corner A to minimum rise-time, and point E (not a corner) to maximum overshoot on the boundary.

Since the designer knows what plant conditions correspond to particular locations in the ABCD region, he may determine if the response is satisfactory for that condition. This allows a certain flexibility in applying the specifications and it is evident how much additional parameter variation beyond the ABCD region is allowable.

For comparison purposes, the region of acceptable T(s) dominant poles from [4] (Figure 6.3) can be converted roughly to the time



ORIGINAL DESIGN: $K = 35.2$, $\alpha_Z = 2.0$, $\beta_Z = 3.6$, $P_1 = 30$

Figure 6.5 LOCI OF PLANT POLE POSITIONS ($\alpha_p + j\beta_p$), FOR WHICH RISETIME (tr) AND OVERSHOOT (OS) ARE CONSTANT. (PLANT POLES IN ROOT LOCUS FORM)

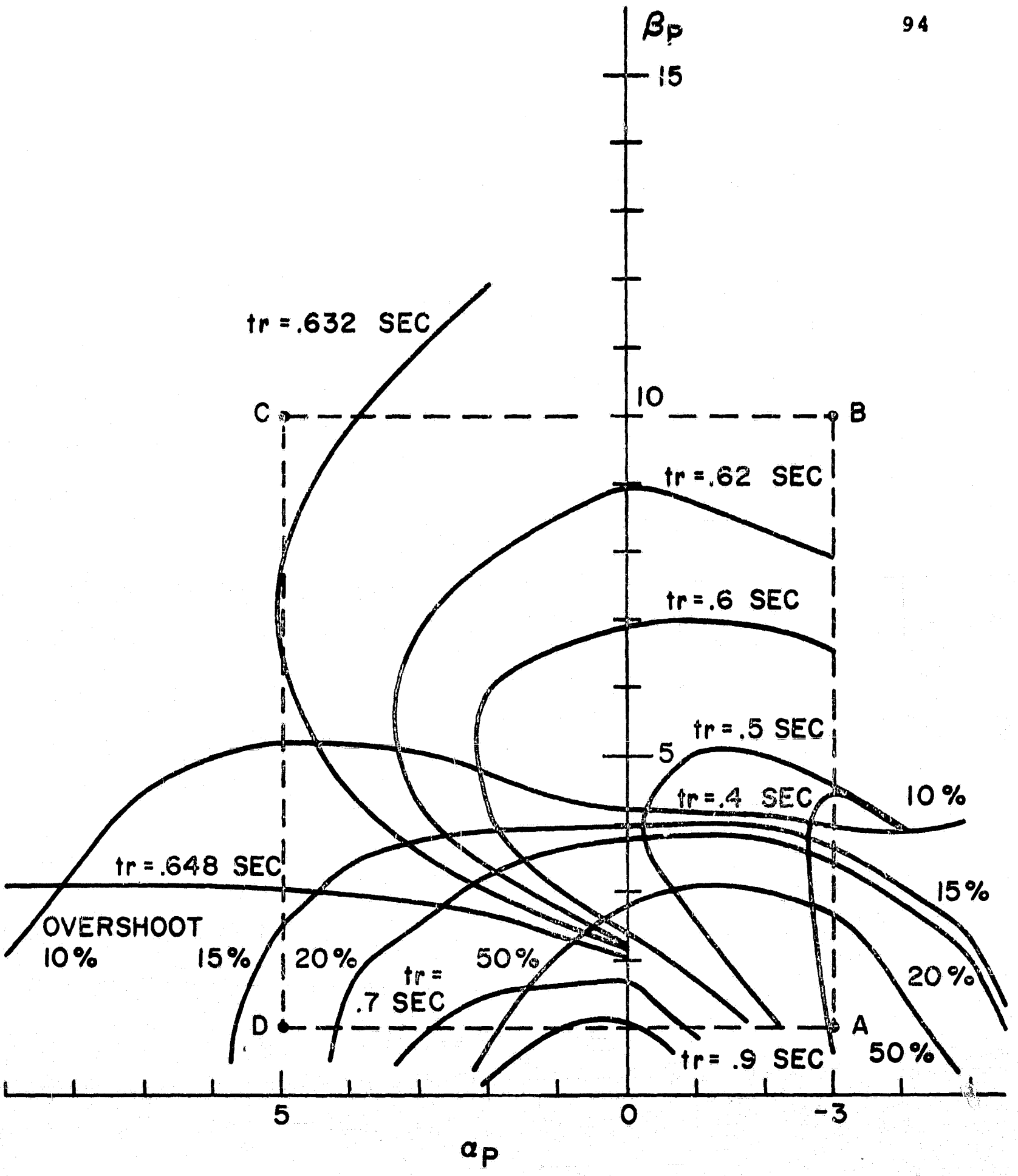
domain specifications of overshoot $\leqq$ 15% and .4 sec $\leqq$ rise-time $\leqq$ .9 sec. Thus it is seen that this design has better rise-time than necessary but it does not satisfy the overshoot specifications. This correlates directly with the information already known about this design from [4] and Figure 6.3.

The computer program which provides the curves of constant performance also calculates the response at the points A,B,C,D, and E, and provides the microfilm plots of the response. It further calculates the closed-loop poles to provide correlation to the earlier methods.

Finally, curves of constant performance for the same problem, but with the plants poles in Bode form, are shown in Figure 6.6. These are not nearly as well-behaved as in the previous formulation. The proper representation, Bode or root locus, cannot be selected without reference to a specific physical plant. However, the Bode representation leads to a more difficult design problem.

6.4 DESIGN PROCEDURE

The design procedure consists of repeated application of analysis with different values for the compensation. This can be extremely ineffective if the correlation between compensation parameter changes and performance changes is not understood. The design begins with a tentative complex zero location, chosen by taking a dominant second order T(s) pole which satisfies the performance requirements and placing the zero closeby. The pole P_1 should be placed about a decade to the



PLANT POLES IN BODE FORM
ORIGINAL DESIGN: $K = 35.2$, $\alpha_Z = 2.0$, $\beta_Z = 3.6$, $P_1 = 30$

Figure 6.6 LOCI OF PLANT POLE POSITIONS ($\alpha_p + j\beta_p$) FOR WHICH RISETIME (tr) AND OVERSHOOT (OS) ARE CONSTANT.

left of the zero. An initial value of gain can be determined from a quick root locus using corners B and D for the plant. If desired, the procedure of [4] could be used for the initial design values.

This establishes a reference design in which the curves of constant performance will enclose the region of variation or be relatively close. The compensation must now be varied to find the best design. Best is defined as having the lowest high frequency loop gain (GFAR) while meeting time response specifications. This design should use as much as possible of the allowable acceptable region in the sense of Figure 6.3.

Perturbation Procedure

This problem is now in the form of an optimization problem with constraints. Gradient methods have been used successfully to solve such problems. Each parameter is perturbed in turn by a small amount ΔP_i and in the resulting change in the performance index ΔQ_i is noted. The ratio $\Delta Q_i/\Delta P_i$ is a component g_i of a gradient vector for reduction of Q. This procedure does not work well for this problem due to difficulties in evaluating effects of the constraints. However, the perturbation procedure has the orderliness required to successfully manipulate several parameters in such a problem. Rather than define specific formulae for evaluating Q and reducing the process to a computer program, the results are evaluated by the human designer who then determines the next choice for compensation parameters.

For the designer to do this successfully, he must know the effect of a given perturbation on the curves of constant performance

and high frequency transmission (GFAR). These are as follows where gain, α_z, β_z, P_1 are defined in Equations 6.2 to 6.4.

1. gain - a) directly proportional to GFAR

   b) an increase drives the roots closer to the compensation zero. If the zero is inside an acceptable region and the nominal gain is beyond the minimum damping ratio point for point E, an increase should increase the area between OS and Tr_{max} curves. There is a tendency for the Tr_{max} curve to move further.

2. α_z - a) no effect on GFAR in root locus form.

   b) an increase generally moves OS curves down without changing Tr_{max} curves very much.

3. β_z - a) no effect on GFAR in root locus form.

   b) an increase generally moves both OS and Tr_{max} curves up.

4. P_1 - a) no effect on GFAR in root locus form and directly proportional in Bode form.

   b) little direct effect on OS or Tr_{max} in Bode form, a decrease of P_1 forces far-off poles closer to the imaginary axis which causes a problem discussed later at corner B.

These cause-effect relations correlate with the root locus analysis earlier.

The specific amounts of change are evaluated by the perturbation process. Each parameter should be changed in turn from its nominal value by say 10%. The resulting changes in the rise-time, overshoot, etc. curves are evaluated. This provides qualitative information about the gradient components. The quantitative evaluation is much more difficult. First, what is the equivalence between percent overshoot and seconds rise-time? Second, scaling or normalization problems well-known in gradient search methods enter here also. Third, if exact gradient information is to be used, a good method for step-size selection is needed. These three problems involve the designer in difficulties not directly related to the basic control problem and are best avoided. The qualitative information only is used to establish gradient component sign and approximate magnitude. A small change in parameters is used to avoid drastic changes in performance. As the designer gains confidence in his ability to evaluate the gradient and select step sizes, the process can be made more precise with improved efficiency.

The design process can best be described with an example. While the comments are directed mostly to this method, the method itself cannot be isolated from root locus and other techniques which influence the designer's choices for new parameter values.

Design Example

The design begun in [4] is continued. The first objective is to obtain parameter values satisfying the performance specifications. The second objective is to reduce GFAR, if possible. Figure 6.7 shows perturbations of K, α_z, β_z, and P_1 (Bode form) from the original design values. The slight rise in the overshoot curve for increased K is due to roots near the ζ_{min} point at nominal gain. Following the qualitative concepts of the previous section, these perturbations indicate that a possible next choice of parameters could well be:

$$K = 35.2$$

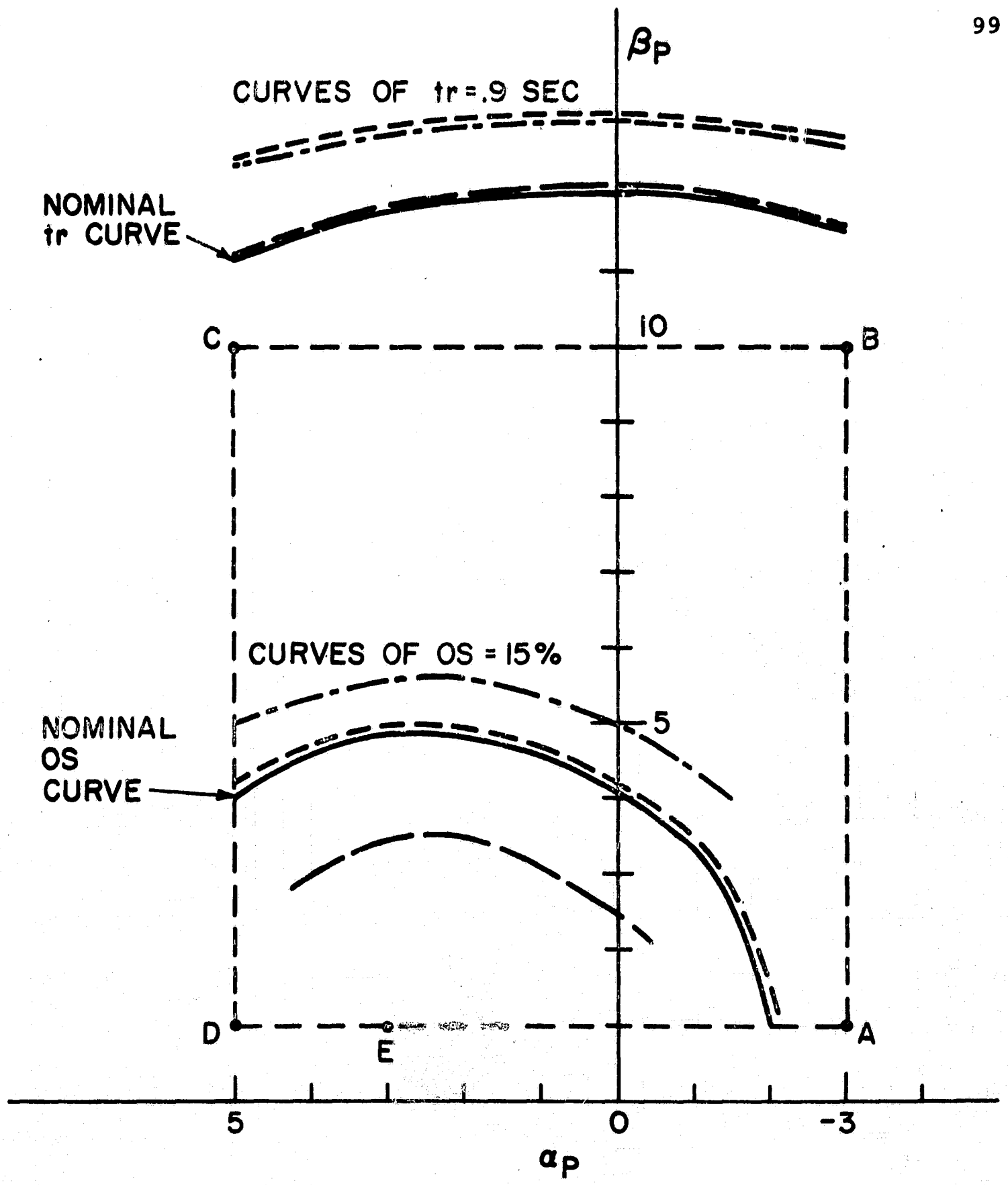
$$\alpha_z = 2.5$$

$$\beta_z = 3.6$$

$$P_1 = 3.0$$

This should move the OS and Tr_{max} curves down to enclose more of the desired ABCD region. In such initial parameter changes, the zero location should be used to position curves, gain should be increased to increase the region between curves only when found necessary, and P_1 should be left fixed until a satisfactory gain-zero combination is found. In the above, the perturbation of β_z suggests it should be reduced. Subsequent perturbations would show this to be the wrong choice.

The procedure is repeated until a combination is found which places both OS and Tr_{max} curves outside ABCD. In this example, such a combination is shown in Figure 6.8 denoted intermediate design.



—— NOMINAL: K = 35.2 α_Z = 2.0 β_Z = 3.6 P_1 = −30

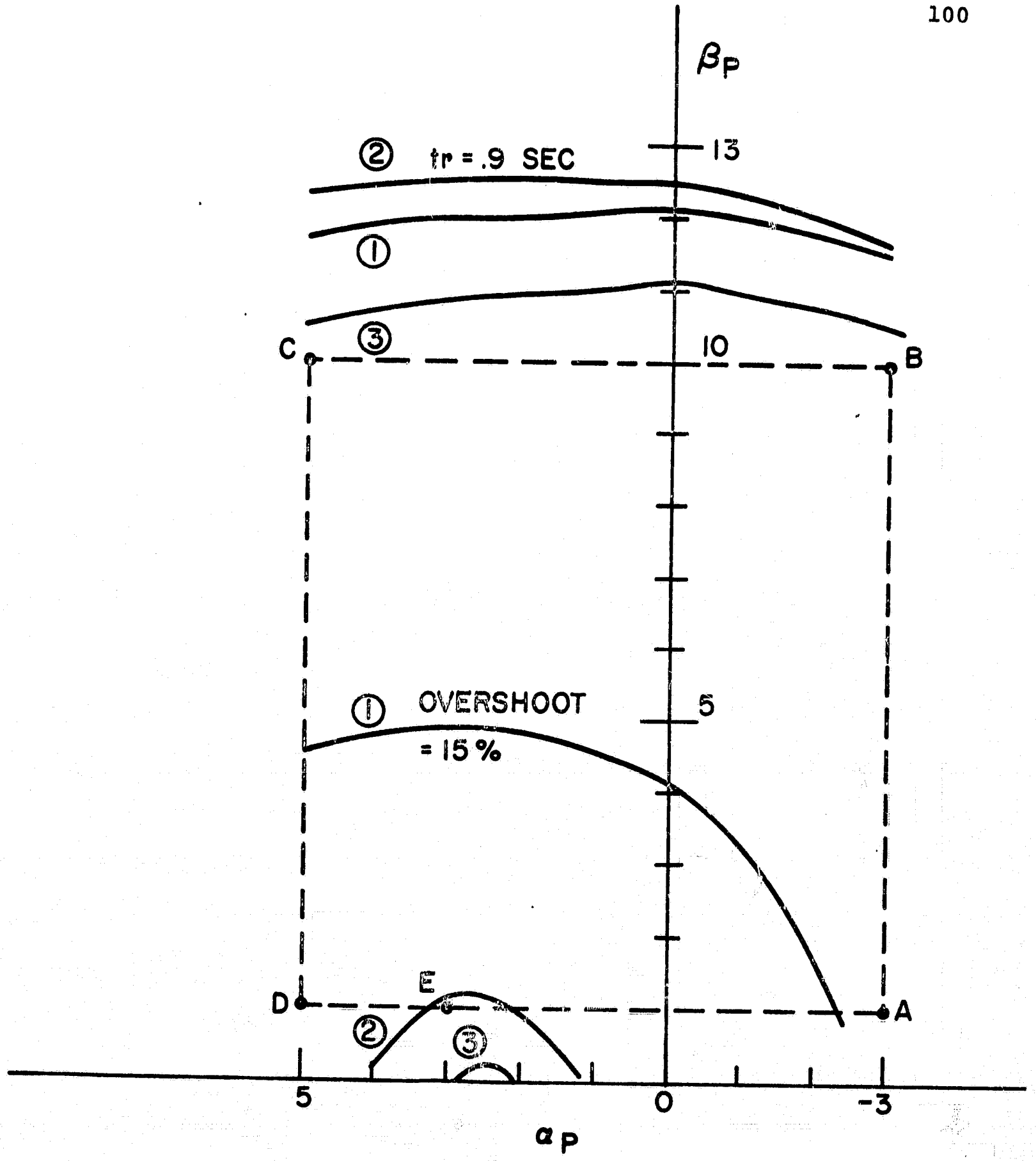
– – – K = 41, α_Z, β_Z, P_1 NOMINAL

— — α_Z = 2.2, K, β_Z, P_1 NOMINAL

–·– β_Z = 3.8, K, α_Z, β_Z NOMINAL

SAME AS NOMINAL P_1 = 20, K, α_Z, β_Z NOMINAL

Figure 6.7 CURVES OF CONSTANT PERFORMANCE FOR K, α_Z, β_Z, P_1 PERTURBED FROM NOMINAL VALUES.



① ORIGINAL DESIGN $K=35.2,\ \alpha_Z=2.0,\ \beta_Z=3.6,\ P_1=30$

② INTERMEDIATE DESIGN $K=32,\ \alpha_Z=2.6,\ \beta_Z=3.8,\ P_1=20$

③ FINAL DESIGN $K=22,\ \alpha_Z=3.0,\ \beta_Z=4.0,\ P_1=25$

Figure 6.8 CURVES OF CONSTANT PERFORMANCES FOR ORIGINAL, INTERMEDIATE, AND FINAL DESIGNS.

GFAR has been reduced from 1060 to 640 in the process. The step responses in Figure 6.9 show small wiggles indicating far-off poles entering the response.

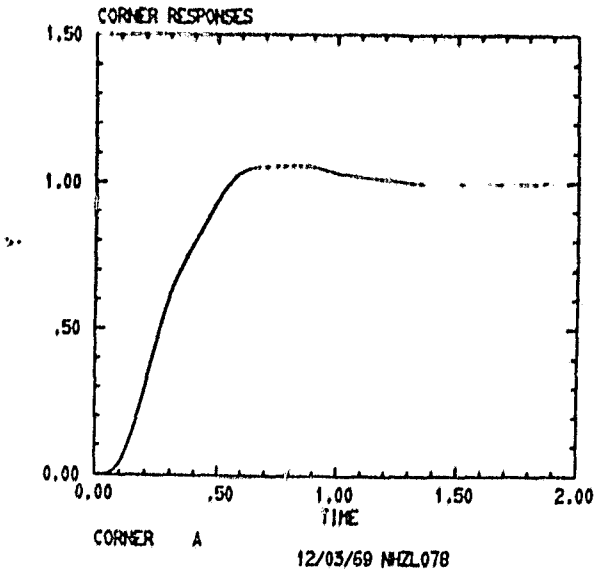
A pertinent question at this point is whether the design can be improved. As yet the curve for constant Tr_{min} has not been shown. It is off the lower right of corner A and difficult for the computer program to locate. But since it does not touch the ABCD square there remains some freedom for varying compensation parameters. The procedure will be to again perturb K, α_z, β_z, and P_1, using reductions in K and P_1 to reduce GFAR, and using α_z and β_z to maintain acceptable performance. Moving the zero up and left should generally bring the Tr_{min} curve up toward corner A. Gain can then be reduced to bring all three performance curves in contact with the ABCD region.

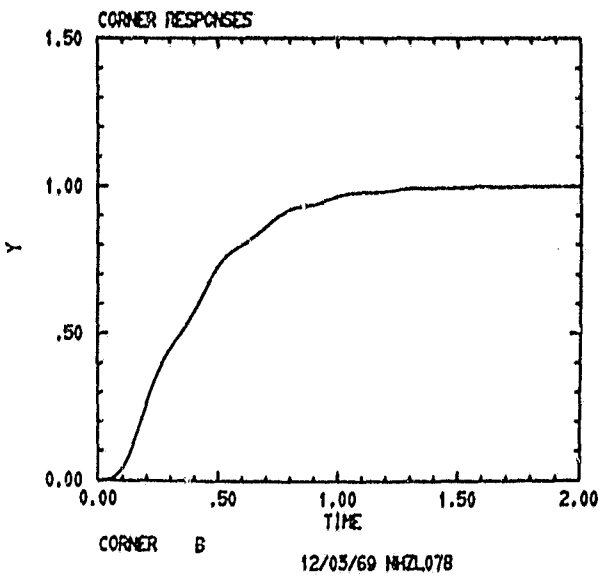
The results of this further work are shown in Figure 6.8 denoted Final Design. Since the Tr_{min} curve cannot be determined easily, the rise-time at corner A is used directly instead. The final time responses are shown in Figure 6.10. The results of the original, intermediate, and final designs are summarized in Table 6.1. On the final design, note that although the OS = 15% curve is significantly below the DEA line, overshoot at point E is 14.5%. This means that a small change will move this curve considerably. No attempt was made to fit the specifications more closely. The outstanding result is that GFAR has been roughly halved from its original value.

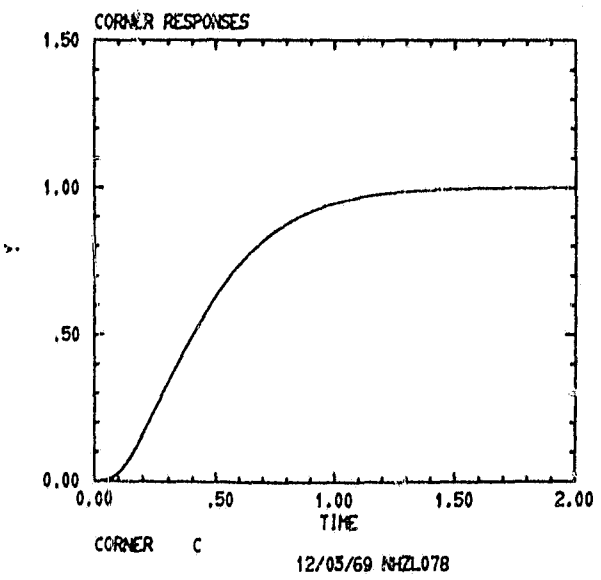
Case	K	α_z	β_z	P_1	GFAR	A t_r	A os	B t_r	B os	C t_r	C os	D t_r	D os	E t_r	E os
1	35.2	2.0	3.6	30	1060	.48	12.8	.78	0.	.8	1.8	.57	18.4	.53	20.9
2	32	2.6	3.8	20	640	.48	5.9	.76	0.	.79	.8	.54	13.8	.50	15.1
3	22	3.0	4.0	25	550	.39	2.6	.87	0.	.86	0.	.53	12.5	.48	14.5

Table 6.1

Comparison of Original, Intermediate, and Final Designs







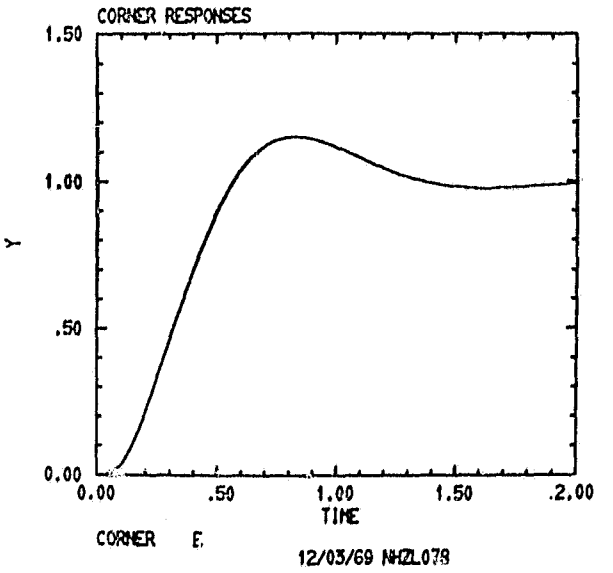
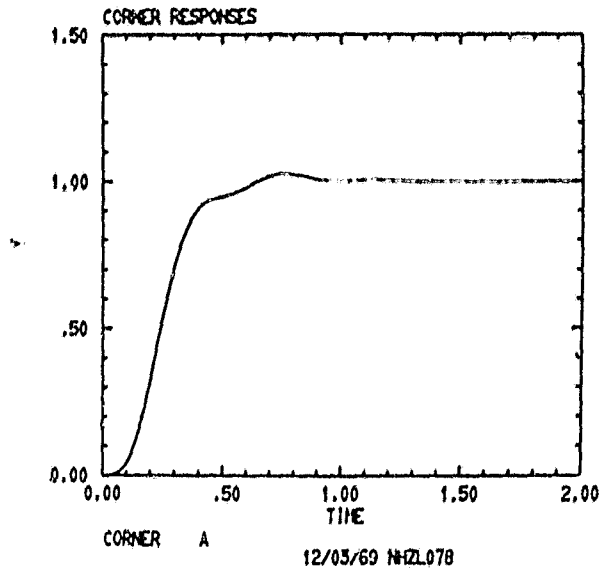
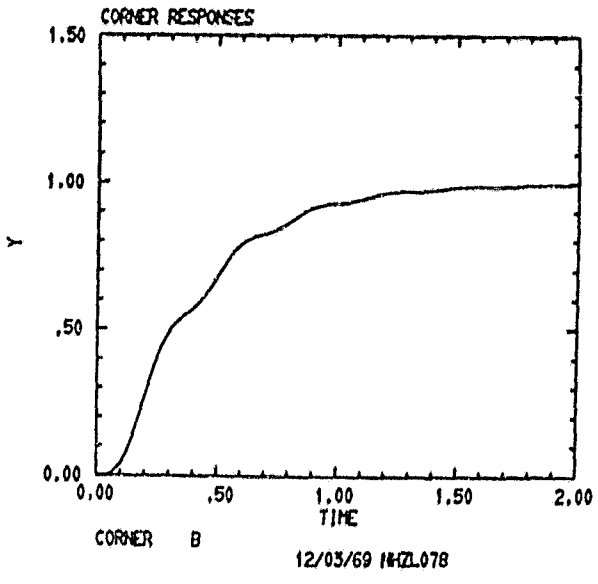
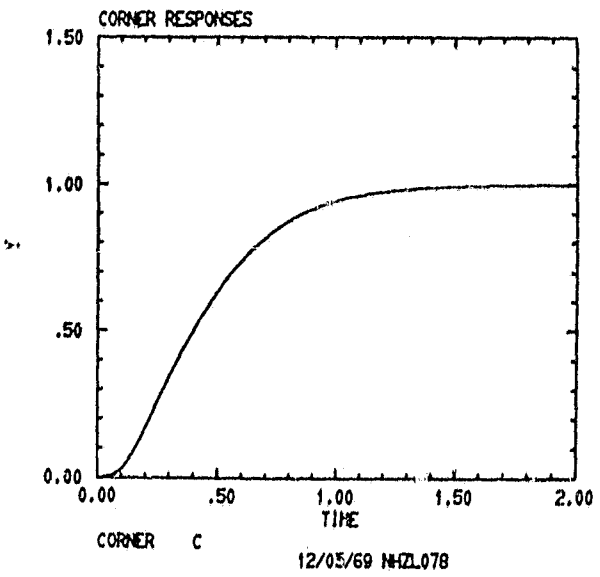


Figure 6.9.

Step Responses for Intermediate Design







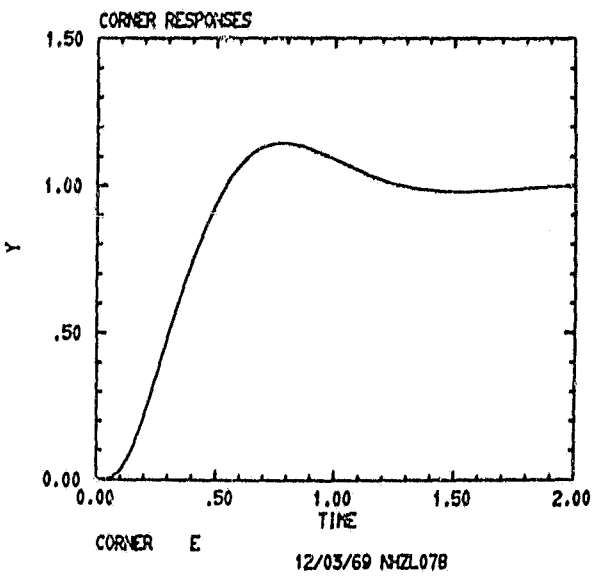


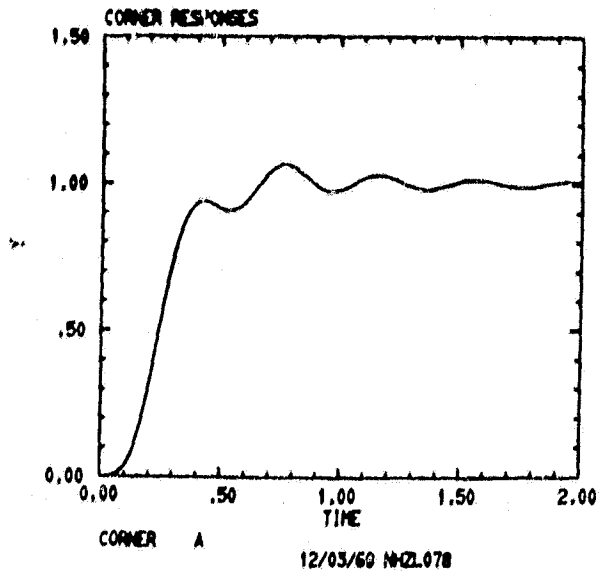
Figure 6.10

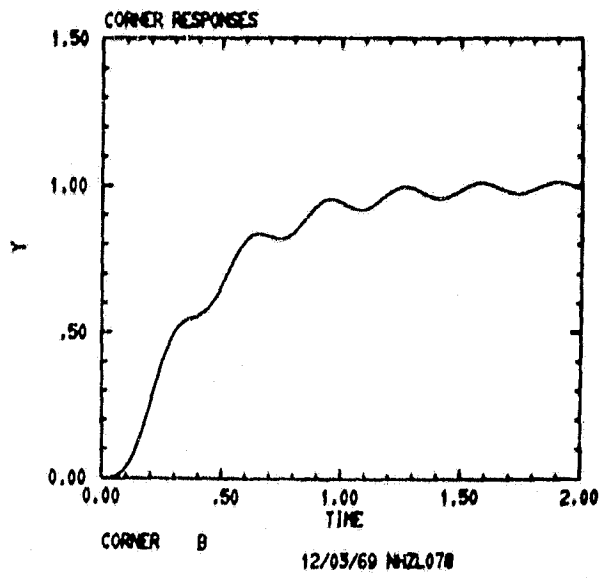
Step Responses for Final Design

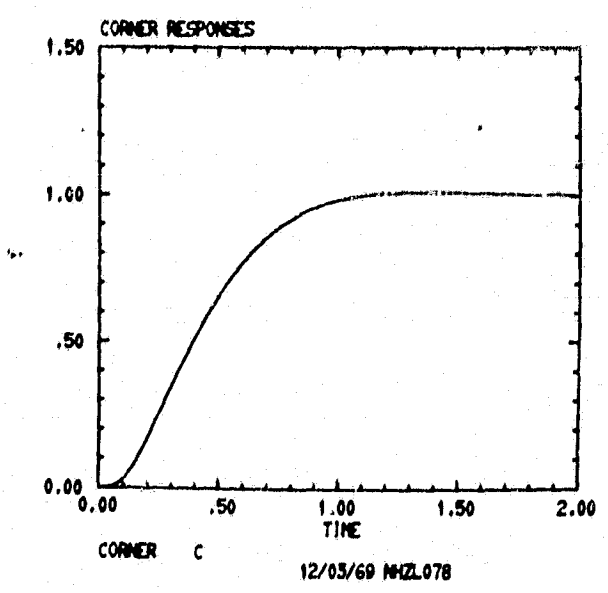
One should note that P_1 is larger at the final value than at the intermediate design, which represents knowledge gained in the design example. Further reduction of P_1 causes an effect termed "low overshoot." The dominant response is a very long-tailed exponential. However, the far-off poles have very little damping and produce a sinusoidal variation superimposed on the exponential. This can cause the derivative of the response to become negative before reaching final value which was deemed undesirable and disallowed. This low overshoot problem is not the same as noted by Smay [19] which is caused by a far-off real pole corrupting a dominant second order response. This again emphasizes the desirability of using a fourth or higher order model. Hence, P_1 was left at +25 rather than introduce a response having low overshoot. Figure 6.11 shows the detrimental effect to the final design of moving P_1 from +25 to +20. The oscillations could not be termed acceptable. More important, the designer sees the exact response.

Evaluation of Design Procedure and Results

It is interesting to compare this design with the design of [4] using the criteria of that method. Even on that basis, it reveals possible improvement. The dominant closed loop poles corresponding to plant poles along the boundary ABCD are shown in Figure 6.12. The original design mapping is given in Figure 6.3. The region of variation now fills more of the allowable region, particularly in the region of large ω_n.







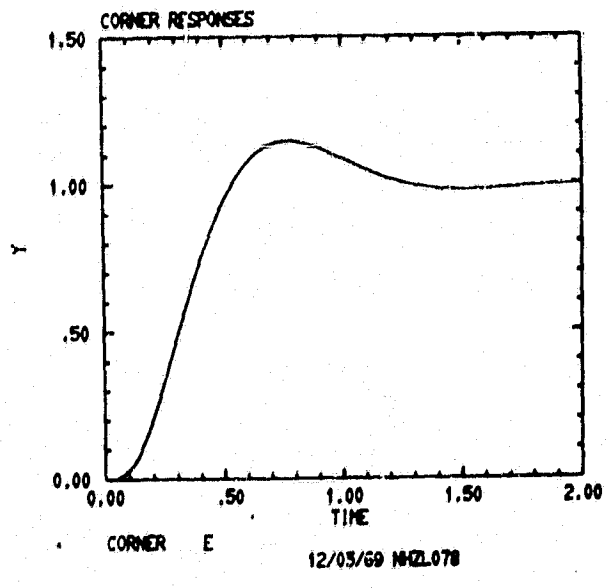
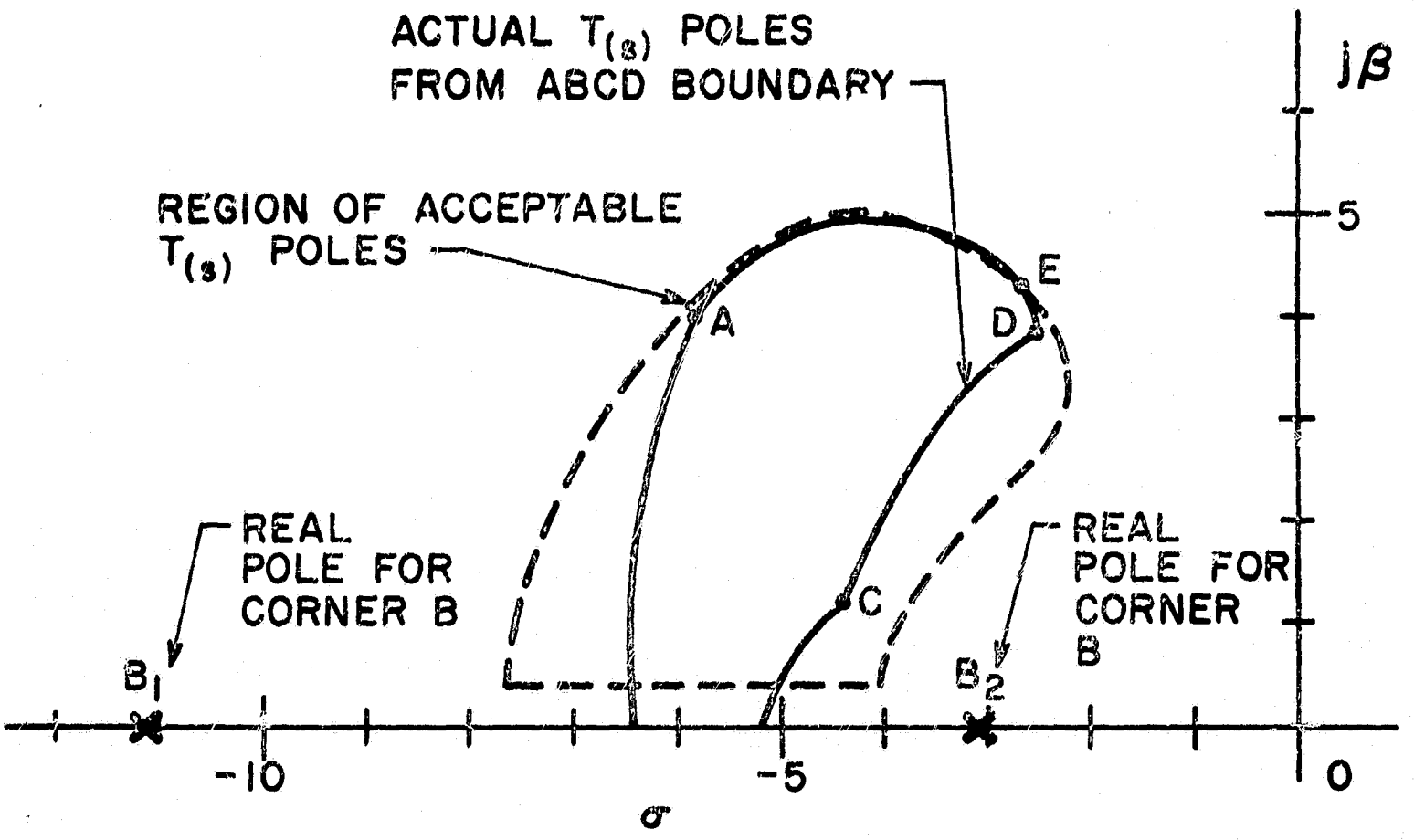


Figure 6.11

Step Responses for Modified Final Design

P_1 changed from 25 to 20

K, α_Z, β_Z as in Final Design



FINAL DESIGN : $K = 22.$, $\alpha_Z = 3.0$, $\beta_Z = 4.0$, $P_1 = 25$

ACTUAL RESPONSE VALUES:

CORNER	% OVERSHOOT	RISE TIME - SEC	SETTLE TIME – SEC
A	2.6	.39	.39
B	0.0	.87	.87
C	0.0	.86	.86
D	12.5	.533	1.025
E	14.5	.48	.97

Figure 6.12 MAPPING OF REGION OF VARIATION TO CLOSED LOOP $T_{(s)}$ PLANE, DOMINANT ROOTS ONLY.

10

The far-off poles do not satisfy the conditions of [4]; however, their exact effect on the response is now known.

In [4], it was concluded that minimization of γ equal to $K\omega_{n_z}^2$ was equivalent to minimization of K. Here $\omega_{n_z}^2$ has increased from 17 to 25, about 50%. Thus it is now known that when using the method of [4], changes in $\omega_{n_z}^2$ may be significant and should be considered.

The root loci for the final design are shown in Figure 6.13. These are not significantly different from Figure 6.2, except that movement of the zero to the left has brought the nominal gain roots for points D and E definitely beyond the minimum ζ point.

The original problem specified that plant gain could also vary. The design was effected using the minimum value which assumes that higher gain will only improve the responses. The root locus intuitively verifies this and actual verification can be made by choosing a higher gain and sweeping around the ABCD boundary looking for unsatisfactory responses.

The most noteworthy result is that a significantly better design has been obtained than that available by previous methods. However, it absolutely requires computer assistance and cannot be adopted to hand methods, which the previous method can do. The use of open-loop space for manipulation of compensation and data presentation is a departure from previous methods. The designer's basic interest in the closed loop transfer function is to obtain the transient response.

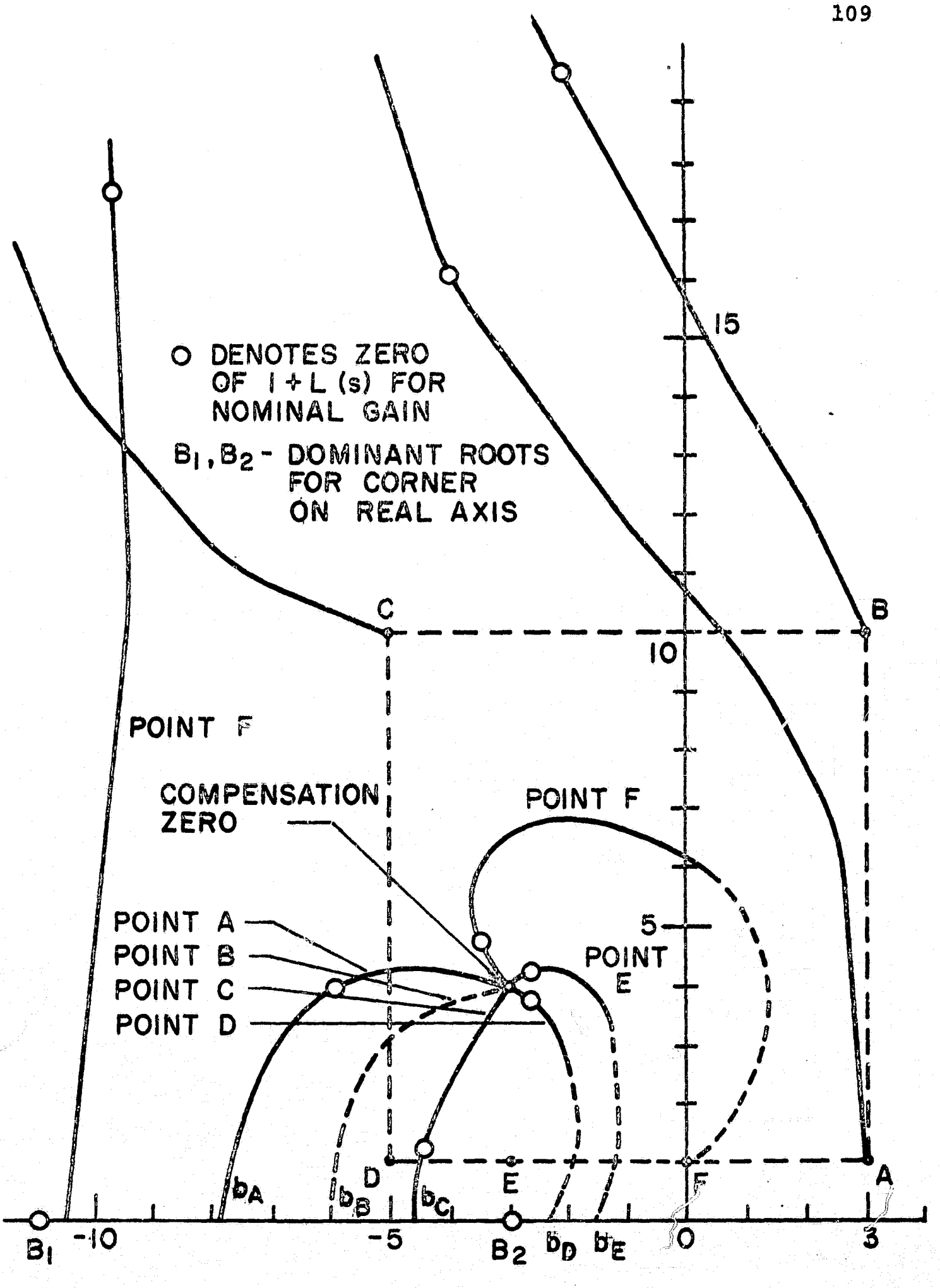


Figure 6.13 ROOT LOCI FOR FINAL DESIGN VALUES

The use of computer simulation for the response values allows the designer to concentrate on the open-loop space which contains variables over which he has direct control and about which he has best information. If constraints are placed on compensation values, such as minimum gain for satisfactory velocity constant, etc., then the open-loop presentation allows a more direct control. Of course, the designer must develop a feel for how compensation changes will affect his design in this space. This has been essential in each design procedure developed to date. The perturbation procedure assists him in this task.

CHAPTER 7

ADDITION OF $T'_{(s)}$ ZEROS TO DESIGN PROCEDURES

7.0 Introduction

Chapter 6 has shown a method of design for systems whose plant poles vary. The design example covered roughly the same class system which Horowitz[4] and Olson[3] had previously solved. The method was able to improve upon these results somewhat. In this chapter extension will be made to systems having zeros in the $T_{(s)}$ transmission. No previous methods could accurately consider these effects since they were restricted to the dominant second order response specification. The addition of the zero will be considered in three phases. First, it will be a pre-filter; second, a fixed compensation zero in the forward path; and third, a drifting real plant zero.

7.1 Real Pre-Filter Zero

It has been noted that the constant performance contour method described here has removed the second-order dominance restriction on $T_{(s)}$ for a basic two-degree-of-freedom configuration. Another class of systems which was previously unmanageable involves zeros in the $T_{(s)}$ transfer function, either from the plant or from the compensation. These zeros have not been allowed because of increased complexity in response specifications. It is known that the introduction of a zero to a second-order pole pair increases the overshoot and reduces the rise-time of the step response. An increase of open-loop gain generally has the same effect. Thus a zero could be introduced

in a pre-filter without affecting GFAR. This could possibly allow reduction of gain, and consequently of GFAR. The $T_{(s)}$ zero has the additional desirable effect of reducing the settling time. The method described here can accomodate the zero with no added difficulty, using the computer simulation to return values of rise-time, etc. The only difference is that there are now five compensation parameters which can be varied.

For an illustration, consider the intermediate design of Figure 6.8 as a reference. A real zero is moved in from $-\infty$ to -25 to -5 with the results shown in Figure 7.1. The rise time and overshoot boundaries both move up. The compensation zero can be changed to move the constant performance contours so that the ABCD square is enclosed.

To illustrate this, a $T_{(s)}$ pre-filter zero at -10 is added to the final design, Figure 6.8. The new constant performance contours are shown in Figure 7.2 and the step responses in Figure 7.3. As expected, overshoot has increased and rise-time and settling time have decreased. The basic specifications are no longer satisfied. "Low overshoot" is much worse due to the differentiation of the zero. Note that maximum rise time has changed little while the minimum has changed significantly.

The parameter perturbation design method was applied under the constraint that GFAR equal to KP_1 remain constant. Two possible final designs are shown in Figure 7.2. Number 1 has worse "low overshoot" while number 2 has more overshoot at E. The designer can choose between these or increase K and try again. The transient responses for number 1 are shown in

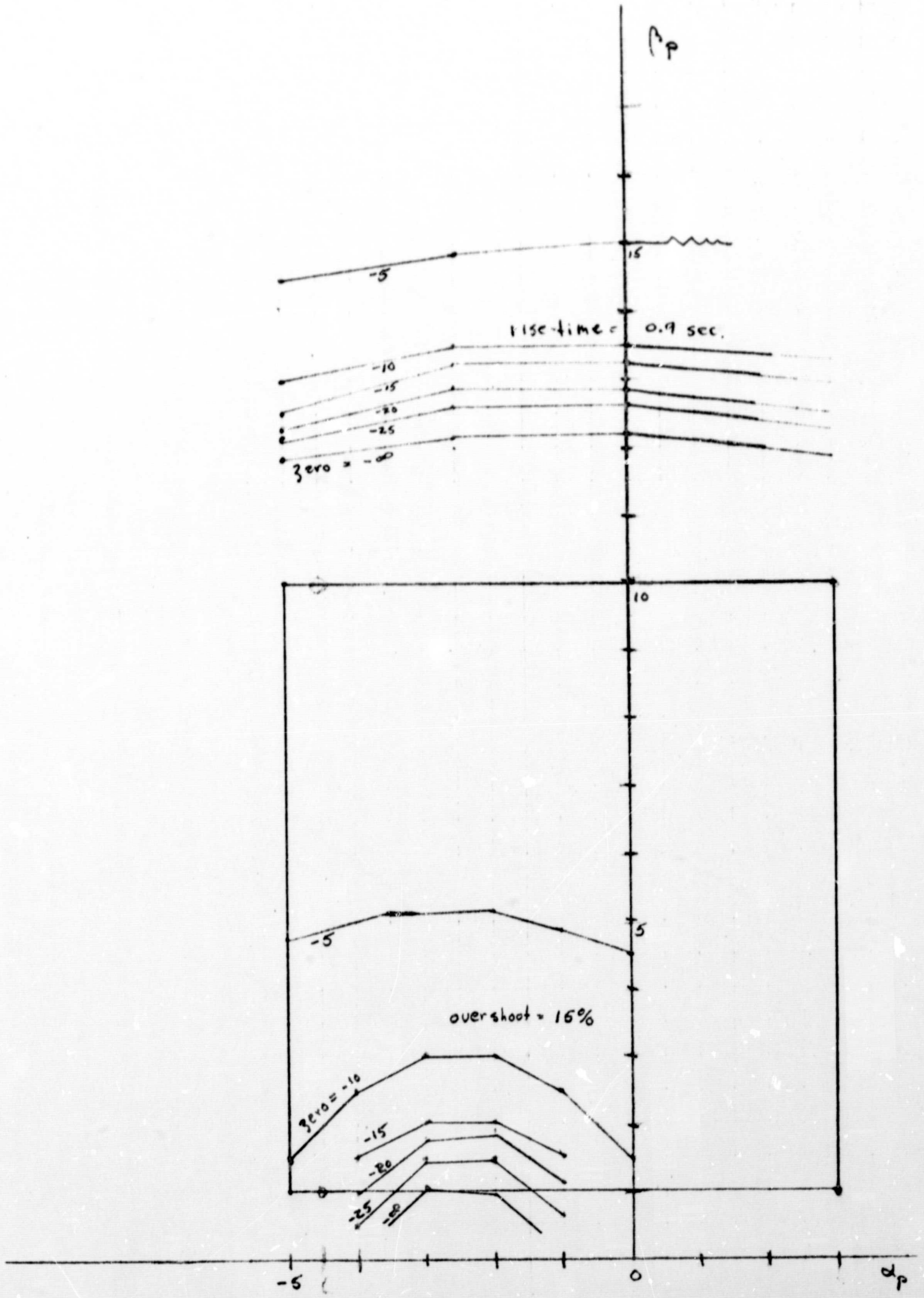
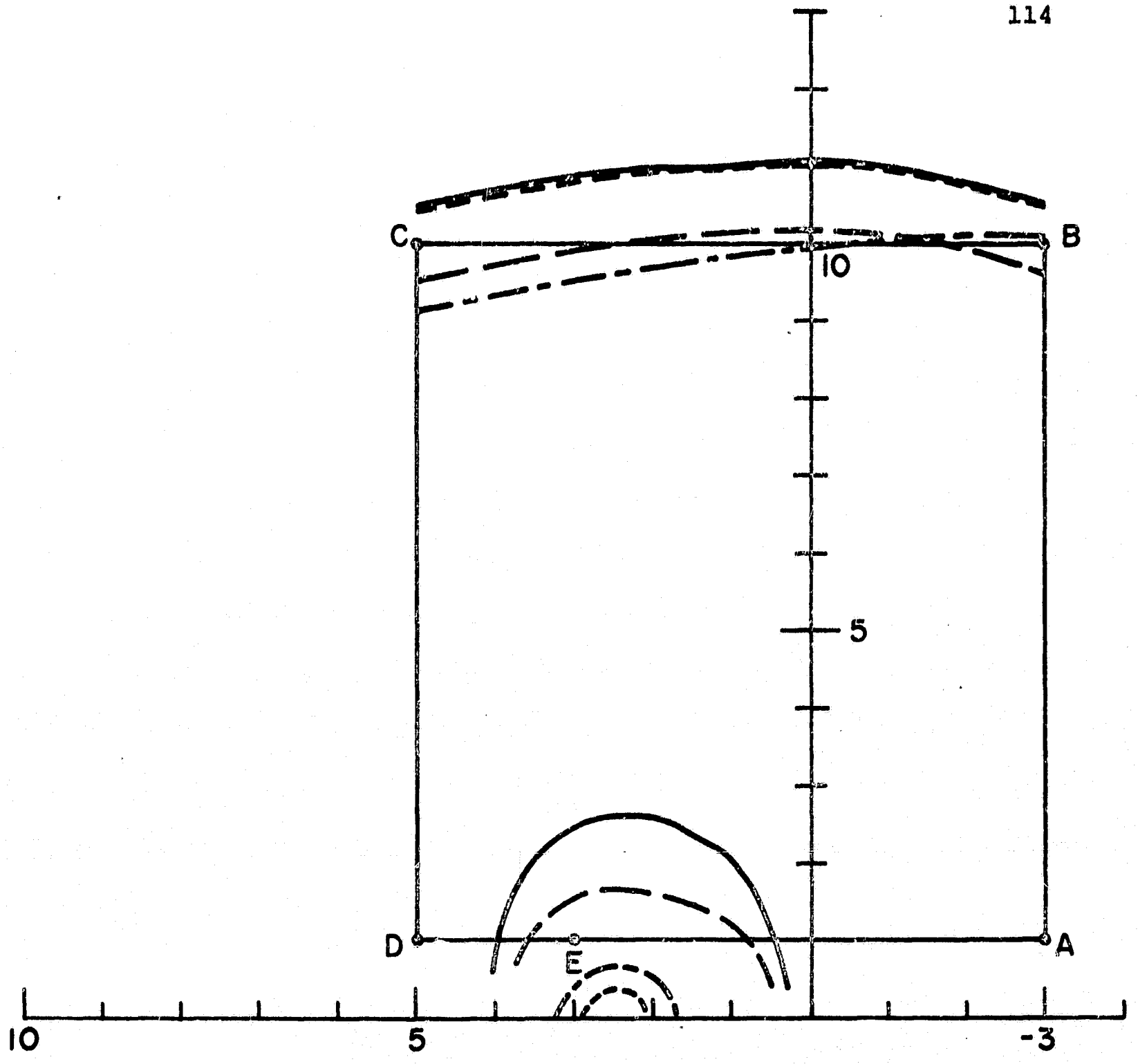


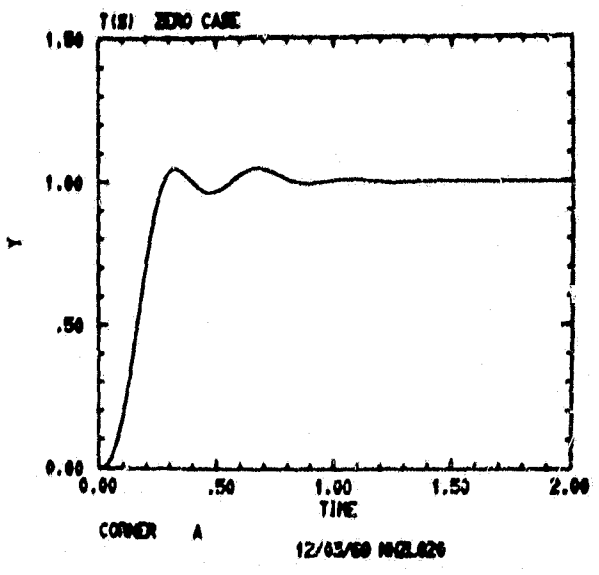
Figure 7.1 Effect of Pre-filter Zero, Intermediate Design

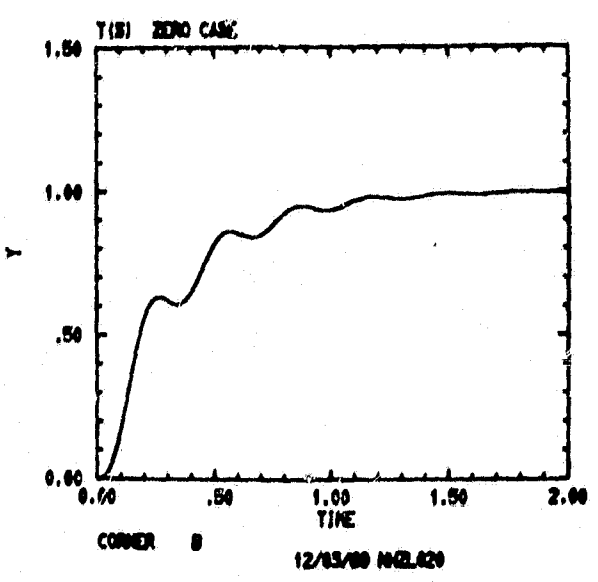


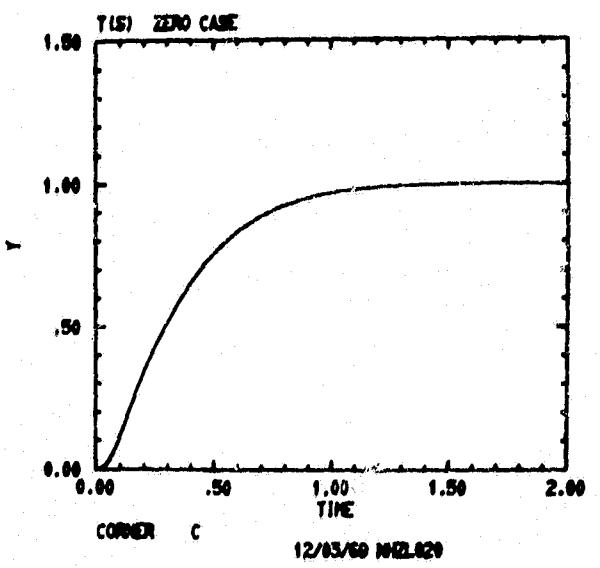
tr AT CORNER A

.39 SEC	----	FINAL DESIGN - NO $T_{(s)}$ ZERO	$\alpha_Z = 3.0,\ \beta_Z = 4.0$
.25 SEC	———	ADDITION OF $T_{(s)}$ ZERO AT -10 TO ABOVE	$\alpha_Z = 3.0,\ \beta_Z = 4.0$
.316 SEC	— - —	FINAL DESIGN - NUMBER 1	$\alpha_Z = 2.6,\ \beta_Z = 3.4$
.325 SEC	— —	FINAL DESIGN - NUMBER 2	$\alpha_Z = 2.6,\ \beta_Z = 3.5$

Figure 7.2. CURVES OF CONSTANT PERFORMANCE SHOWING EFFECTS OF $T_{(s)}$ PRE-FILTER ZERO







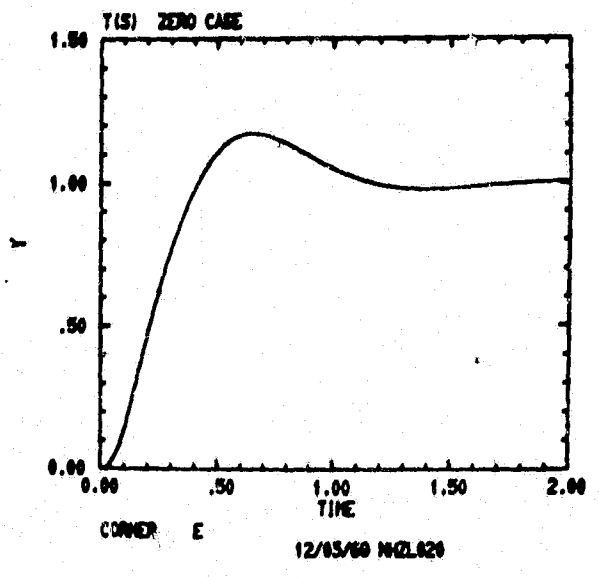


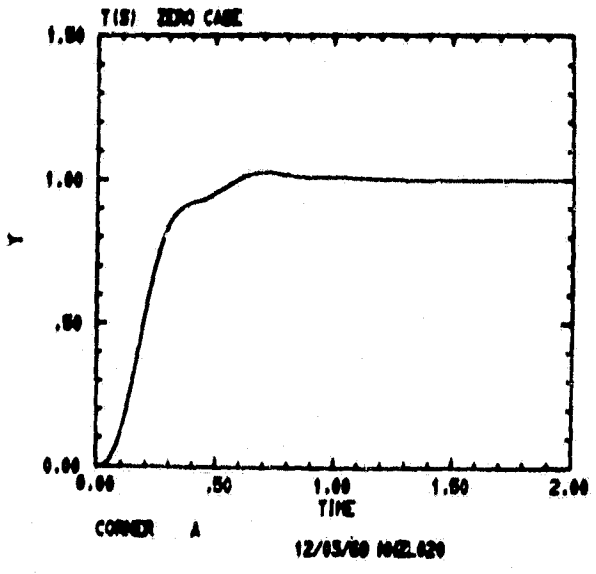
Figure 7.3

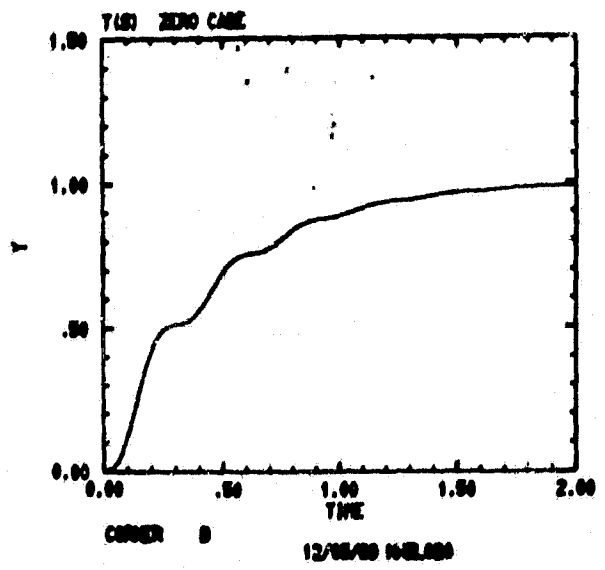
Step Responses for Addition of T(s) zero to Final Design Figure 6.8

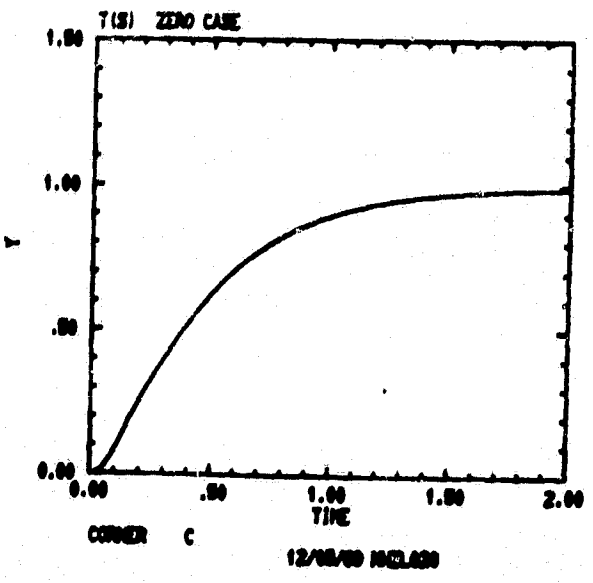
$K = 22.$
$\alpha_Z = 3.0$
(zero at -10) $\beta_Z = 4.0$
$P_1 = 25$

Figure 7.4.

Since the transient responses are no longer simple second order, attempting to push a design this far without knowing the exact response could lead to serious problems. In this example, the addition of the real pre-filter zero has not allowed a reduction of GFAR as desired. The specifications are such that the settling time improvement is not appreciable. Considering the increased oscillation at corner B, the addition of the zero has been of marginal usefulness.







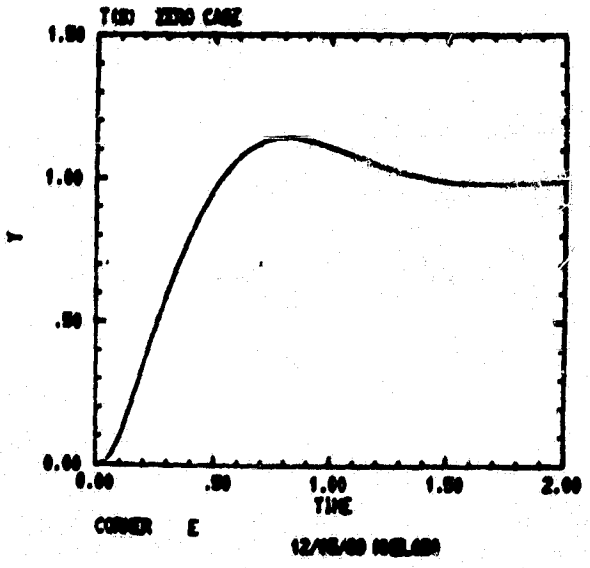


Figure 7.4

Step Responses for Final Design No. 1
for T(s) Pre-filter zero

$K = 22.,\ \alpha_z = 2.2,\ \beta_z = 3.4,\ P_1 = 25,$ zero at $=10.$

7.2 REAL ZERO IN $L_{(s)}$ AND $T_{(s)}$

This section extends the previous by placing the real zero inside the loop in the forward path, thereby included in both $L_{(s)}$ and $T_{(s)}$. For a first try, the zero, in Bode form, is placed in the forward path, using the final case of Section 6.4 as a reference. The damping is increased considerably so that overshoot is decreased and rise time is increased. The compensation zero is changed to $\alpha_z = 3.1$, $\beta_z = 4,3$ with the result that the specifications are again satisfied.

However, use of the zero in Bode form with no associated pole has changed the system from one with a pole-zero difference of two to only one. Thus GFAR is now 5500 (ten times worse). For consistency, another real pole should be included. For a pole at -100, the change in transient response is small and GFAR would be ten times worse at all high frequencies. The perturbation method was used to bring the real poles to a lower frequency to reduce GFAR. It was possible to bring it down to about 1500 or three times greater than without the zero inside the loop. Thus the noise degradation outweighs the performance improvement. Should the specifications be such that improvement of settling time be significant, the balance could be in favor of the zero.

For future reference, the real pole in the range between the origin and the real zero has considerable effect on the response for corners A, B, and C. For points D and E, it is within 10% of the zero resulting in a small residue.

7.3 DESIGN FOR PLANT WITH DRIFTING REAL ZERO

This section will consider the design problem in which the plant has a drifting real zero in addition to the variable complex poles and gain. The range of the zero overlaps the range of real parts of the desired dominant second order response. The zero position is not correlated with any of the other variations.

Olson[3] considers this problem as an extension of his previous work. The philosophy is to place a real pole in the interval of zero variation and use sufficient gain to drive it to the zero, wherever it may be. This accomplishes two purposes: 1) the pole is cancelled by the zero leaving a dominant second order system, and 2) the zero is cancelled so that it does not appear in $T_{(s)}$. The design procedure is approached via the X-Y plane as before. P_1 is not used since this would make the system equation fifth-order. When values for compensation are chosen, the variation region of complex poles is mapped into the closed loop s-plane. Two mappings are made, one each with the extreme values of the zero range. The region of variation should be within the acceptable region in both cases. The variation of the real pole is

obtained but not used. More arbitrary choices for parameters must be made in the process now. But Olson's method does definitely lead to a design.

As systems grow larger, it is more difficult to force them to have a simple dominant second order response. Olson's work provides an interesting example. His design example begins by simply adding the drifting zero to the previous final design without the zero. The result is that corners B and C are considerably outside the acceptable second order region. However, when the system step response was examined, it was found to be quite acceptable based on time response requirements. The residue of the real pole was 57% of the residue of the pole at the origin which is hardly negligible. Following the same procedure on his final design, it was found that the design was much better than necessary. The maximum overshoot was 9% compared to allowable 20%. Rise time varied between .43 and .56 seconds when the allowable range was .35 to 1.0 seconds. Although not yet optimum with respect to minimizing noise transmission, it is nevertheless a design which meets the response requirements.

It is reasonable to assume that to achieve a near-optimal design with respect to reducing the noise transmission, the approximation of the response by only its

second order complex roots must be improved. The computer simulation procedure used here will do that. The question remains whether a workable design procedure can be developed to accommodate the additional variations and additional compensation parameters. This can best be shown with an example.

The intermediate case of Figure 6.8 was selected as a starting point and the drifting zero was added with the same range as in Olson, namely [-3, -1]. The zero is considered in the root locus form. A real pole is added at the mid-point of the interval, -2. The resulting curves of constant rise time and overshoot are shown in Figures 7.5 and 7.6 along with the curves from Figure 6.8 for reference. We see that for the zero at -1, the rise time boundary became worse while the overshoot boundary became better. For the zero at -3, the rise time boundary became better and the overshoot became worse. The design is not acceptable as two boundaries exclude portions of the variation region. One must now proceed to the design phase to improve the results.

The techniques of perturbing parameters, seeking ones which improve overall results is continued here. Since the intermediate design already fit the region of variation closely except for minimum rise-time, the compensation zero was not perturbed. Rather parameters new to the problem were examined. The new real pole at -2

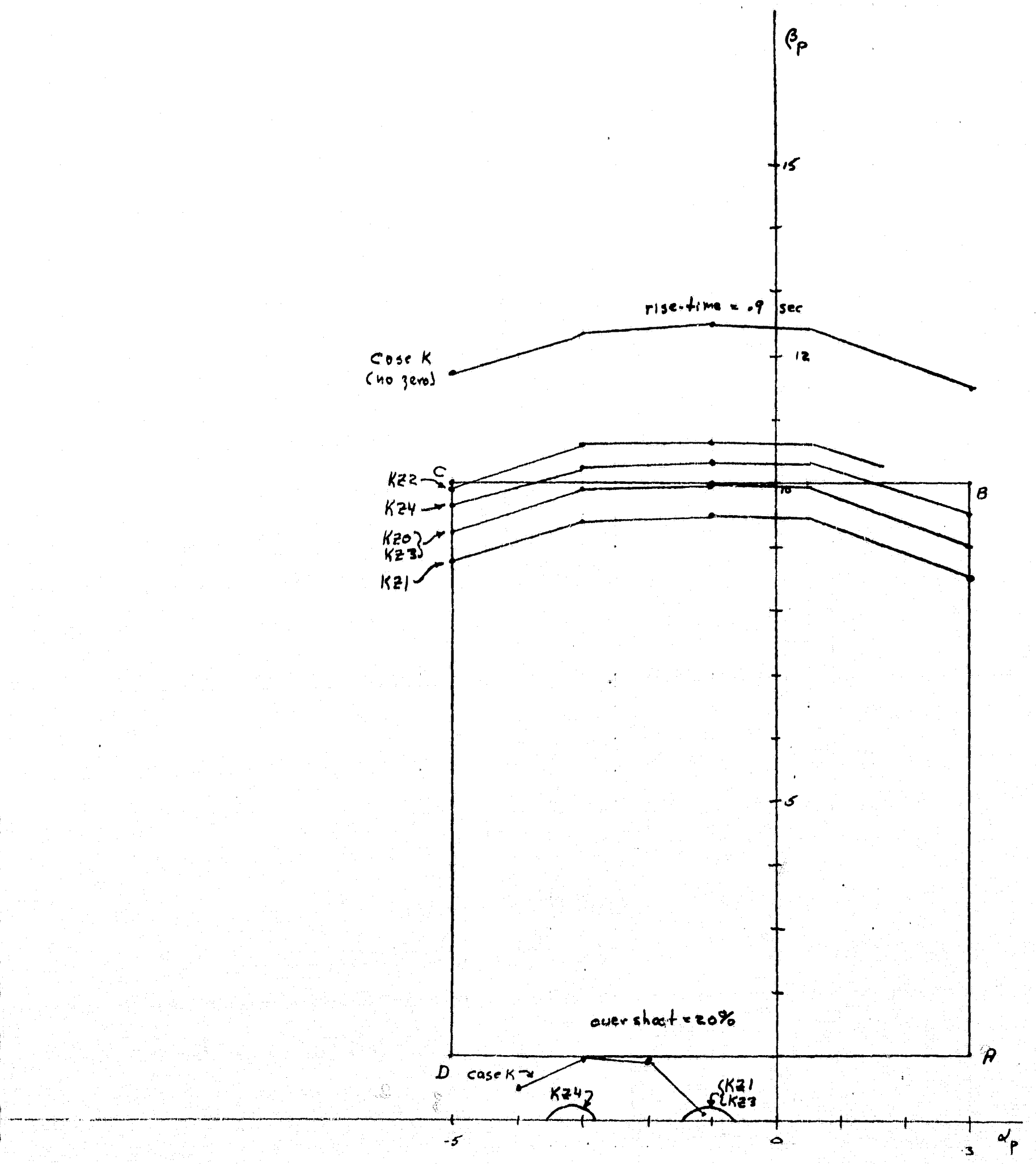


Figure 7.5

Design with Drifting, Real Plant Zero at -1

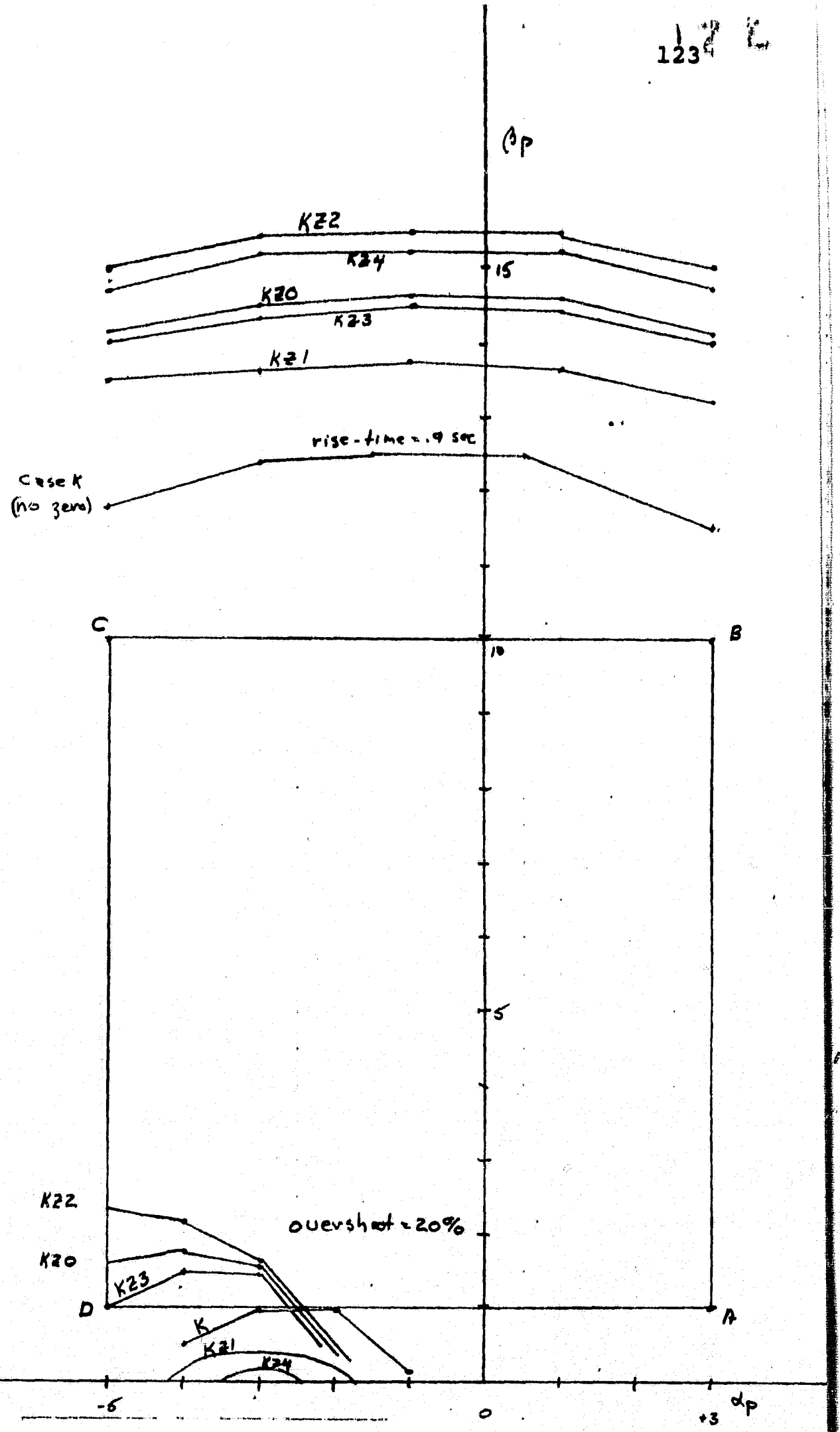


Figure 7.6

Design with Drifting, Real Plant Zero at -3

was moved each way from the midpoint of the interval. The far-off pole P_1 was moved out. Finally the gain was increased to drive all poles closer to the zeros. These are given as Cases KZ1, KZ2, KZ3, and KZ4 in Figures 7.5 and 7.6. The results are summarized in Table 7.1.

We see that moving the real pole from the midpoint had mixed results with no indication that it should go either way. Movement of the real far-off pole had little effect. The increase of gain was the only move which gave all around improvement. In fact, Case KZ4 is very nearly a satisfactory design in itself.

It seems unusual that the mid-point of the range of zero variation is the best place for the compensation pole. To investigate this further, the pole was moved outside the range entirely to -4 and to -.5. The results followed the same trend as before and were too poor to plot. Next the pole was moved entirely away from the zero to -15. This also did not improve the results. These perturbations are summarized as Cases KZ5, KZ6, KZ7 in Table 7.1.

These results indicate that it is best to partially cancel the effect of the drifting zero. This conclusion is very dependent on the location of the range relative to the drifting complex poles and to the approximate

Case	K_k	P_z	P_1	α_z	β_z			Overshoot & Z_p at -1	Rise time Z_p at -1	Overshoot Z_p at -3	Rise time Z_p at -3
K	32	-	25	2.6	3.8		original a/o drifting zero.	just touches lower boundary crosses $\alpha = -2$ at 1.1	crosses $\alpha=0$ at 12.4	same as left	same as left.
KZO	32	2.0	25	2.6	3.8		Central Point	Crosses $\alpha=-2$ @ .2	10.0 @ 0	1.74 e-4	14.7 @ -1
KZ1	32	2.3	25	2.6	3.8			better - on real axis	worse 9.5 @ 0	better on real axis	worse 14. @ -1
KZ2	32	1.7	25	2.6	3.8			Same .2 @ -2	better 10.6 @ 0	worse 2.2 @ -4	better 15.6 @ -1
KZ3	32	2.0	30	2.6	3.8			Worse .29 @ -2.	Same 10.02 @ -1	better 1.6 e-4	same 14.7 @ -2
KZ4	35	2.0	25	2.6	3.8			better - on real axis	better - 10.4 @ -1.	better .01 @ -4	better 15.4 @ -2
KZ5	32	4.	30	2.6	3.8			better - on real axis	worse 7.6 @ -1	better 1.28 @ -2	worse 10.8 @ -1
KZ6	32	.5	25	2.6	3.8			worse 2.09 @ -4.	better 14.1 @ -1	worse 6.6 @ -5	better 20.6 @ -1
KZ7	32	15	100	2.6	3.8			same 1 @ -2	worse 5 @ -1	better 1. @ -2	worse 5.9 @ 0
KZ8	36	2.0	25	2.6	3.8	$P_7 = .0167$		on real axis	better 10.8 @ -1	worse 2.4 @ -4	better 16.5 @ -1
KZ9	35	2.0	25	2.6	3.6	$P_7 = .0167$		on real axis	better 10.6 @ -1	better on real axis	better 16.0 @ -1

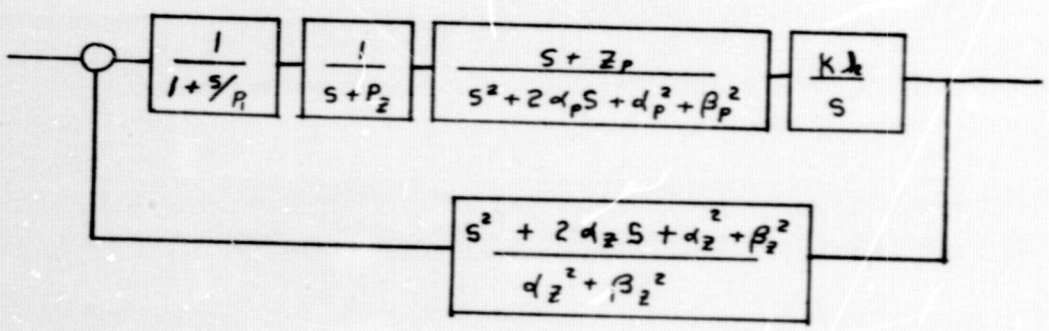


TABLE 7.1

Results of Design Perturbations, Drifting Plant Zero

dominant second order acceptable region. If the range is more to the left of the range used here, the effects will be different.

To answer one further question regarding use of a zero in $T_{(s)}$, a pre-filter zero at -15. was added to Case KZ4 and denoted Case KZ8. The overshoot curve for the zero at -3 is now unacceptable. Recalling from previous work that movement of the complex compensation zero could help, it was moved down to -2.6 ± j3.6. This yields Case KZ9 which is another acceptable design.

Comparing Cases KZ4 and KZ9, we see that the latter is slightly better on all four boundary curves. This indicates that the gain could be reduced on Case KZ9 making it a better design. The amount of reduction would be slight.

Examination of Case KZ4 with the QD017 program indicates as before that the residue of the real pole in the zero range is quite significant. This again demonstrates the need for going beyond the dominant second order response concept for specifying the acceptable response.

Root locus plots for plant poles at each of the four corners for Case KZ4 are shown in Figures 7.7 and 7.8. The plant zero is at -1 and -3 respectively. The roots which combine to make the supposedly dominant complex pair are quite different on the two plots. This

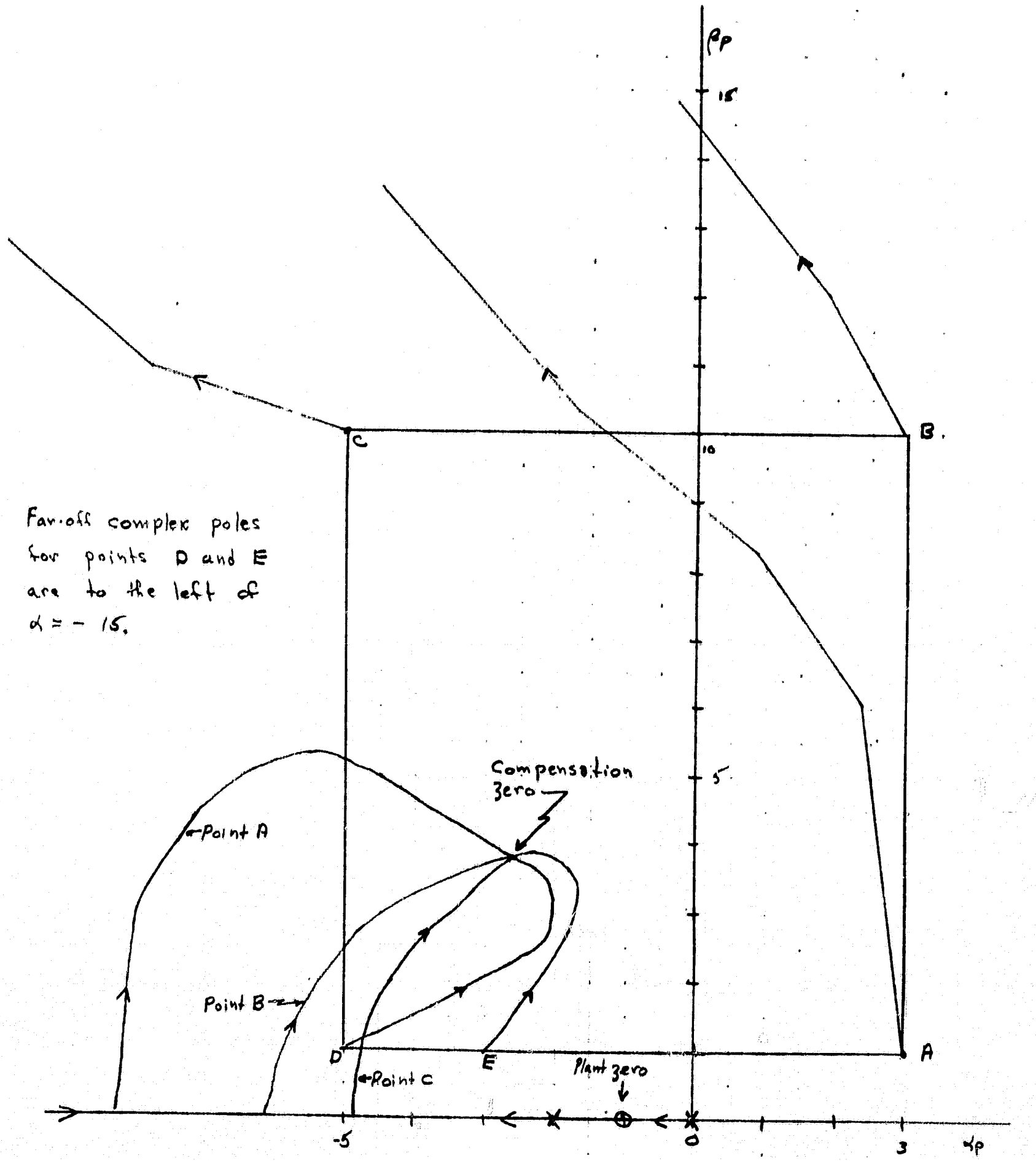
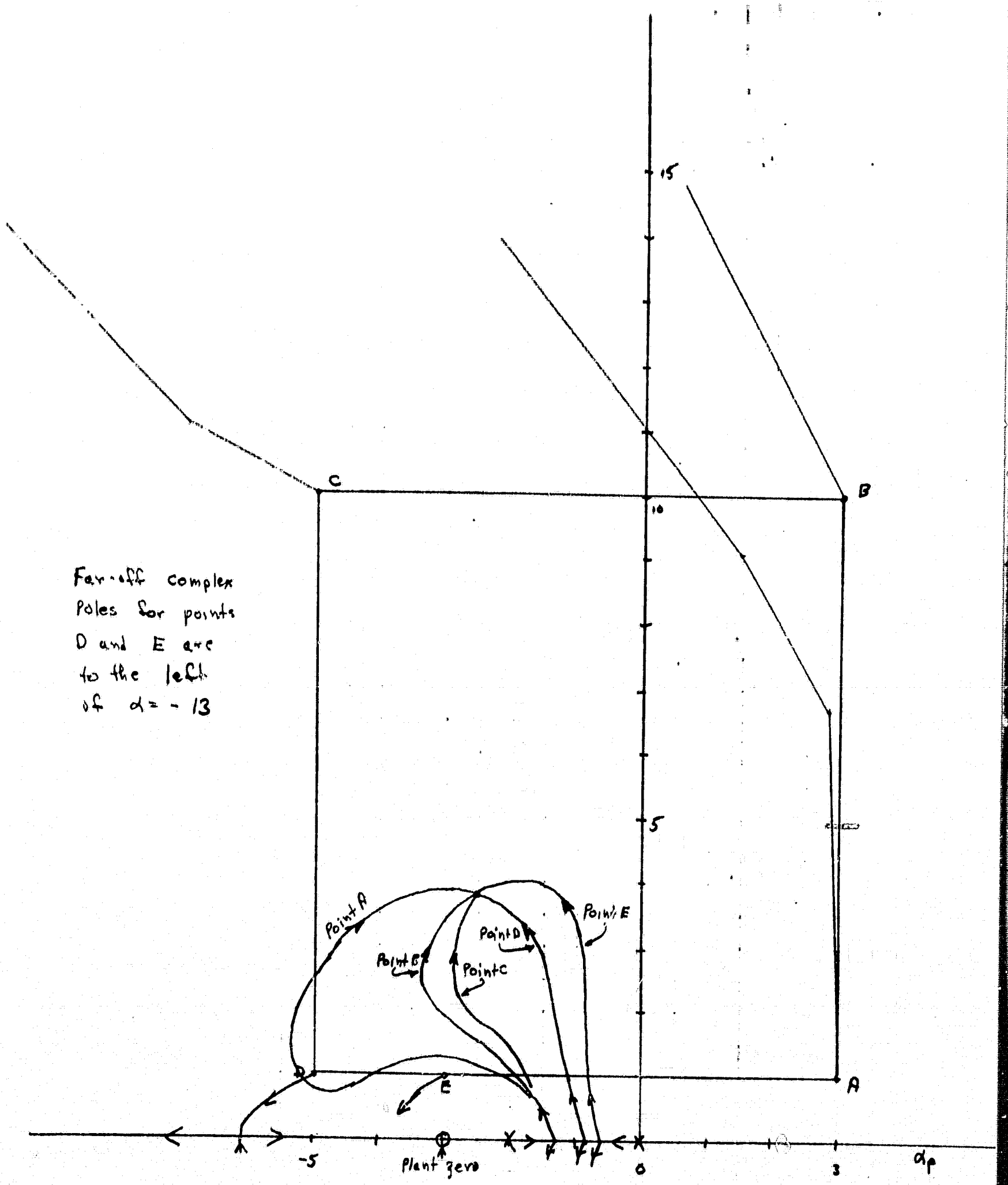


Figure 7.7
Root Loci for KZ-4 Design
Zero at -1



Far-off complex poles for points D and E are to the left of $\alpha = -13$

Figure 7.8

Root Loci for KZ-4 Design
Zero at -3

further illustrates the desirability of combining insight from this method with root locus and other methods. The complexity of the problem is such that one method will not suffice.

CHAPTER 8

CONCLUSIONS

A. The computer solution of a mathematical model to provide time and frequency response data allows use of higher order system models. The exact response data for a given configuration allows use of additional degrees of freedom with a confidence that approximate mappings cannot provide. This provides a closer fit of actual response to desired response while minimizing noise transmission.

B. Operations in the open-loop parameter space are feasible. This allows direct access to variable plant poles and provides the compensation parameters directly upon design completion.

C. The work has shown that design to specific time response requirements, usually inequality relations, is feasible. It also shows that correlation with s-plane methods, particularly root locus, is desirable.

D. The concept of perturbation or gradient search for logically manipulating more than two parameters toward a specified goal works satisfactorily, both in an automatic computer mode and in a human or manual mode.

E. The use of the computer makes possible extensions of current methods that would be otherwise difficult.

The human must still understand the fundamental relationships. The computer can only relieve him of tedious calculations in the choices of parameters and the display of results.

BIBLIOGRAPHY

1. Eveleigh, V. W., Adaptive Control and Optimization Techniques, McGraw-Hill, 1967, pp 117-132.

2. Bekey, G. A., "Optimization of Multi-parameter Systems by Hybrid Computer Techniques," Simulation, Fall 1963 and March 1964.

3. Olson, D. E., Design Procedures for Dominant Type Systems with Large Parameter Variations, University of Colorado M. S. Thesis, 1968.

4. Horowitz, I. M., "Optimum Linear Adaptive Design of Dominant Type Systems with Large Parameter Variations," University of Colorado, 1969. (To be published)

5. Kreindler, E., Synthesis of Flight Control Systems Subject to Vehicle Parameter Variations, AFFDL-TR-66-209, April, 1967.

6. Coffey, T. C., The Application of Modern Computing Technology to Control System Analysis & Design Problems, Aerospace Corp., Los Angeles, Calif., June 1967, SAMSO-TR-67-9.

7. Wertz, H. J. and Wiederholt, L. F., The Eclectic Simulator Program, Aerospace Corp., Los Angeles, Calif., June 1968, TOR-0200(4307-02)-1.

8. IBM Corporation, Numerical Techniques for Real-Time Digital Flight Simulation, E20-0029-1.

9. IBM Corporation, Application of Numerical Techniques to Flight Simulation, E20-8186.

10. IBM Corporation, System/360 Scientific Subroutine Package, Version II, H20-0166-4.

11. Kavanaugh, W. P., Stewart, E. C., and Brocker, D., "Optimal Control of Satellite Attitude Acquisition by a Random Search Algorithm on a Hybrid Computer," Proceedings SJCC, 1968, Thompson Book Co., Washington.

12. Wilde, D. J., Optimum Seeking Methods, Prentice-Hall, 1964.

13. Wilde, D. J., and Beightler, C. S., Foundations of Optimization, Prentice-Hall, 1967.

14. Levine, L., Methods for Solving Engineering Problems Using Analog Computer, McGraw-Hill, 1964.

15. Bekey, G. A., and Karplus, W. J., Hybrid Computation, Wiley, 1968.

16. Nesbit, R. A., and Engel, R. D., "An Example Program for the Determination of Chemical Rate Coefficients from Experimental Data," Simulation, March, 1967.

17. Rohrer, R. A., "Fully Automated Network Design by Digital Computers; Preliminary Considerations," Proc. IEEE, November, 1967.

18. Hannauer, G., Basics of Parallel Hybrid Computers, Electronics Associates, Inc., West Long Branch, N. J., 1968.

19. Smay, J. W., Computer-Aided Design of Second- and Third-Order Systems with Parameter Variations and Time-Response Constraints, University of Colorado M. S. Thesis, 1968.

20. Horowitz, I. M., Synthesis of Feedback Systems, Academic Press, 1963.

21. Temes, G. C. and Calahan, D. A., "Computer-Aided Network Optimization, The-State-of-the-Art," Proc. IEEE, November, 1967, pp 1832-1863.

22. Barber, H., Third-Order Feedback System Synthesis for Prescribed Time Domain Tolerances to Parameter Variation, University of Colorado, M.S. Thesis, 1968.

23. LaBounty, R. H. and Houpis, C. H., "Root Locus Analysis of High Gain Linear System with Variable Coefficients: Application of Horowitz's Method," IEEE Trans. on Automatic Control, Vol. AC-11, No. 2, April, 1966, pp 255-263.

24. Fletcher, R. and Powell, M.J.D., "A Rapidly Convergent Descent Method for Minimization," Computer Journal, Vol. 6, July, 1963, pp 163-168.

25. Ralston, A. and Wilf, M. S., Mathematical Methods for Digital Computer, Wiley, 1960, pp 95-104.

26. Control Data Corp., Control Data MIMIC, A Digital Simulation Language, St. Paul, April, 1968.

27. Giese, C., "Determination of Best Kinetic Coefficients of a Dynamic Chemical Process by On-line Digital Simulation," Simulation, Vol. 3, March, 1967, pp 141-145.

28. Parisot, P., Nesbit, R., and Hara, H., "A Hybrid Program for Solving the Monsanto BenchMark Problem," AICHE, Columbus, Ohio, April, 1966

29. Savas, E. S., Computer Control of Industrial Processes, McGraw-Hill, 1965, pp 142-146.

30. Ibid, Chapter 13.

31. Horowitz, I. M., "Plant Adaptive System vs. Ordinary Feedback System," IRE Trans. on Automatic Control, IEEE, New York, AC-7, January, 1962, pp 48-56.

32. Rolnik, J. A., Feedback Control Systems for Plants with Large Parameter Variations, Doctoral Dissertation, Polytechnic Institute of Brooklyn, June, 1965, (see for references prior to 1965).

33. IBM Corporation, 1800/1130 Control Optimization Program, H2-0208-2, 1967.

34. Truxal, J. G., Automatic Feedback Control System Synthesis, McGraw-Hill, 1955, Chapter 5.