CONTROL SYSTEM DESIGN USING OPTIMIZATION TECHNIQUES

R. T. Stefani
J. H. Dial
T. L. Williams
D. E. Silverstone

August, 1969

Prepared under NASA Contract NGR-03-002-115

ENGINEERING EXPERIMENT STATION
COLLEGE OF ENGINEERING
THE UNIVERSITY OF ARIZONA
TUCSON, ARIZONA

Final Report

CONTROL SYSTEM DESIGN USING OPTIMIZATION TECHNIQUES

by

R. T. Stefani
J. H. Dial
T. L. Williams
D. E. Silverstone

August, 1969

Prepared under Contract NGR-03-002-115

The University of Arizona
Tucson, Arizona

for

Lewis Research Center
National Aeronautics and Space Administration

TABLE OF CONTENTS

TABLE OF CONTENTS--Continued

PREFACE

This report is divided into four sections. The first two sections are concerned with applying optimization techniques to the control of the fuel valve system for an air-breathing jet engine. In the last report, a system design which was referred to as a modified observer system was discussed. The resulting control system consisted of a series compensator which was one order less than the plant and a feedback compensator of the same order as the series compensator. The design which resulted did not include a specification of the poles of the feedback compensator. The specification of the poles of the feedback compensator is the subject of the first two sections. The pole positions must be chosen so that the feedback compensator and the series are both stable. This restriction imposes an inequality constraint on the optimization process. In addition, the poles should be chosen so that the system has a low sensitivity to changes in the parameters of the plant. This requirement is the objective function which we are attempting to minimize.

The first procedure of this report uses the Sequential Unconstrained Minimization Technique. This procedure incorporates the inequality constraints into the objective function and minimizes this function subject to the equality constraints which result from the modified observer design. These equality constraints relate the poles of the feedback compensator to the poles of the series compensator.

These relationships were discussed in the last report. The final result is a design which has a small low frequency sensitivity, realizes the desired closed-loop transfer function, and results in stable compensator transfer functions. Two such designs are presented. One is for an eighth roder transfer function, and one is for a seventh order transfer function.

The second procedure which is addressed to the fuel valve problem incorporates a different optimization procedure which is called the method of Parallel Tangents or PARTAN. This procedure constructs a vector search in a direction which is orthogonal to all the previous searches. This procedure is usually more effective than the first one gradients are difficult to evaluate. It is especially effective when the objective function is quadratic or nearly quadratic. In this procedure, a design is obtained which is different from the first but would still be a useful design for the fuel valve servo.

The last two optimization procedures are applied to the 40-60 inlet control problem. This problem entails minimizing the response of the shock wave in the inlet to pressure disturbances from the compressor side of the inlet. The first procedure utilizes a Pattern Search to optimize the feedback compensator between the exit pressure and the bypass doors. The search procedure starts with an initial choice of parameters for the compensator and makes changes in these parameters until an improvement is obtained. If an improvement is obtained, steps are continued in the same direction until no further improvement is possible and the process is repeated until even very small steps do not result in an improvement. This particular application was to determine the best numerator of the feedback compensator with two zeros.

The last section uses the Variable Metric optimization procedure to determine the best control system for the inlet. The Variable Metric procedure assumes that the objective function is nearly quadratic and determines the optimum in one less step than the number of parameters if the objective function is quadratic. This procedure is especially useful when the gradients of the objective function are readily available. As in the procedure above, the strategy seeks to force the total closed-loop response of the system to match a desired frequency response over a range of frequencies. The objective function is the total squared error over this range of frequencies. A physically realizable control is achieved for a variety of parameter values.

SEQUENTIAL UNCONSTRAINED MINIMIZATION

Introduction

It is the object of this section to show that the fuel valve servo problem may be solved by conventional optimization techniques. Specifically, the fuel valve servo problem is shown to be equivalent to the problem of minimizing a nonlinear objective function y(x) subject to both equality and inequality constraints (i.e., a nonlinear programming problem).

To help solve this problem a technique is considered (the sequential unconstrained minimization technique or SUMT) which concerns minimization in the presence of inequality constraints. In the SUMT procedure a new objective function Y(X,r) is selected such that its minimization will yield a solution $X^*(r)$ (the* indicates the best choice) which satisfies the inequality constraints. Then as r approaches zero, $X^*(0)$ becomes the value of X^* which minimizes the original objective function y(X), while satisfying the various inequality constraints. We then may consider the minimization of Y(X,r) subject to some additional set of equality constraints. The introduction of equality constraints means that not all N of the variables X are independent.

A technique for accomplishing minimization in the presence of equality constraints is the constrained derivative or Jacobian technique. If there are N variables and N_s constraints, one may obtain a set of N equations in N unknowns where the first $(N-N_s)$ equations are constrained derivatives (to be equal to zero when the minimum is achieved) and the remaining N_s equations are the equality constraints. Combining both the SUMT method and the constrained method, we then obtain a set of N equations in N unknowns which, when solved, yields a value $X^*(r)$ which satisfies both the equality constraints and the inequality constraints while minimizing $Y(X,r)$. Then, as r approaches zero, $X^*(r)$ approaches X which minimizes $y(X)$ while satisfying all the equality and inequality constraints. In order to solve the N nonlinear equations in N unknowns, a Newton Raphson method is used and extensive use is made of digital computers.

Alternate procedures for solving the fuel valve problem fall into two categories, namely different ways of solving the optimization problem or alternate expressions (strategies) concerning the basic problem. It is felt that other optimization techniques (PARTAN, Fletcher Powell) are difficult to apply in the presence of the many equality constraints. Also, alternate strategies for attacking the fuel valve problem which consider the equality constraints must somehow treat the inequality constraints. The above procedure is straightforward, but, as will be shown concerning the computer program, coding for high dimension problems can be a tedious job.

In the remainder of this section, the optimization procedures are presented, the algorithm for solving the fuel valve problem is derived, the computer program is discussed, results are presented and conclusions are drawn.

Optimal Procedure

Suppose that we wish to minimize the objective function y(x) subject to a set of N inequality constraints $G(X) \geq 0$ each of the form

$$g_i(x) \geq 0 \tag{1}$$

where

$$i = 1,2,\ldots,N.$$

This problem may be solved by considering a dual problem, namely minimizing the function

$$Y(X,r) = y(x) + r \sum_{i=1}^{n} \frac{1}{g_i(x)} \tag{2}$$

Suppose that, for some choice of r, some $g_i(x)$ are positive and large, and the rest are near zero. The large positive ones will contribute little penalty to the function Y(X,r) while the $g_i(x)$ which are near zero will contribute heavily. Consider, for a fixed r, those x that cause

$$\frac{\partial L(X,r)}{\partial X} = 0 \tag{3}$$

The resulting x is such that L(X,r) is a minimum for each value of r; hence we can call the solution X*(r), that is, the best choice of X for each value of r. If one considers only X in the allowable (feasible) region $G(X) \geq 0$, and, after solving $\partial L/\partial x = 0$ for some fixed r, one then takes the resulting X*(r) as a starting point for a new minimization procedure with a lower value of r: one has a sequential process for minimizing a series of unconstrained objective functions $L(X,r)$ having decreasing values of r for each successive step in the process. Hence, this technique is called the sequential unconstrained minimization technique, abbreviated SUMT.

The end result is that

$$\text{LIMIT}_{r \to 0} X^*(r) = X^*(0) \tag{4}$$

where $X^*(0)$ is the value of X which minimizes $y(X)$ subject to the constraining equations $G(X) \geq 0$.

Let us now drop the subscript r, thus assuming r to be fixed for each sequence of unconstrained minimization process and consider the more general case where the N variables X are not independent; that is, there exists a set of N_s equality constraints $F(x) = 0$ each of the form

$$f_i(x) = 0 \tag{5}$$

where

$$i = 1,2,\ldots,N_s.$$

There are now N_s dependent variables (let us define these as state variables s) and $N-N_s$ independent variables (let us define these as decision variables d). Then the state variables s depend on d. Diagrammatically, one has the situation shown below for the interdependency of the variables. Arrows indicate one variable influencing another.

$$Y \begin{array}{l} \leftarrow s \\ \;\;\uparrow \\ \leftarrow d \end{array} \qquad\qquad F \begin{array}{l} \leftarrow s \\ \;\;\uparrow \\ \leftarrow d \end{array}$$

In order to minimize the objective function $Y(s,d)$ is necessary that the derivative of Y with respect to the independent variables d is zero. We know that the derivative of the equation $F(x) = 0$ with respect to the independent variables must be zero; hence, we have the following N equations in N unknowns using the chain rule of differentiation

$$\frac{dY(d)}{dd} = 0 = \frac{\partial Y(d,s)}{\partial d} + \left[\left(\frac{\partial Y(d,s)}{\partial s}\right)^T \frac{dS(d)}{dd}\right]^T \tag{6}$$

$$\frac{dF(d)}{dd} = 0 = \frac{\partial F(d,s)}{\partial d} + \left(\frac{\partial F(d,s)}{\partial s}\right)\left(\frac{dS(d)}{dd}\right) \tag{7}$$

The difficulty in solving the above lies in obtaining $S(d)$ which for nonlinear constraint equations,is a formidable task. Hence, $[dS(d)]/dd$ can be solved for in Eq. (7) and substituted into Eq. (6) giving the following $N\text{-}N_s$ equations in N unknowns

$$0 = \frac{\partial Y(d,s)}{\partial d} - \left[\left(\frac{\partial Y(d,s)}{\partial s}\right)^T \left(\frac{\partial F(d,s)}{\partial s}\right)^{-1} \left(\frac{\partial F(d,s)}{\partial d}\right)\right]^T \tag{8}$$

This result is called the constrained derivative of L with respect to d. If we include the N_s constraint equations

$$0 = F(d,s) \tag{9}$$

we have, in Eqs. (8) and (9), a set of N equations in N unknowns, all the elements of which are readily available from the objective function and the equality constraints.

The technique selected for solving the above system of equations is to write a Taylor series expansion for the N equations, to take only the linear terms and then to solve for $\begin{pmatrix} d \\ s \end{pmatrix}$. The result is the Newton-Raphson iterative procedure. An initial guess $\begin{pmatrix} do \\ so \end{pmatrix}$ is assumed. Then a better estimate of $\begin{pmatrix} d \\ s \end{pmatrix}$ is obtained, and the process is repeated. Once a value of $\begin{pmatrix} d \\ s \end{pmatrix}$ results which satisfactorily solves the above set of equations for a fixed value of r [see Eq. (2)], we can call the result $\begin{bmatrix} d^*(r) \\ s^*(r) \end{bmatrix}$. Then, repeating the process for even lower values of r, we note that

$$\underset{r \to 0}{\text{LIMIT}} \begin{bmatrix} d^*(r) \\ s^*(r) \end{bmatrix} = \begin{pmatrix} d^* \\ s^* \end{pmatrix} = X^*$$

where X* is the value of X which minimizes the original objective function y(X) subject to the inequality constraints G(X) ≥ 0 and the equality constraints of F(X) = 0. The Newton-Raphson approach results in the following equation:

$$X^*(r) = \begin{bmatrix} d^*(r) \\ s^*(r) \end{bmatrix} = \begin{pmatrix} do \\ so \end{pmatrix} - \left[\begin{array}{c|c} \multicolumn{2}{c}{\dfrac{\partial}{\partial(d,s)} \dfrac{dy}{dd}} \\ \hline \dfrac{\partial F}{\partial d} & \dfrac{\partial F}{\partial s} \end{array} \right]^{-1}_{\begin{pmatrix} do \\ so \end{pmatrix}} \begin{bmatrix} \dfrac{dy}{dd} \\ F \end{bmatrix}_{\begin{pmatrix} do \\ so \end{pmatrix}} \tag{10}$$

Note that the Newton-Raphson algorithm [Eq. (10)] requires the evaluation of the partial derivatives of $\frac{dY}{dd}$. From Eq. (7) it is evident that the partial derivatives of the second term on the right side of the equation are rather involved; hence, perturbation techniques are used to compute them. All other partials and equations are readily available, although somewhat teduous to derive.

It is now in order to redefine the above algorithm (stated in optimization terms) and use specific equations for the fuel valve servo study. The following change of variables adequately describes the problem in a form suitable for the fuel valve servo. The decision variables and state variables become

$$\begin{aligned} d &= PR \\ s &= CR \end{aligned} \tag{11}$$

The dual objective function, including inequality constraints, for a fixed r is defined

$$Y(PR,CR) = \tfrac{1}{2}L^2(PR,CR) \tag{12}$$

Finally, the equality constraints are defined

$$0 = F(PR,CR) = C(CR) - [T_4]P(PR) - [T_6] \tag{13}$$

In Eq. (13), T_4 and T_6 are matrices of constants, while C is a vector quantity, each element of which depends only on the vector CR, and P is a vector quantity, each element of which depends only on the vector PR. The above choice of Eqs. (11), (12), and (13) is made clear in a subsequent paragraph of this section in which the fuel valve servo problem is discussed. For now, it is sufficient to take the above equations and substitute them directly into Eq. (10). The result of this is

$$X^*(r) = \begin{bmatrix} PR^*(r) \\ CR^*(r) \end{bmatrix} = \begin{bmatrix} PR_0 \\ CR_0 \end{bmatrix} -$$

$$\left\{ \left[\begin{array}{c|c} \multicolumn{2}{c}{L \dfrac{\partial}{\partial PR, CR} \dfrac{dL}{dPR}} \\ \hline -T_4 \dfrac{dP}{dPR} & \dfrac{dC}{dCR} \end{array} \right] + \left[\begin{array}{c|c} \left(\dfrac{dL}{dPR}\right)\left(\dfrac{\partial L}{\partial PR}\right)^T & \left(\dfrac{dL}{dPR}\right)\left(\dfrac{\partial L}{\partial CR}\right)^T \\ \hline 0 & 0 \end{array} \right] \right\}^{-1}_{\begin{pmatrix} PR_0 \\ CR_0 \end{pmatrix}} \left[\begin{array}{c} L \dfrac{dL}{dPR} \\ \hline C(CR) - T_4 \ P(PR) \\ -T_6 \end{array} \right]_{\begin{pmatrix} PR_0 \\ CR_0 \end{pmatrix}} \tag{14}$$

In Eq. (14)

$$\frac{dL(PR)}{dPR} = \frac{\partial L(PR,CR)}{\partial PR} - \Delta$$

$$\Delta = \left[\left(\frac{\partial L}{\partial CR} \right)^T \left(\frac{dC}{dCR} \right)^{-1} \left(-T_4 \frac{dP}{dPR} \right) \right]^T \tag{15}$$

Note from the above that it is necessary to obtain the partial derivatives of Δ; hence, perturbation techniques are suggested whereas all other partials are readily available although tedious to derive.

A computer program was written to facilitate the above Newton-Raphson iterative procedure.

Computer Program for the Design Procedure

The computer program has two distinct parts. The first part, a subroutine (RDR), sets up the matrices shown in Eqs. (14) and (15). These equations may be compactly written as

$$X^*(r) = \begin{bmatrix} PR^*(r) \\ CR^*(r) \end{bmatrix} = X_o - \left(DR\big|_{Xo}\right)^{-1} R\big|_{Xo} \tag{16}$$

The first part of the program, then, obtains the matrix DR and the vector R given the vector X_o. The second part of the program (the main program NWRP) obtains $\begin{pmatrix} PR^*(r) \\ CR^*(r) \end{pmatrix}$ by solving for DR^{-1} R using Gauss-Jordan elimination rather than inverting DR directly.

Fig. 1 shows the flow diagram for subroutine RDR. The letters "A" through "U" are used for reference and correspond to those similarly marked portions of the program shown in the listing in Appendix A.

In "A" through "F" the program obtains P(PR) and $\frac{dP}{dPR}$ when K = 1 and C(CR) and $\frac{dC}{dCR}$ when K = 2. Dummy variables Q and $\frac{dQ}{dQR}$ are used with Q = PR for K = 1 and Q = CR for K = 2. This facet of the program makes use of the fact that the polynomial coefficients P(PR) and C(CR) are of the same form. In general, this might not be the case.

In "H" through "K" the matrix Δ [Eq. (15)] is evaluated. In "L" through "Q" the matrix DR is calculated except for $\frac{\partial \Delta}{\partial(PR,CR)}$. "L" through "Q" is used only once for each time RDR is called (that is, when the counter n = 1).

In "R" through "T" $\frac{\partial \Delta}{\partial(PR,CR)}$ is evaluated by perturbing the variables PR and CR. A counter n is varied from 2 to $(N - N_s + 1)$ to perturb the $N - N_s$ decision variables PR (using "R" and "S") and from $(N - N_s + 1)$ to $(N + 1)$ to vary the N_s state variables CR (using "R" and "T"). The numbers shown in the decision blocks refer to a case where N = 12 and $N_s = N - N_s = 6$.

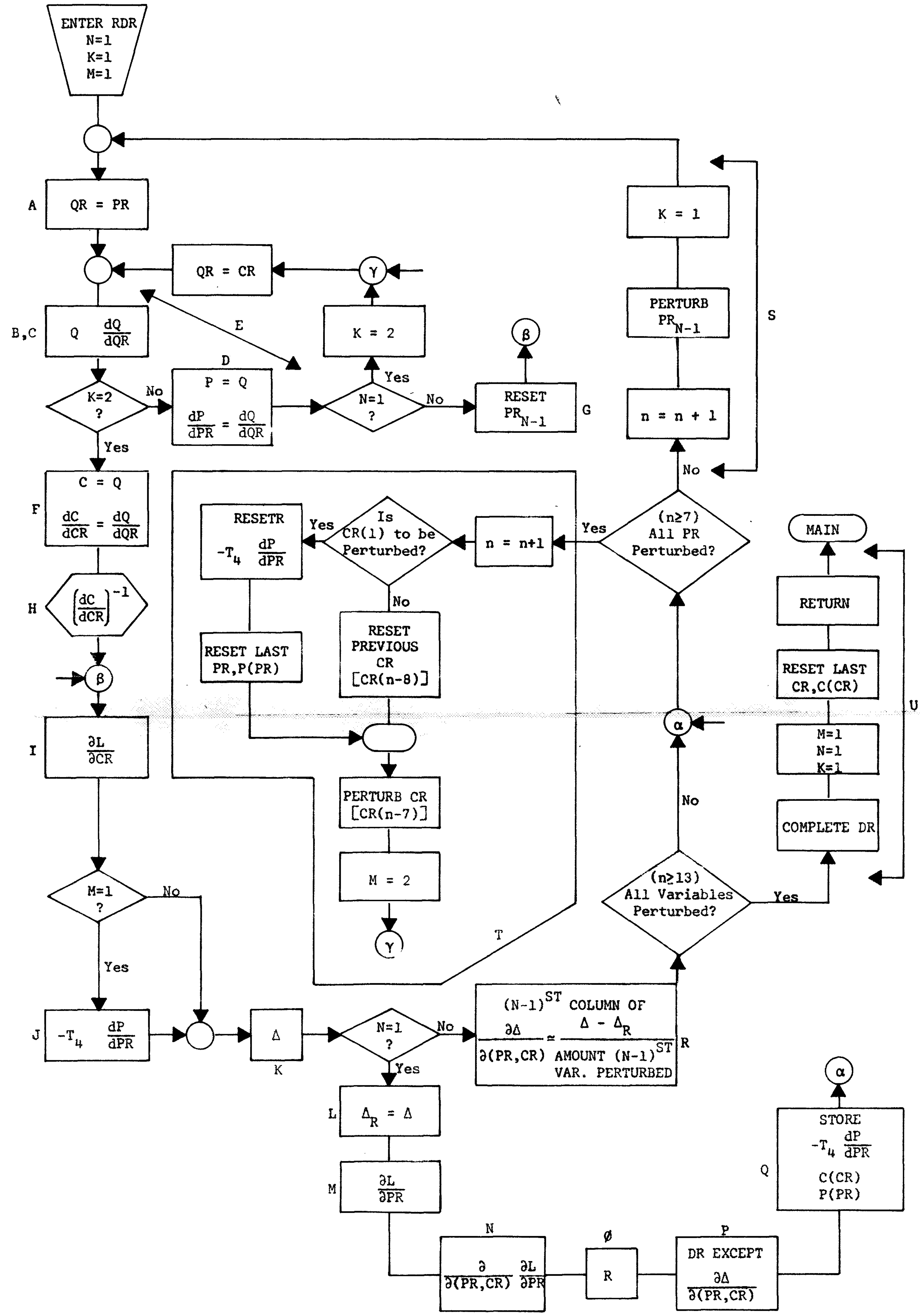


Fig. 1. Flow Diagram of Subroutine RDR.

Finally, when the counter $n = N + 1$, the matrix DR is completed in "U" and R and DR are returned to the main program. It should be noted that all equations other than the logic must be changed for each application of this program.

It is the job of the main program (NWRP) to monitor the Newton-Raphson procedure for solving the N equations in N unknowns. A flow diagram for NWRP is shown in Fig. 2. It is necessary to input to NWRP initial values of r [Eq. (2)], X_o, and the matrices T_4 and T_6 [Eq. (16)]. Then NWRP calls RDR and prints out the useful results. If the error (a measure of how well the equality constraints are met) is "small", a lower value of r is selected (until some minimum r value is reached). If the error is "large", one iterates up to 150 iterations and counts successive increases in the error. Eq. (16) is solved for $(DR^{-1})R$ by calling a subroutine INVERT in which DR and R are adjoined (e.g., [DR⋮R]) and row and column (Gauss-Jordan) elimination is done by seeking maximum pivotal elements, thus minimizing the effects of zero pivotal elements, round off, and ill conditioning. The main program also checks the possibility that Eq. (16) might result in a negative (forbidden) value for one of the variables. An arbitrarily small positive member replaces any resulting negative value, thus adding considerable penalty to the objective function [Eq. (2)]. Finally, X_o is replaced with the newest value of $X^*(r)$ and the process is repeated. In essence, the end result of one computing cycle is $X^*(r)$. The print statements call for, in order, the independent variables PR, the dependent variables CR, the vector R [Eqs. (14) and (16)], the functions P(PR), C(CR), the values of C(CR) that

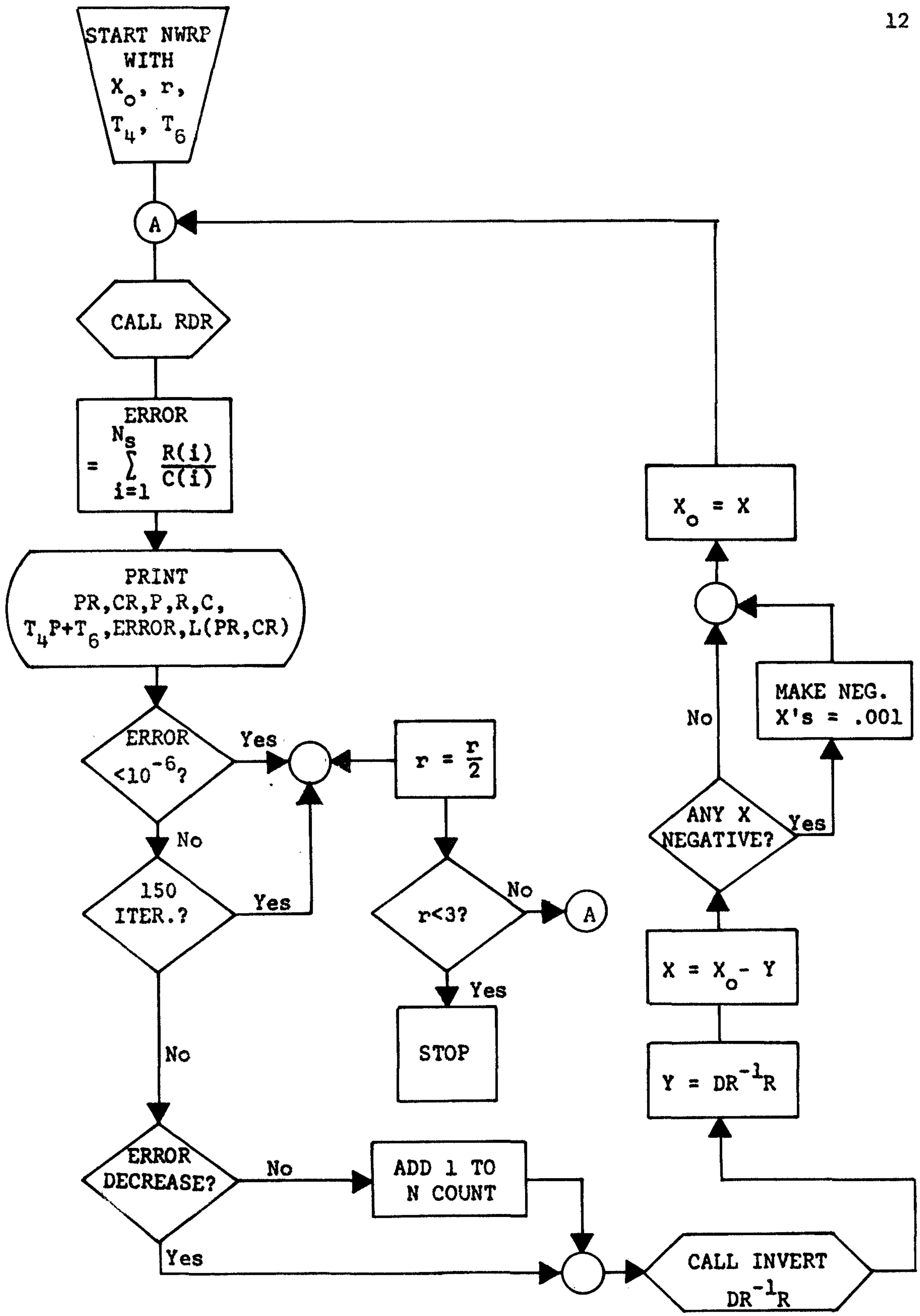


Fig. 2. Flow Diagram of Main Program (NWRP).

exactly satisfy the equality constraints (e.g., $C(CR) = [T_4]P(PR) + T_6$), the error (with regard to the equality constraints), and the objective function L(PR,CR) [Eq. (14)].

It will now be shown that the fuel valve servo problem may be structured as suggested in Eqs. (11) through (13) so that Eqs. (14) and (15) (and the program NWRP) may be used to obtain the design.

Application to the Fuel Valve Servo Problem

In the fuel valve servo problem, it is necessary to design a control system utilizing series and feedback compensation (Fig. 3).

We are given the polynomials $G_N(S)$ and $G_D(S)$ for the open loop plant and the polynomials $T_N(S)$ and $T_D(S)$ for the desired closed loop transfer function. The remaining polynomials P(S), C(S), and H(S) must be chosen to complete the design (equality constraints). Moreover, since the system must be realizable and reasonably insensitive to disturbances, the polynomials P(S) and C(S) must contain only left half plane roots (inequality constraints). One immediate question is: How can we ensure that P(S) and C(S) contain only left half plane roots, and, if possible, can we obtain some real valued variables to use later in the optimization process? Consider the following decomposition of the third order case for the polynomial P(S). Note that C(S) can be treated in exactly the same manner.

$$P(S) = S^3 + P_2S^2 + P_1S + P_0 = (S + P_{20})(S^2 + P_{11}S + P_{10}) \qquad (17)$$

Any order polynomial P(S) or C(S) may be similarly decomposed into the product of several second order polynomials and, if the original polynomial is of an odd order, one first order polynomial. In general, a polynomial such as P(S) or C(S) contains pairs of real roots and/or pairs of complex conjugate roots as well as one additional real root if the polynomial is of odd order. The conclusion to be reached is that coefficients such as P_{20}, P_{11}, and P_{10} are real valued, and if positive, the polynomial P(S) with coefficients P_2, P_1, and P_0 must have all left half S plane roots. Additionally, if P_{20}, P_{11}, and P_{10} are known, the actual poles are easily found if desired. The following definitions are made:

P = Coefficients of P(S) (i.e., P_2, P_1, P_0)

PR = Coefficients such as P_{20}, P_{11}, P_{10}

C = Coefficients of C(S) (i.e., C_2, C_1, C_0)

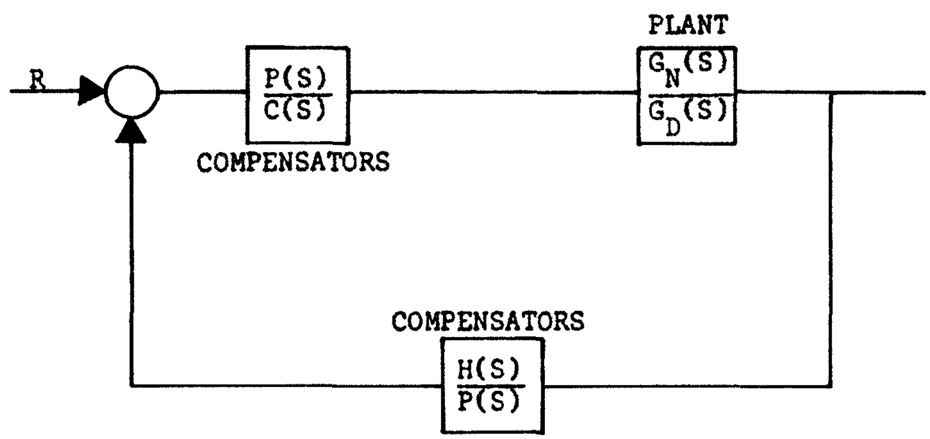
CR = Coefficients such as C_{20}, C_{11}, C_{10}

H = Coefficients of H(S)

The letter "R" in PR and CR shows that these coefficients are indicaters concerning the roots of P(S) and C(S) (i.e., if the PR and CR are positive, all the roots of P(S) and C(S) have negative real parts). The variables X to be used in the optimization process may now be chosen to be the vector $\begin{pmatrix} PR \\ CR \end{pmatrix}$. Then the transfer functions $\frac{P}{C}$ and $\frac{H}{P}$ are stable if C, P, and H are of the same order and if $\begin{pmatrix} PR \\ CR \end{pmatrix} \geq 0$. Thus, inequality constraints of the form G(S) $\geq$ 0 are defined by

$$\begin{pmatrix} PR \\ CR \end{pmatrix} \geq 0 \qquad (18)$$

From Fig. 3, it is evident that if $G_N(S) = T_N(S)$, then



$$\frac{Y}{R} = \frac{T_N(S)}{T_D(S)} = \frac{P(S)*G_N(S)}{C(S)*G_D(S) + G_N(S)*H(S)}$$

Fig. 3. Closed Loop Control System.

$$G_D(S)C(S) + G_N(S)H(S) = P(S)T_D(S) \tag{19}$$

The coefficients of C(S) and H(S) may then be related linearly to the coefficients of P(S) to realize the desired transfer function, thus providing equality constraints. If the coefficients of the highest powers of C(S) and P(S) are both one, then the following equality constraints result. The coefficients C depend only on the variables CR, and the coefficients P depend only on the variables PR.

$$\begin{bmatrix} C(CR) \\ H \end{bmatrix} = [T_1]P(PR) + T_2 \tag{20}$$

Partioning the above we obtain

$$0 = C(CR) - [T_4]P(PR) - T_6 \tag{21}$$

$$0 = H - [T_3]P(PR) - T_5 \tag{22}$$

Since the roots of H are not of particular concern, Eq. (21) is the equality constraint equation linearly relating the coefficients C and P but non-linearly relating the variables CR and PR. The above conditions require the following relations between the orders of the polynomials involved.

Polynomial	Order	Undetermined (Free) Coefficients
$T_D(S)$	i	0
C(S),CR(S)	i-1	i-1
P(S),PR(S)	i-1	i-1
H(S)	i-1	i

(23)

Consequently, for an ith order realization problem, there are $N = 2(i-1)$ variables (PR and CR) with $N-N_s = i-1$ independent (decision) variables and $N_s = i-1$ dependent (state) variables. The variables PR may be considered (arbitrarily) independent while the variables CR may be considered dependent.

Let us now choose an objective function $y(X)$ to be the low frequency sensitivity S_T^K of the closed loop transfer function,

$$T = \frac{T_N(S)}{T_D(S)},$$

to changes in the forward loop gain (K). Since $G_D(S)$, for the cases we shall consider, has a free S, let us define the low frequency sensitivity as

$$y(X) = S_T^K\,(PR,CR) = \underset{S\to 0}{\text{LIMIT}} \left[\frac{1}{S}\,\frac{T_N(S)}{T_D(S)}\,\frac{C(S)}{P(S)}\,\frac{G_D(S)}{G_N(S)}\right] \tag{24}$$

The fuel valve servo problem for $T_D(S)$ of order i may now be formulated into an optimization problem. We wish to optimize the objective function S_T^K (PR,CR) with PR the independent variables and CR the dependent variables. The inequality constraints (for stability) can be treated by minimizing a dual objective function, for a fixed number where $r>0$.

$$Y(PR,CR) = \tfrac{1}{2}\left[S_T^K\,(PR,CR) + r\sum_{j=1}^{i-1}\left(\frac{1}{PR_i} + \frac{1}{CR_i}\right)\right]^2 = \tfrac{1}{2}\left[L(PR,CR)\right]^2 \tag{25}$$

There are also i-1 equality constraints.

$$0 = C(CR) - [T_4]\, P(PR) - T_6 \qquad (26)$$

Eqs. (25) and (26) are of the form suggested in Eqs. (12) and (13); hence, Eq. (14) provides an algorithm for finding the variables $\begin{bmatrix} PR^*(r) \\ CR^*(r) \end{bmatrix}$ and the NWPR program can be used to carry out the algorithm. Once acceptable values of PR and CR are obtained, the polynomials P(S) and C(S) are determined. From Eq. (22), the coefficients of H(S) are obtained and the design is complete.

Two designs were obtained. Fig. 4 shows the final results for $T_D(S)$ of order i = 8; hence, there were 2(i-1) = 14 variables (7 independent and 7 dependent). Satisfactory results occurred after 119 iterations (43 seconds of central processor time). Fig. 5 shows the final results for $T_D(S)$ of order i = 7; hence, there were 12 variables (6 independent and 6 dependent). Satisfactory results occurred after 33 iterations (10 seconds of central processor time). In both cases, satisfactory results meant that the constraint Eq. (21) was satisfied to a high degree of accuracy. Then the final value of P(S) was taken, and Eq. (20) was used to specify the final values for C(S) and H(S), thus satisfying the constraint equations exactly with all the roots of P(S) and C(S) in the left half plane.

The above results represent a stable realization in which all roots of P(S) and C(S) are well damped (>0.7). In view of the many constraint equations and variables involved, the cost of the computing time must be considered minimal compared to the cost of man hours required by

$$\frac{T_N(S)}{T_D(S)} = \frac{3.056\times10^{30}}{S^8+2.498\times10^7S^7+6.231\times10^8S^6+8.194\times10^{12}S^5+6.724\times10^{16}S^4+3.721\times10^{20}S^3+1.311\times10^{24}S^2+2.486\times10^{27}S+3.056\times10^{30}}$$

$$\frac{G_N(S)}{G_D(S)} = \frac{3.056\times10^{30}}{S(S^7+2.911\times10^4S^6+5.897\times10^8S^5+8.579\times10^{12}S^4+4.036\times10^{16}S^3+1.596\times10^{20}S^2+1.664\times10^{23}S+1.905\times10^{26})}$$

$$P(S) = S^7+4.533\times10^4S^6+7.350\times10^8S^5+5.010\times10^{12}S^4+1.225\times10^{16}S^3+2.293\times10^{18}S^2+2.085\times10^{17}S+4.396\times10^{15}$$

$$C(S) = S^7+4.120\times10^4S^6+7.014\times10^8S^5+6.515\times10^{12}S^4+3.689\times10^{16}S^3+1.148\times10^{20}S^2+2.545\times10^{23}S+5.399\times10^{23}$$

$$H(S) = -(0.665S^7+1.841\times10^5S^6+1.387\times10^9S^5+7.583\times10^{12}S^4+1.726\times10^{16}S^3+1.701\times10^{19}S^2+7.657\times10^{21}S-4.396\times10^{15})$$

Roots of P(S)	Roots of C(S)	Roots of H(S)
-0.0577	-2.124	+22508
-0.0332	-1446+j3857	-1729+j1468
-203	-1446-j3857	-1729-j1468
-5179	-5318+j6644	-2161+j3862
-9263	-5318-j6644	-2161-j3862
-16080	-13832+j3946	-3319+j1776
-14600	-13832-j3946	-3319-j1776

Fig. 4. Design for an 8th Order Closed Loop Transfer Function.

$$\frac{T_N(S)}{T_D(S)} = \frac{5.120\times10^{26}}{S^7+1.898\times10^4 S^6+5.092\times10^8 S^5+5.139\times10^{12} S^4+3.641\times10^{16} S^3+1.536\times10^{20} S^2+3.890\times10^{23} S+5.120\times10^{26}}$$

$$\frac{G_N(S)}{G_D(S)} = \frac{5.120\times10^{26}}{S(S^6+2.807\times10^4 S^5+5.601\times10^8 S^4+7.997\times10^{12} S^3+3.212\times10^{16} S^2+1.286\times10^{20} S+1.011\times10^{23})}$$

$$P(S) = S^6+4.569\times10^4 S^5+6.898\times10^8 S^4+3.438\times10^{12} S^3+6.204\times10^8 S^2+2.216\times10^4 S+1.920\times10^{-2}$$

$$C(S) = S^6+3.660\times10^4 S^5+4.787\times10^8 S^4+3.001\times10^{12} S^3+1.053\times10^{16} S^2+3.314\times10^{18} S+3.425\times10^{19}$$

$$H(S) = 2.625S^6+3.229\times10^4 S^5-3.626\times10^8 S^4+7.505\times10^{11} S^3-1.442\times10^{15} S^2-3.816\times10^{18} S+1.920\times10^{-2}$$

Roots of P(S)	Roots of C(S)	Roots of H(S)
-8.888×10^{-7}	-3619+j5671	1839+j4395
-4.764×10^{-5}	-3619-j5671	1839-j4395
-1.319×10^{-4}	-14508+j647	16414
-12610	-14508-j647	-2439
-15550	-335	-3071
-17533	-10.698	-20176

Fig. 5. Design for a 7th Order Closed Loop Transfer Function.

a trial-and-error approach to the same problem. The information needed by the computer to design the system includes the constraint equation matrices in Eqs. (20) to (22), initial guesses at all the variables ($PR_j = CR_j = 1000$ was used for $j = 1, 2, \ldots, i-1$), initial value for r in Eq. (25) ($r = 5$ was used), and explicit values for all the partial derivatives appearing in Eq. (14) (except for partials of Δ in Eq. (15), which partials are obtained by perturbing the variables PR and CR).

Two alternate approaches were tried for the fuel valve servo problem. Both approaches were aimed at simplifying the calculation of the matrix DR [Eqs. (14) and (16)]. In the first, using a Fletcher-Powell technique, DR^{-1} was estimated where in the second, an original approach to the problem, a somewhat different treatment was attempted.

The crux of the Fletcher-Powell approach is that the vector R [Eqs. (14) and (16)] is the gradient vector for some objective function Z(X). However, in this case, the vector R consists of constrained derivatives and equality constraints. It is felt that the fact that R is not a gradient vector explains the inability of the technique to converge to an answer. Briefly, if R were a gradient vector for some Z(X), then the following algorithm estimates DR^{-1} and minimizes Z(X) with quadratic convergence. Let H_o = initial guess of $(DR|_{Xo})^{-1}$.

$$S = -H_o R|_{Xo}$$

$$\text{LET } X = X_o + \alpha S \quad (\alpha \text{ is a scalar})$$

$$\text{OBTAIN } \alpha^* \text{ SO THAT } \frac{dZ[X(\alpha)]}{d\alpha} = 0$$

$$\text{THEN } X^* = X_o + \alpha^* S$$

$$\text{LET } Y = R|_{X^*} - R|_{Xo}$$

$$(DR|_{X^*}) \approx H_o + \frac{\alpha S S^T}{S^T Y} - \frac{H_o Y Y^T H_o}{Y^T H_o Y}$$

$$H_o = (DR|_{X^*})^{-1}$$

$$X_o = X^*$$

REPEAT PROCESS (27)

The major difficulty in applying the Fletcher-Powell approach lies in the one-dimensional gradient search for α^*. An attempt was made at finding α^* so that

$$\left(\frac{\partial Z(X,\alpha)}{\partial X}\right)^T \frac{dX}{d\alpha} = 0 = (R|_{Xo+\alpha S})^T S \equiv G(\alpha) \tag{28}$$

A Newton-Raphson technique was used to find α^* such that $G(\alpha^*) \approx 0$. This procedure never did converge to a satisfactory result (satisfying the equality constraints). Additionally, the computing time taken by the one-dimensional search for α^* caused each iteration of Fletcher-Powell to take as much time as each iteration of the Newton-Raphson technique in which $(DR|_{Xo})^{-1}$ is found directly.

The second alternate procedure is aimed at satisfying the equality and inequality constraints only. A vector R [as in Eq. (16)] is so chosen that, if every element is near zero, then all constraints are satisfied, and the design is completed. Also, the matrix DR is simple to compute. As in the Fletcher-Powell case (above) the vector R is neither the gradient of an objective function nor a vector of constrained derivatives and equality constraints; hence, convergence is not guaranteed. The vector R is chosen as follows, where the upper half is chosen to satisfy the equality constraints and the bottom half is to satisfy the inequality constraints:

$$R = \begin{bmatrix} C(CR) - [T_4]P(PR) - T_6 \\ \hline \frac{1}{PR_1}+\frac{1}{CR_1} & & \\ & \ddots & \\ & & \frac{1}{PR_{i-1}}+\frac{1}{CR_{i-1}} \end{bmatrix} \tag{29}$$

Then the matrix DR is

$$DR = \frac{\partial R}{\partial(PR,CR)} = \left[\begin{array}{ccc|ccc} & -T_4\,\frac{dP}{dPR} & & & \frac{dC}{dCR} & \\ \hline -\frac{1}{PR_1^{\,2}} & & & -\frac{1}{CR_1^{\,2}} & & \\ & \ddots & & & \ddots & \\ & & -\frac{1}{PR_{i-1}^{\,2}} & & & -\frac{1}{CR_{i-1}^{\,2}} \end{array}\right] \tag{30}$$

With the above definitions of R and DR, Eq. (16) can be used to iteratively obtain the solution $\begin{pmatrix} PR^* \\ CR^* \end{pmatrix}$. The above procedure took 1/3 of the computing time per iteration required by the SUMT--constrained derivative --Newton-Raphson procedure, had far better convergence than did the Fletcher-Powell prodedure, but had more erratic behavior than the SUMT--

constrained derivative--Newton-Raphson procedure. It is felt that Eqs. (29) and (30) hold some promise as a method of solving the equality and inequality constraint, but that the SUMT--constrained derivative--Newton-Raphson technique is far superior.

Summary and Conclusions

The fuel valve servo problem requires the design of a control system utilizing series and feedback compensation (Fig. 3). Given are the polynomials $G_N(S)$ and $G_D(S)$ for the open loop plant and $T_N(S)$ and $T_D(S)$ for the closed loop plant. The remaining polynomials P(S), C(S), and H(S) must be chosen to complete the design (equality constraints). Moreover, since the system must be realizable and reasonably insensitive to disturbances, the polynomials P(S) and C(S) must contain only left half plane roots (inequality constraints). The polynomials P(S) and C(S) are factored into first and second order polynomials, and the resulting coefficients become the variables. In general, if $G_N(S)$ and $T_N(S)$ are constants and if $T_D(S)$ and $G_D(S)$ are ith order, P(S) and C(S) are (i-1)th order and there are 2 (i-1) variables, half of which are independent and half of which are dependent. The design problem is restructured into an optimization problem. The objective function is low frequency sensitivity [Eq. (24)]. A Sequential Unconstrained Minimization Technique (SUMT) is used to treat the inequality constraints [Eqs. (2) and (25)]. The constraint equation [Eq. (21)] is treated using the constrained derivative approach [Eqs. (6) to (9)]. The resulting Newton-Raphson algorithm [Eq. (14)] was used to obtain the designs shown in Figs. 4 and 5.

The cost of computing time must be considered minimal as compared to the cost of man hours required to solve the above multi-variable problem by trial-and error. The only difficulty in setting up the program lies in obtianing the required partial derivatives [Eq. (14)] which is somewhat tedious.

Alternate procedures (Fletcher-Powell and a different choice of the vector R) were attempted, but the SUMT--constrained derivative--Newton-Raphson technique was far superior, and the resulting design exhibited excellent damping characteristics for the roots of P(S) and C(S).

```
STEFAN,T144,CM60000,BN3900043X.
RUN(S)
LGO.

      PROGRAM NWRP(INPUT,OUTPUT)
      DIMENSION X(12),R(12),Y(12),XI(12),LOC(12),A(12,13),CK(12),
     1DR(12,12),T4(6,6),T6(6),C(6),P(6),CH(6)
     2,D(12,12),YF(12)
C INPUT MATRICES T4 AND T6
      READ 51,((T4(I,J),J=1,6),I=1,6),T6
   51 FORMAT(6(6E10.3/),6E10.3)
      PRINT 52,T4,T6
   52 FORMAT(6(1X,6E12.3/),1X,6E12.3)
C  MULTIPLIER R IS CALLED S IN THIS PROGRAM. SET INITIAL VALUE
      S=5.
      FACTOR=2
C  O AND N IS THE NUMBER OF VARIABLES M =N+1
      O=12
      N=12
      M=13
C ZERO OUT STORAGE
      DO 18 I=1,N
      Y(I)=0.
   18 R(I)=0.
      DO 19 I=1,N
      DO 19 J=1,N
   19 DR(I,J)=0.
C INITIAL GUESS AT PR AND CR
      DO 50 I=1,N
   50 X(I)=10000.
   28 CONTINUE
      ITER=0
      PRINT 20
   20 FORMAT(1H1)
      NCOUNT=0
      ENM1=1000.
C  CALCULATE VECTOR R AND MATRIX OF PARTIALS DR
   10 CALL RDR(X,O,R,DR,S,FCN,T4,T6,P,C)
C  CALCULATE ERROR IN EQUALITY CONSTRAINTS
      ERROR=0.
      NP=N/2
      DO 1 I=1,NP
    1 ERROR=ERROR+R(I+NP)/C(I)
C  CALCULATE VALUE OF C(CR) THAT EXACTLY SATISFIES EQUALITY CONSTRAINTS--CH(I
      DO 53 I=1,NP
      CH(I)=T6(I)
      DO 53 L=1,NP
   53 CH(I)=CH(I)+T4(I,L)*P(L)
C ITER IS NO OF ITERATIONS. NCOUNT IS NO OF COSECUTIVE ERROR INCREASES
    8 PRINT 2, ITER,NCOUNT,       (X(I),I=1,N),
     1(R(I),I=1,N),P,C,CH,ERROR,FCN
    2 FORMAT(1H ,21H NUMBER OF ITERATIONS,I5,
     127H SUCCESSIVE ERROR INCREASES,I5/
     24H PR ,7X,6E14.5/4H CR ,7X,6E14.5/11H R(1)-R(6) ,
     36E14.5/11H R(7)-R(12),6E14.5/3H P ,8X,6E14.5/
     43H C ,8X,6E14.5/11H CORRECT C ,6E14.5/6H ERROR,5X,
     52E14.5,18H VALUE OF L(PR,CR)/)
C  IF ERROR IS SMALL, LOWER S
      IF(ABS(ERROR)-1.E-6) 3,3,4
C HAVE 150 ITERATION BEEN RUN
```

```
    4 IF(ITER-150) 5,3,3
    5 CONTINUE
C   COUNT SUCCESSIVE ERROR INCREASES
      IF(ABS(ERROR)-ABS(ENM1)) 13,15,15
   13 NCOUNT=0
      ENM2=ENM1
      ENM1=ERROR
      GO TO 14
   15 NCOUNT=NCOUNT+1
      ENM2=ENM1
      ENM1=ERROR
   14 CONTINUE
C   GET SOLUTION OF (DR-1)*R  USING GAUSS JORDAN ELIMINATION
      CALL INVERT (A,Y,LOC,CK,R,DR,N,M)
      ITER=ITER+1
C CALCULATE NEW VALUES FOR VARIABLES
      DO 9 I=1,N
    9 X(I)=X(I)-Y(I)
C LET NO VARIABLE BE LESS THAN .CO1
      DO 31 I=1,N
      IF(X(I)) 32,31,31
   32 X(I)=.001
   31 CONTINUE
       GO TO 10
C   REDUCE S
    3 S=S/FACTOR
C   TERMINATE FOR S LESS THAN 3
      IF(S-3) 27,27,29
   29 CONTINUE
C   ITERATE AGAIN WITH LOWER VALUE OF S
      GO TO 28
   27 STOP
      END
```

```
      SUBROUTINE INVERT (A,Y,LOC,CK,R,DR,N,M)
      DIMENSION A(N,M),Y(N),LOC(N),CK(N),R(N),DR(N,N)
      DO 7 I=1,N
      DO 7 J=1,N
    7 A(I,J)=DR(I,J)
      DO 8 I=1,N
    8 A(I,N+1)=R(I)
      NP=N+1
      DO 1 I=1,N
    1 CK(I)=0.
      DO 100 I=1,N
      IP=I+1
C     FIND MAX ELEMENT IN I-TH COL
      AMAX=0.
      DO 2 K=1,N
      IF(AMAX-ABS(A(K,I))) 3,2,2
C     IS NEW MAX IN ROW PREVIOUSLY USED AS PIVOT
    3 LCK=CK(K)
      IF(LCK) 4,4,2
    4 LOC(I)=K
      AMAX=ABS(A(K,I))
    2 CONTINUE
C     MAX ELEMENT IN I-TH COL IS A(L,I)
    5 L=LOC(I)
      CK(L)=1.
C     PERFORM ELIMINATION,L IS PIVOT ROW,A(L,I),IS PIVOT ELEMENT
      DO 50 J=1,N
      IF(L-J) 6,50,6
    6 F=-A(J,I)/A(L,I)
      DO 40 K=IP,NP
   40 A(J,K)=A(J,K)+F*A(L,K)
   50 CONTINUE
  100 CONTINUE
      DO 200 I=1,N
      L=LOC(I)
  200 Y(I)=A(L,N+1)/A(L,I)
      RETURN
   99 NFLAG=1
      RETURN
      END
```

```
      SUBROUTINE ROP(X,Q,R,DR,S,FCN,T4,T6,P,C)
      DIMENSION C(6),P(6),DQ(7,7),DP(6,6),DC(6,6),DR(12,12),R(12),
     1DLS(6),DLD(6),T4DPM(6,6),DCI(6,6),T4(6,6),DELTA(6),DDLD(6,12),
     2TEMP(6), T4DPS(6,6),DELTR(6),DELTD(6),DDELT(6,12),PR(6),CR(6),T6
     3(6),PS(6),CS(6),X(12)
      K=1
      N=1
      M=1
C  NP IS THE ORDER OF PR(AND CR).NP2 IS TOTAL NO. OF VARIABLES
      NP=Q/2
      NP2=Q
C  VALUES OF PR AND CR
      DO 46 I=1,NP
      PR(I)=X(I)
   46 CR(I)=X(I+NP)
```

A

```
    1 Q40=0.
      Q31=PR(1)
      Q30=PR(2)
      Q21=PR(3)
      Q20=PR(4)
      Q11=PR(5)
      Q10=PR(6)
C  CALCULATE P(PR) AND DP/DPR WHEN K =1
C  CALCULATE C(CR) AND DC/DCR WHEN K =2
C  Q IS USED TO CALCULATE P(PR) AND C(CR)
C  DQ IS USED TO CALCULATE DP/DPR AND DC/DCR
```

B

```
    2 Q6=Q40+Q31+Q21+Q11
      Q5=Q31*Q40+Q30+Q21*Q40+Q21*Q31+Q20+Q11*Q40+Q11*Q31+Q11*Q21+Q10
      Q4=Q30*Q40+Q21*Q31*Q40 +Q21*Q30+Q20*Q40+Q20*Q31+Q11*Q31*Q40
     1+Q11*Q30+Q11*Q21*Q40+Q11*Q21*Q31+Q11*Q20+Q10*Q40+Q10*Q31+Q10*Q21
      Q3=Q21*Q30*Q40+Q20*Q31*Q40+Q20*Q30+Q11*Q30*Q40+Q11*Q21*Q31*Q40
     1+Q11*Q21*Q30+Q11*Q20*Q40+Q11*Q20*Q31+Q10*Q31*Q40+Q10*Q30
     2+Q10*Q21*Q31+Q10*Q21*Q40+Q10*Q20
      Q2=Q20*Q30*Q40+Q11*Q21*Q30*Q40+Q11*Q20*Q31*Q40+Q11*Q20*Q30
     1+Q10*Q30*Q40+Q10*Q21*Q31*Q40+Q10*Q21*Q30+Q10*Q20*Q31+Q10*Q20*Q40
      Q1=Q11*Q20*Q30*Q40+Q10*Q21*Q30*Q40+Q10*Q20*Q31*Q40+Q10*Q20*Q30
```

C

```
      DQ(1,2)=1.
      DQ(1,3)=0.
      DQ(1,4)=1.
      DQ(1,5)=0.
      DQ(1,6)=1.
      DQ(1,7)=0.
      DQ(2,2)=Q40+Q21+Q11
      DQ(2,3)=1.
      DQ(2,4)=Q40+Q31+Q11
      DQ(2,5)=1.
      DQ(2,6)=Q40+Q31+Q21
      DQ(2,7)=1.
      DQ(3,2)=Q21*Q40+Q20+Q11*Q40+Q11*Q21+Q10
      DQ(3,3)=Q40+Q21+Q11
      DQ(3,4)=Q31*Q40+Q30+Q11*Q40+Q11*Q31+Q10
      DQ(3,5)=Q40+Q31+Q11
      DQ(3,6)=Q31*Q40+Q30+Q21*Q40+Q21*Q31+Q20
      DQ(3,7)=Q40+Q31+Q21
      DQ(4,2)=Q20*Q40+Q11*Q21*Q40+Q11*Q20+Q10*Q40+Q10*Q21
      DQ(4,3)=Q21*Q40+Q20+Q11*Q40+Q11*Q21+Q10
      DQ(4,4)=Q30*Q40+Q11*Q31*Q40+Q11*Q30+Q10*Q31+Q10*Q40
      DQ(4,5)=Q31*Q40+Q30+Q11*Q40+Q11*Q31+Q10
      DQ(4,6)=Q30*Q40+Q21*Q31*Q40+ Q21*Q30+Q20*Q40+Q20*Q31
      DQ(4,7)=Q31*Q40+Q30+Q21*Q31+Q21*Q40+Q20
```

```
      DQ(5,2)=Q11*Q20*Q40+Q10*Q21*Q40+Q10*Q20
      DQ(5,3)=Q20*Q40+Q11*Q21*Q40+Q11*Q20+Q10*Q40+Q10*Q21
      DQ(5,4)=Q11*Q30*Q40+Q10*Q31*Q40+Q10*Q30
      DQ(5,5)=Q30*Q40+Q10*Q31+Q11*Q31*Q40+Q10*Q40+Q11*Q30
      DQ(5,6)=Q21*Q30*Q40+Q20*Q31*Q40+Q20*Q30
  C   DQ(5,7)=Q30*Q40+Q21*Q31*Q40+Q21*Q30+Q20*Q31+Q20*Q40
      DQ(6,2)=Q10*Q20*Q40
      DQ(6,3)=Q11*Q20*Q40+Q10*Q21*Q40+Q10*Q20
      DQ(6,4)=Q10*Q30*Q40
      DQ(6,5)=Q11*Q30*Q40+Q10*Q31*Q40+Q10*Q30
      DQ(6,6)=Q20*Q30*Q40
      DQ(6,7)=Q21*Q30*Q40+Q20*Q31*Q40+Q20*Q30
      IF(K-2) 4,3,3
C   STORE P AND DP/DPR
    4 P(1)=Q6
      P(2)=Q5
      P(3)=Q4
      P(4)=Q3
      P(5)=Q2
      P(6)=Q1
  D   P40=Q40
      P31=Q31
      P30=Q30
      P21=Q21
      P20=Q20
      P11=Q11
      P10=Q10
      DO 17 I=1,NP
      DO 17 J=I,NP
   17 DP(I,J)=DQ(I,J+1)
      IF(N-1) 15,15,10
   15 Q40=0.
      Q31=CR(1)
      Q30=CR(2)
  E   Q21=CR(3)
      Q20=CR(4)
      Q11=CR(5)
      Q10=CR(6)
      K=2
      GO TO 2
C   RESET PR(N-1)
   10 PR(N-1)=PR(N-1)/1.001
  G   EPS=.001*PR(N-1)
      GO TO 43
C   STORE C AND DC/DCR
    3 C(1)=Q6
      C(2)=Q5
      C(3)=Q4
      C(4)=Q3
      C(5)=Q2
  F   C(6)=Q1
      C40=Q40
      C31=Q31
      C30=Q30
      C21=Q21
      C20=Q20
      C11=Q11
      C10=Q10
      DO 18 I=1,NP
      DO 18 J=1,NP
```

```
   16 DC(I,J)=DD(I,J+1)
F        DO 60 I=1,NP
         DO 60 J=1,NP
   60 DCI(I,J)=DC(I,J)
C  CALCULATE INVERSE OF DC/DCR
4        CALL MATRIX(10,NP,NP,D,DCI,NP,DETERP)
C  CALCULATE DL/DCR CALLED DLS(I)
   43 CON=(1.905/3.056)*.0001*C(6)/P(6)
         DLS(1)=-S/(C31**2)
         DLS(2)=CON/C30-S/(C30**2)
I        DLS(3)=-S/(C21**2)
         DLS(4)=CON/C20-S/(C20**2)
         DLS(5)=-S/(C11**2)
         DLS(6)=CON/C10-S/(C10**2)
         IF(M-1) 5,5,6
C  CALCULATE -T4*(DP/DPR) CALLED T4DPM(I)
    5 DO 19 I=I,NP
J        DO 19 J=1,NP
         T4DPM(I,J)=0.
         DO 19 L=1,NP
   19 T4DPM(I,J)=T4DPM(I,J)-T4(I,L)*DP(L,J)
    6 CONTINUE
C  CALCULATE MATRIX DELTA
         DO 7 I=1,NP
         TEMP(I)=0.
K        DO 7 L=1,NP
    7 TEMP(I)=TEMP(I)+DLS(L)*DCI(L,I)
         DO 20 I=1,NP
         DELTA (I)=0.
         DO 20 L=I,NP
   20 DELTA(I)=TEMP(L) *T4DPM(L,I)+DELTA(I)
         IF(N-1) 9,9,8
C  CALCULATE MATRIX DR EXCEPT FOR PARTIALS OF DELTA
L   9 DO 30 I=1,NP
   30 DELTP(I)=DELTA(I)
C  CALCULATE DL/DPR CALLED DLD(I)
         CON=-(1.905/3.056)*(C(6)/P(6))*.0001
         DLD(1)=-S/(P31**2)
M        DLD(2)=CON/P30-S/(P30**2)
         DLD(3)=-S/(P21**2)
         DLD(4)=CON/P20-S/(P20**2)
         DLD(5)=-S/(P11**2)
         DLD(6)=CON/P10-S/(P10**2)
C  CALCULATE PARTIALS OF DL/DPR CALLED DDLD(I,J)
         DO 37 I=1,NP
         DO 37 J=1,NP2
   37 DDLD(I,J)=0.
         DDLD(2,2)=-2./(P30**2)
         DDLD(2,4)=-1./(P20*P30)
         DDLD(2,6)=-1./(P10*P30)
N        DDLD(2,8 )=1./(P30*C30)
         DDLD(2,10)=1./(P30*C20)
         DDLD(2,12)=1./(P30*C10)
         DDLD(4,2)=DDLD(2,4)
         DDLD(4,4)=-2./(P20**2)
         DDLD(4,6)=-1./(P20*P10)
         DDLD(4,8 )=1./(P20*C30)
         DDLD(4,10)=1./(P20*C20)
         DDLD(4,12)=1./(P20*C10)
         DDLD(6,2)=DDLD(2,6)
```

```
      DDLD(6,4)=DDLD(4,6)
      DDLD(6,6)=-2./(P10**2)
      DDLD(6,8 )=1./(P10*C30)
      DDLD(6,10)=1./(P10*C20)
      DDLD(6,12)=1./(P10*C10)
N     DO 38 I=1,NP
      DO 38 J=1,NP2
   38 DDLD(I,J)=DDLD(I,J)*CON
      DO 39 I=1,NP
   39 DDLD(I,I)=DDLD(I,I)+2.*S/(P(I   )**3)
C  CALCULATE VECTOR R
      DO 16 I=1,NP
   16 R(I)=DLD(I)-DELTA(I)
      NPM1=NP-1
Ø     DO 21 I=1,NP
      R(I+NP)=0.
      DO 22 L=1,NP
   22 R(I+NP)=R(I+NP)+T4(I,L)*P(L)
   21 R(I+NP)=C(I)-R(I+NP)-T6(I)
C  STORE ABOVE IN DR
      DO 23 I=1,NP
      DO 23 J=1,NP2
   23 DR(I,J)=DDLD(I,J)
P     DO 24 I=1,NP
      DO 24 J=1,NP
   24 DR(I+NP,J)=T4DPM(I,J)
      DO 25 I=1,NP
      DO 25 J=1,NP
   25 DR(I+NP,J+NP)=DC(I,J)
C  STORE FOR FUTURE USE
      DO 26 I=1,NP
      DO 26 J=1,NP
   26 T4DPS(I,J)=T4DPM(I,J)
      DO 41 I=1,NP
      PS(I)=P(I)
   41 CS(I)=C(I)
Q     FCN=0.
      DO 48 I=1,NP
   48 FCN=FCN+S*(1./CR(I)+1./PR(I))
      FCN=FCN+(1.305/3.056)*(C(6)/P(6))*.0001
      DO 52 I=1,NP
      DO 52 J=1,NP
   52 DR(I,J)=FCN*DR(I,J)+R(I)*DLD(J)
      DO 53 I=1,NP
      DO 53 J=1,NP
   53 DR(I,J+NP)=FCN*DR(I,J+NP)+R(I)*DLS(J)
      DO 54 I=1,NP
   54 R(I)=FCN*R(I)
      GO TO 13
C  APPROXIMATE PARTIALS OF DELTA USING PERTURBATIONS.CALLED DDELT(I,J)
    8 DO 31 I=1,NP
R  31 DELTD(I)=DELTA(I)
      DO 32 J=1,NP
   32 DDELT(J,N-1)=(DELTD(J)-DELTR(J))/EPS
      IF(N-1-NP2) 12,11,11
   12 IF(N-1-NP) 13,14,14
C  PERTURB PR(N-1)
   13 N=N+1
S     PR(N-1)=PR(N-1)*1.001
      K=1
```

S
```
      GO TO 1
   14 N=N+1
      IF(N-2-NP) 28,28,27
C  USE T4DPM FOR NOMINAL(UNPERTURBED) VALUES OF PR
   28 DO 35 I=1,NP
      DO 35 J=1,NP
   35 T4DPM(I,J)=T4DPS(I,J)
C  RESET LAST PR AND P(PR)
      P10=PR(NP)
      DO 42 I=1,NP
   42 P(I)=PS(I)
      GO TO 29
C  RESET CR(N-2-NP) AND PERTURB CR(N-1-NP)
   27 CR(N-2-NP)=CR(N-2-NP)/1.001
   29 CR(N-1-NP)=CR(N-1-NP)*1.001
      EPS=.001*CR(N-1-NP)/1.001
      M=2
      GO TO 15
   11 DO 36 I=1,NP
      DO 36 J=1,NP2
   36 DR(I,J)=DR(I,J)-DDELT(I,J)*FCN
C  RESET LAST CR AND C(CR).COMPLETE DR BY INCLUDING PARTIALS OF DELTA
      CR(N-1-NP)=CR(N-1-NP)/1.001
      C10=CR(NP)
      DO 47 I=1,NP
   47 C(I)=CS(I)
      M=1
      K=1
      N=1
      RETURN
      END
```
T

U


```
 1.000E+00
-9.093E+03 1.000E+00
 2.044E+08-9.093E+03 1.000E+00
-3.504E+12 2.044E+08-9.093E+03 1.000E+00
 6.090E+16-3.504E+12 2.044E+08-9.093E+03 1.000E+00
-1.065E+21 6.090E+16-3.504E+12 2.044E+08-9.093E+03 1.000E+00
-9.093E+03 2.044E+08-3.504E+12+6.090E+16-1.065E+21 1.870E+25
```

PARALLEL TANGENTS

Introduction

The method of Parallel Tangents (or Partan) as developed by Shah, Buehler and Kepthorne [1] is ideally suited to cost functions of the form

$$\omega = Q(\bar{Z})$$

$$\text{Cost} = y = M(\omega) = Y(\bar{Z})$$

where

$$\bar{Z} = (Z_1, Z_2, Z_3, \ldots, Z_N)$$

and Q and M are quadratic and monotonic cost functions respectively. The individual parameters (Z_i) are unconstrained.

In this report, a particular form of Partan known as Continued Gradient Partan is discussed. The algorithm for solving the class of cost functions defined above is given along with general comments regarding Partan's efficiency. This is followed by a description of the Partan computer program including modifications needed for problems having constrained parameters. After a section giving I/O formats and program flow charts, the application of Partan to the Lewis Fuel Valve problem is described.

Optimization Procedure

The Partan algorithm serves as a master program for the parameter search, initiating the search at any given starting point, and then guiding the search until termination. The Partan procedure consists of several parts:

1. The Partan algorithm
2. The gradient calculation
3. The vector search (Golden Section)
4. The cost calculation
5. The constraint calculation (if any)

The Partan Algorithm

Let the set of parameters at any step (j) of the search be denoted by

$$\bar{Z}_j = [Z_1(j), Z_2(j), \ldots, Z_N(j)]$$

Then $\bar{Z}_o$ corresponds to the given starting point. According to the Partan algorithm (illustrated graphically in Fig. 1), the cost function gradient (∇y) is evaluated at $\bar{Z}_o$ (and subsequently at $\bar{Z}_2$, $\bar{Z}_4$, . . . , $\bar{Z}_{even}$) and the optimum along that gradient vector is found at $\bar{Z}_2$ (and subsequently $\bar{Z}_3$, $\bar{Z}_5$, . . . , $\bar{Z}_{odd}$). This step is called a gradient step. When the search routine reaches $\bar{Z}_3$ (and later $\bar{Z}_5$, $\bar{Z}_7$, . . . , $\bar{Z}_{odd}$), the optimum $\bar{Z}_4$ is then found along the vector $(\bar{Z}_3 - \bar{Z}_o)$ or in general $(\bar{Z}_{odd} - \bar{Z}_{odd-3})$. This is known as an acceleration step.

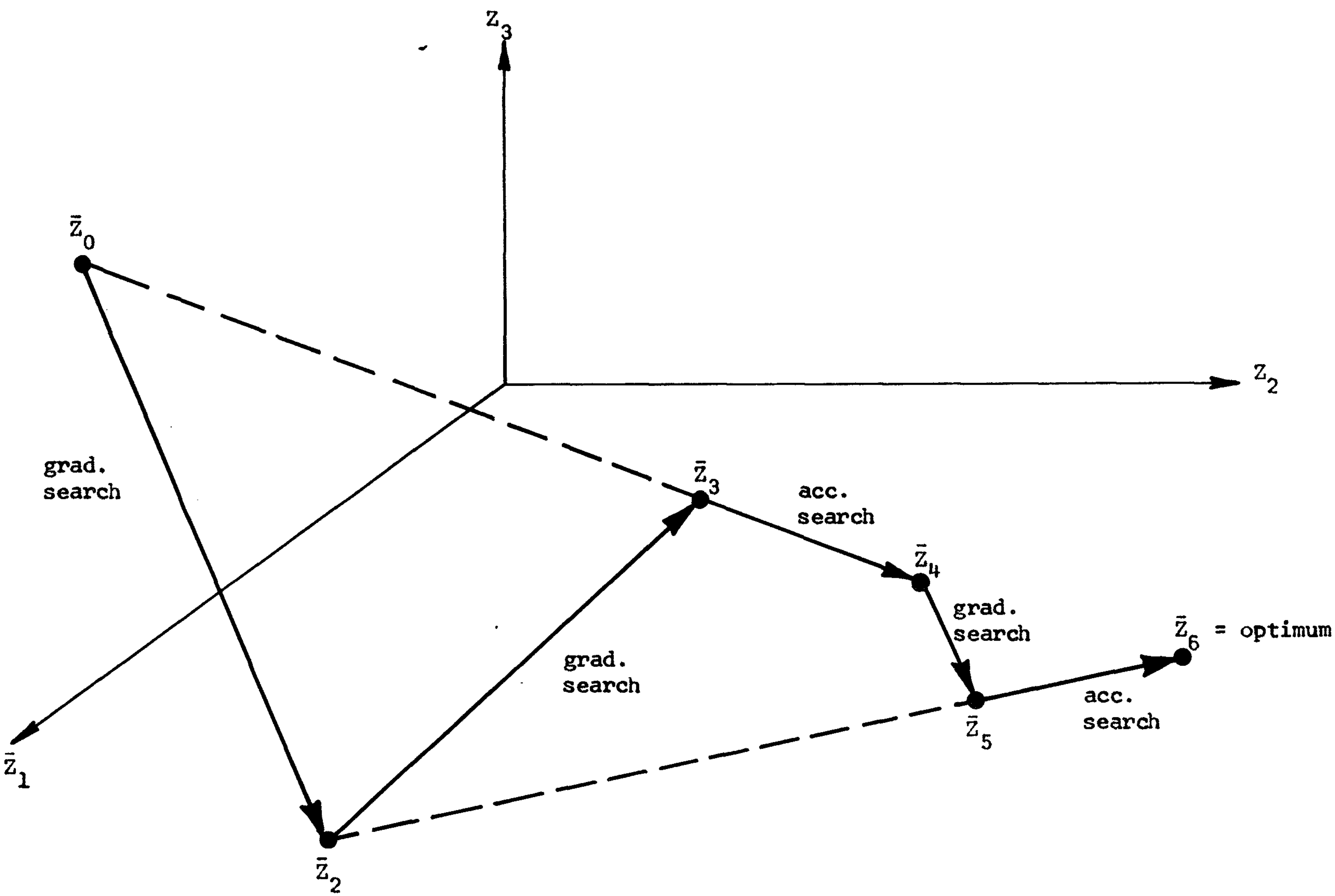


Fig. 1. Graphic Outline of Partan Search.

If the individual vector optimum points $\bar{Z}_j$ are exactly determined along precisely computed vectors, the optimum set of parameters having a quasi-quadratic cost will be found in 2N-1 steps. No other procedure has been shown to guarantee such convergence.

If the cost function is not quasi-quadratic, it is often not desirable to exactly determine the vectors and their resulting optimum points (see Harkins [3]). This inexactness introduces some amount of randomness into the search which is beneficial for problems having cost functions with highly complex contours in the parameter space.

Gradient Calculation

The gradient procedure estimates y by making perturbations about each $\bar{Z}_{even}$ and measuring the change in cost. This method is not only desirable for non-quasi-quadratic problems but is often necessary when ∇y is too complex to determine analytically. The method does, however, require extra cost function evaluations and is therefore suitable only to problems where the cost may be quickly computed.

In the gradient procedure, each

$$\Delta y_i = Y(Z_1, \ldots, Z_i + \alpha R_i, \ldots, Z_N) - Y(\bar{Z})$$

$$i = 1, 2, \ldots, N$$

is evaluated where R_i is the range estimation of Z_i and α, a constant (usually 10^{-4}). Then the norm

$$|\Delta y| = \left[\sum_{i=1}^{N} \Delta y_i^{\,2} \right]^{\frac{1}{2}}$$

is found and the incremental change (D_i) for each Z_i is computed,

$$D_i = \beta R_i(\Delta y_i/\Delta y)$$

where β is the "step size" constant supplied by the calling program.

Vector Search

The particular vector search used is the golden-section search (see Wilde and Beightler [2]). Using the D_i's supplied by the gradient subprogram in the case of a gradient search or using D_i's given by

$$D_i = (Z_i - Z_{i-3})/3$$

for an acceleration search, the Z_i's are stepped along the vector (according to the size of β) and the cost (y) is computed at each point. The search proceeds by either expanding or contracting step sizes until the optimal point along the vector is reached.

As an optimal vector point (or eventually the optimum) is approached, β decreases. During each vector search, β is allowed to decrease only a fixed number of times (3 if the cost function is "ridgy" or 5 if it's "smooth") while there is no restriction on the number of increases. However, if during a vector search, β decreases below a level E selected by the program user, the Partan search is terminated.

The cost subprogram must be supplied by the user. The calling program provides the current value of $Z_1, Z_2, \ldots, Z_N$ and the subprogram should return the corresponding cost. If there are constraints on the Z_i's, they can often be entered by augmenting the cost function.

Another method, which is used in this report, is to set a flag when any constraint is violated. The flag prevents acceptance of the nonfeasible parameter and in the next gradient calculation, augments the violating parameter's gradient.

$$D_j = \beta R_j \left[\frac{\Delta y_j}{|\Delta y|} + \sum_k \frac{\Delta F_k}{|\Delta F|}\right]$$

where $\Delta F_k/|\Delta F|$ is the normalized gradient of any violated constraint function with respect to Z_j. This method was proposed by Klingman and Himmelblau [4].

Now that the Partan technique has been described, some of its programming advantages are apparent. The program is relatively short, requiring less than one hundred instructions. The core storage requirements are minimal since only the two previous tries are retained.

Although the inexact determination of y and each $\bar{Z}_j$ degrades the convergence for quasi-quadratic cost functions (ideally 2N-1 steps), Harkin has demonstrated that the number of steps required is still proportional to the dimension N of $\bar{Z}$. Thus, Partan is superior to normal steepest ascent techniques.

A common non-quasi-quadratic cost function used to measure convergence performance is Rosenbrock's function

$$y = 100(Z_2 - Z_1^{\ 2})^2 + (1 - X_1)^2$$

with $\bar{Z}_o = (-1.2, 1)$ and $y_{opt} = 0$. Partan will converge to $y \leq 10^{-5}$ in less than 180 cost function evaluations (less than 30 steps). This is superior to the normal steepest ascent method (does not converge), sectioning method (no convergence), Spider method (>400 cost function evaluations) and Simplicial method (>1200 evaluations).

The Program

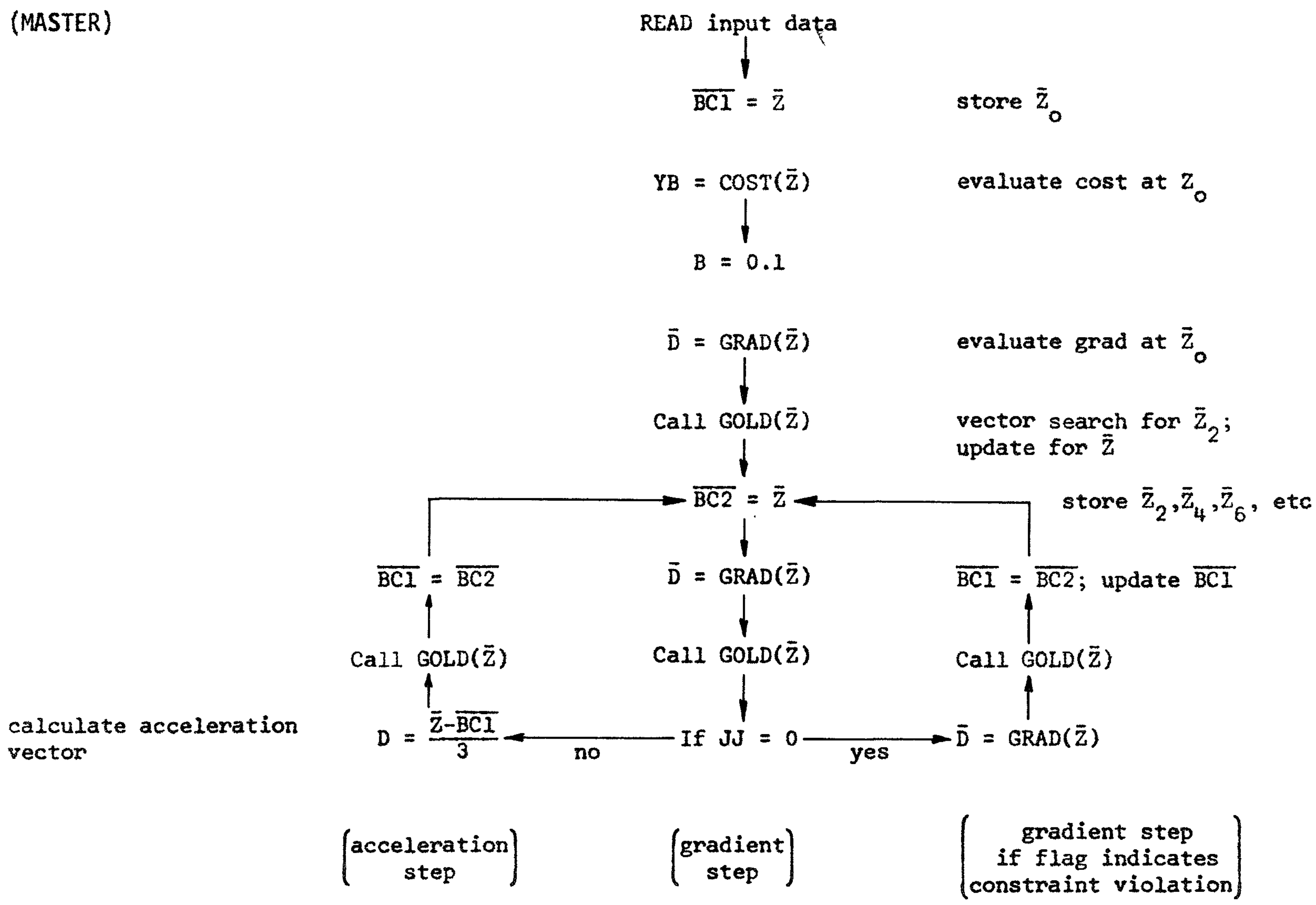
The input data is read into the program via the master Partan program. The first card contains search parameter information and the following N cards contain the starting value and range of each Z_i.

CARD 1	Col 1-10	- E	(termination criteria)	in E10.0
	Col 11-20	- A	(alpha-perturbation)	in E10.0
	Col 21-30	- N	(number of Z_i's)	in I10
	Col 31-40	- L	(no. of constraint eqs.)	in I10
CARD 2	Col 1-10	- Z_1	(initial value)	in E10.0
	Col 11-20	- R_1	(range)	in E10.0
CARD 3	Col 1-10	- Z_2	(initial value)	in E10.0
	Col 11-20	- R_2	(range)	in E10.0
CARD N				

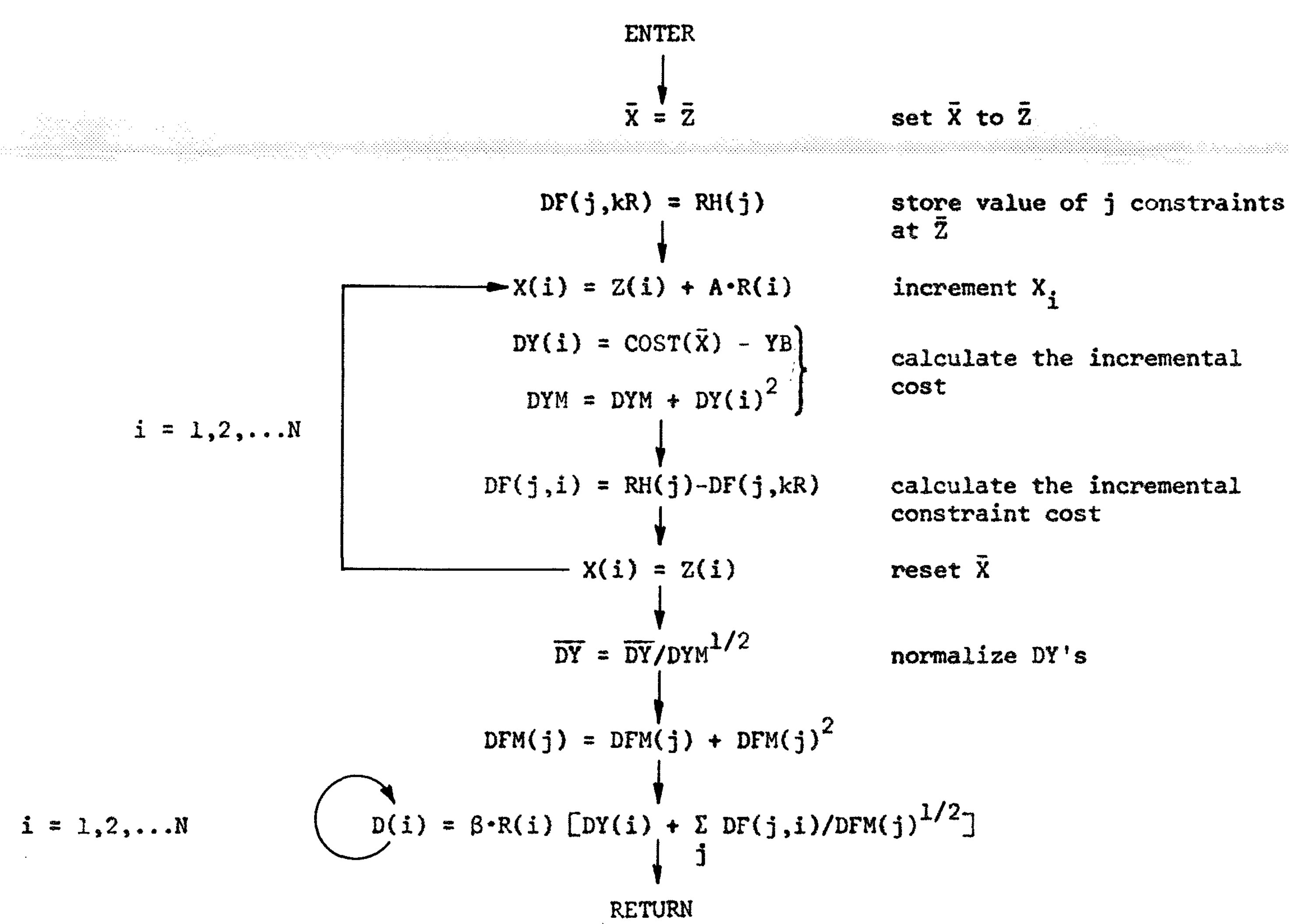
The flow charts appear on the following three pages.

The program output lists the number (N) of parameters being optimized, the perturbation constant (α) and the termination criteria (E). Following this is a listing of the search and its result. At each point ($\bar{Z}_o, \bar{Z}_2, \bar{Z}_3, \ldots, \bar{Z}_p$) the current step number (P) is given along with the total number of cost function evaluations. The values of each Z_i at that point are then listed, followed by the value of the cost function (y) there. An optional printout,that shows how progress was made during the optimization is a listing of the step size parameter (B).

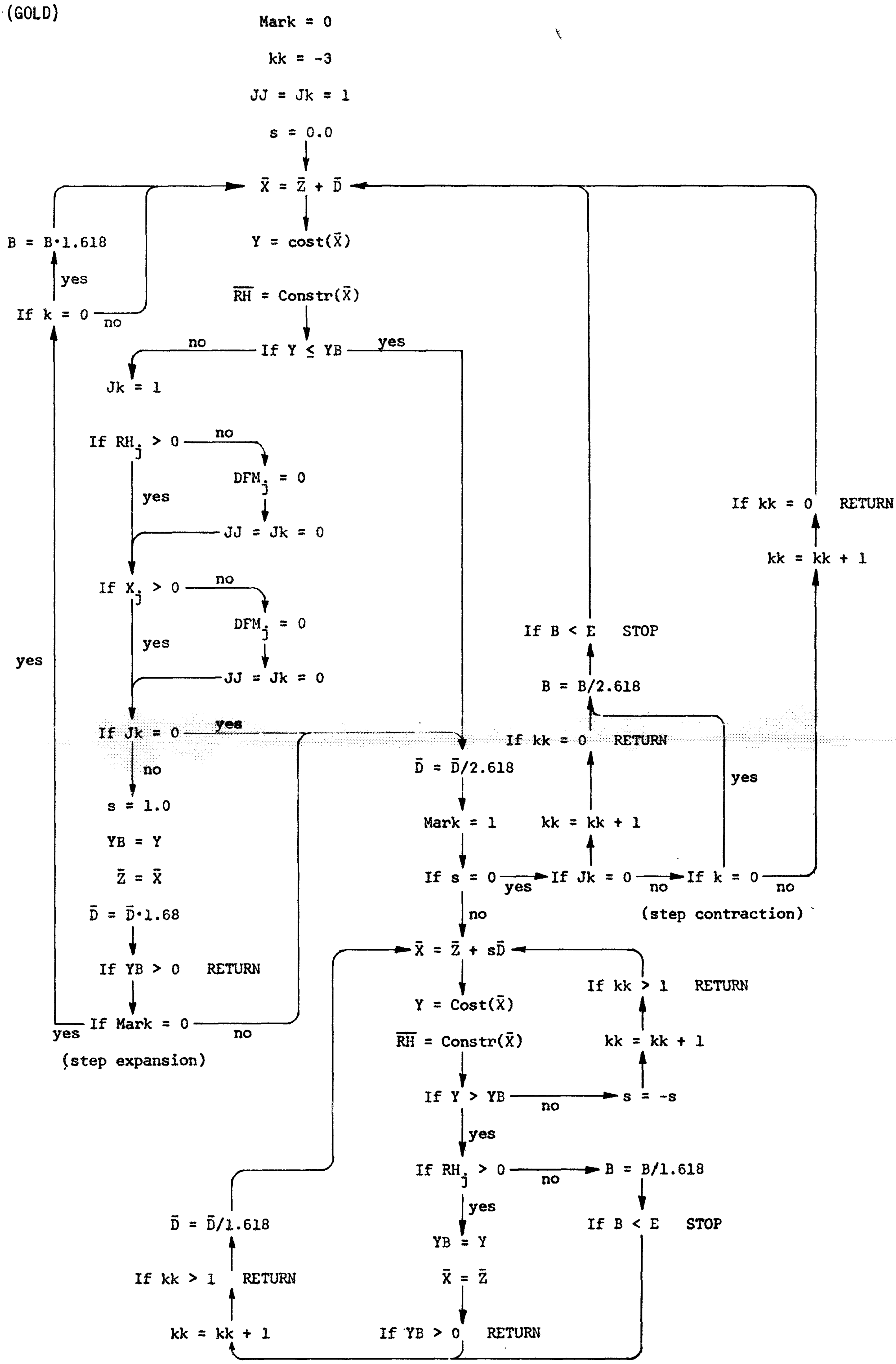
(MASTER)


READ input data
$\overline{BC1} = \bar{Z}$
store $\bar{Z}_o$
$YB = COST(\bar{Z})$
evaluate cost at Z_o
$B = 0.1$
$\bar{D} = GRAD(\bar{Z})$
evaluate grad at $\bar{Z}_o$
Call GOLD($\bar{Z}$)
vector search for $\bar{Z}_2$; update for $\bar{Z}$
$\overline{BC2} = \bar{Z}$
store $\bar{Z}_2, \bar{Z}_4, \bar{Z}_6$, etc.
$\overline{BC1} = \overline{BC2}$
$\bar{D} = GRAD(\bar{Z})$
$\overline{BC1} = \overline{BC2}$; update $\overline{BC1}$
Call GOLD($\bar{Z}$)
Call GOLD($\bar{Z}$)
Call GOLD($\bar{Z}$)
calculate acceleration vector
$D = \frac{\bar{Z}-\overline{BC1}}{3}$
no
If JJ = 0
yes
$\bar{D} = GRAD(\bar{Z})$
acceleration step
gradient step
gradient step if flag indicates constraint violation

(GRAD)


ENTER
$\bar{X} = \bar{Z}$
set $\bar{X}$ to $\bar{Z}$
DF(j,kR) = RH(j)
store value of j constraints at $\bar{Z}$
X(i) = Z(i) + A·R(i)
increment X_i
DY(i) = COST($\bar{X}$) - YB
DYM = DYM + DY(i)^2
calculate the incremental cost
i = 1,2,...N
DF(j,i) = RH(j)-DF(j,kR)
calculate the incremental constraint cost
X(i) = Z(i)
reset $\bar{X}$
$\overline{DY} = \overline{DY}/DYM^{1/2}$
normalize DY's
DFM(j) = DFM(j) + DFM(j)^2
i = 1,2,...N
$D(i) = \beta \cdot R(i) [DY(i) + \sum_j DF(j,i)/DFM(j)^{1/2}]$
RETURN

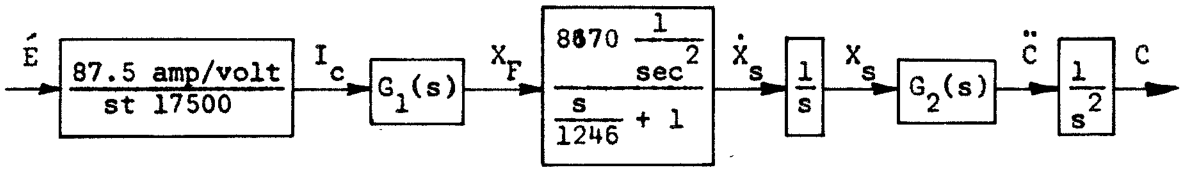
(GOLD)



(uniblock search)

Application to Fuel Valve Problem

The plant G(s) is shown below:



where

$$G_1(s) = \frac{\frac{1}{93}\, s^2\, (s+1256)\, \frac{in}{in\text{-}lb\text{-}sec}}{[(s+1970)^2 + 4030][(s+488)^2 + 1190^2]}$$

and

$$G_2(s) = \frac{11.995\text{x}10^{11}\, s\, \frac{1}{sec}}{(s+3342)^2 + 17820^2}$$

The desired response with 500 Htz bandwidth and 0.707 damping ratio is

$$\frac{C(s)}{R(s)} = 1/\left(\frac{s^2}{329\text{x}10^6} + \frac{67\text{x}10^3 s}{329\text{x}10^6} + 1\right)\left(\frac{s}{6\text{x}10^3} + 1\right)\left(\frac{s}{3.5\text{x}10^3} + 1\right)\cdots$$

$$\cdots\left(\frac{s^2}{36\text{x}10^6} + \frac{4.8\text{x}10^3 s}{36\text{x}10^6} + 1\right)\left(\frac{s^2}{12.3\text{x}10^6} + \frac{4\text{x}10^3}{12.3\text{x}10^6} + 1\right)$$

to be obtained using the following configuration (Fig. 2).

$$C(s) = G(s)\, E'(s)$$

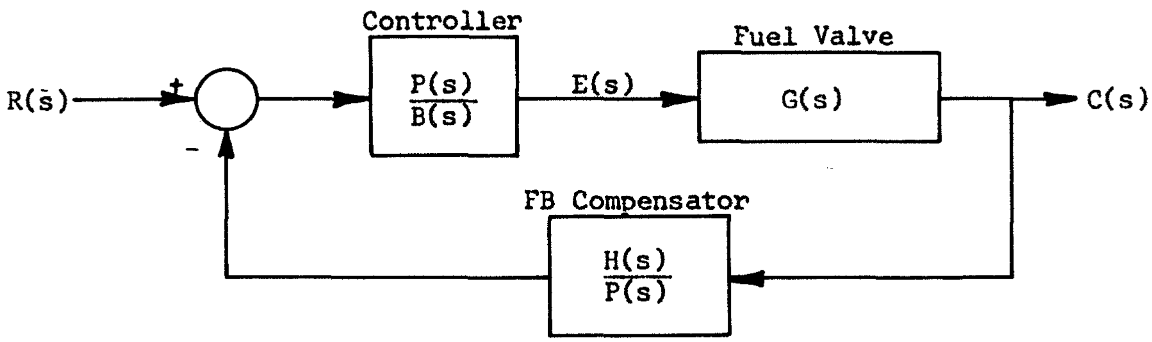


Fig. 2. Fuel Valve Block Diagram.

$$E'(s) = \frac{P(s)}{B(s)} E(s)$$

and

$$E(s) = R(s) - \frac{H(s)}{P(s)} C(s)$$

For a given G(s) and C(s)/R(s), H(s) and B(s) are related to P(s) by the transformations

$$H(s) = T_H[P(s)]$$

and

$$B(s) = T_B[P(s)]$$

The computer program is modified to pick P(s) such that both B(s) and P(s) have LHP roots and such that low frequency sensitivity is optimally small.

The coefficients of s which determine P(s) are constrained such that P(s) is stable.

$$P(s) = (s + P_1)(s^2 + P_2 s + P_3)(s^2 + P_4 s + P_5)(s^2 + P_6 s + P_7)$$

Thus, all $(P_1, P_2, \ldots, P_7)$ must be constrained positive. To insure C(s) is stable, the Routh-Hurwitz column coefficients are constrained positive. Low frequency sensitivity is improved by trying to force the polynomial B(s) to have a "free s."

Since Partan is constrained from entering a nonfeasible region (where a constraint is violated), the initial point must not violate any constraint. Experience has shown that an initial point cannot be selected such that both P(s) and B(s) are stable without some previous knowledge. Therefore, the Partan program has been modified so that the program starts with only those constraints that are not violated. Using each constraint which is violated as a cost function, each constraint is eventually satisfied. Then the program optimizes the desired cost function (low frequency sensitivity in this case).

The progress toward the solution of the fuel valve problem can be seen in the output listing. The listing shows that at the initial point $\bar{P} = (10^4, 10^4, 10^8, 10^4, 10^8, 10^4, 10^8)$ (selected arbitrarily) the Routhian array had the form

$(R_1 > 0)$	(-)	(-)	(-)
$(R_2 > 0)$	(-)	(-)	(-)
$(R_3 > 0)$	(-)	(-)	
$(R_4 > 0)$	(-)	(-)	
$(R_5 > 0)$	(-)		
$(R_6 > 0)$	(-)		
$(R_7 < 0)$			
(R_8 not computed)			

Since R_7 is the first negative value $(-2.123\text{x}10^{20})$ reached in the first column, the cost function y is set equal to R_7 and the Partan search is initiated. After one gradient calculation and the resulting vector search, the cost function (R_7) is made positive ($5.349\text{x}10^{19}$) at some point $\bar{P}'$ on the gradient.

The Routhian array is now checked for sign changes beyond R_7 and the last coefficient, R_8, is found negative ($-1461\text{x}10^{25}$). Starting at $\bar{Z}_1 = \bar{Z}'_o$, a single gradient search forces R_8, the new cost function, positive ($1.402\text{x}10^{23}$) at $\bar{P}''$. Thus, after twenty function evaluations, the Routh Table indicates that for the stable polynomial P(s) given by the coefficient $\bar{P}''$, the polynomial B(s) is also stable (LHP roots).

For good low frequency sensitivity, it is desirable that the zeroth power of s coefficient be zero. This coefficient is selected as the cost function, and its initial value at $\bar{P}''$ is $1.653\text{x}10^{28}$. After 13 acceleration and gradient steps involving 116 function evaluations, this coefficient is reduced 16 orders of magnitude to $1.002\text{x}10^{12}$. The polynomial P(s) is

$$(s + 12{,}310)(s^2 + 15.450s + 433)^3 = s^7 + 5.9\text{x}10^4 s^6 + 1.29\text{x}10^9 s^5$$

$$+ 1.25\text{x}10^{13} s^4 + 4.54\text{x}10^{16} s^3 + 3.82\text{x}10^{15} s^2 + 1.07\text{x}10^{14} s + 1.002\text{x}10^{12}$$

and the corresponding B(s) is

$$s^7 + 5.4\text{x}10^4 s^6 + 1.19\text{x}10^9 s^5 + 1.38\text{x}10^{13} s^4$$

$$+ 9.15\text{x}10^{16} s^3 + 3.53\text{x}10^{20} s^2 + 6.82\text{x}10^{23} s + 1.18\text{x}10^{27}$$

The result shows that the improvement of low frequency sensitivity in the output has been achieved by a controller with three poles at about 0.003 sec^{-1}, and the other poles at frequencies greater than 12,000 sec^{-1}. The low frequency gain will be on the order of 10^{15}.

The result also indicates how this search technique can be improved and how very high order problems of this type can be treated. Since the initial guess $\bar{P}$ was a polynomial with a neg-real root and a third order, complex pair of roots, the number of search parameters could actually have been reduced to three. This would have reduced computation time for each of the six gradient calculations by a factor of 3/7, since fewer perturbations are required. Also, if P(s) were given as 13th order instead of 7th order, the initial guess could have been

$$(s + 10{,}000)(s^2 + 10{,}000s + 10^8)^6$$

i.e., still only three search parameters. There is, however, no guarantee that a solution for such a P(s) exists. If the search failed to converge to a stable B(s), the initial guess could be generalized to

$$(s + 10{,}000)(s^2 + 10{,}000s + 10^8)^3(s^2 + 40{,}000s + 2\times10^8)^3$$

and the search conducted with five parameters, etc.

Conclusion

Partan is an efficient program, which can be modified to handle complicated, constrained cost functions. It is especially suited to problems where the cost is explicit but the gradients must be computed by perturbation.

References

[1] Buehler, R. J., Shah, B. V., and Kempthorne, O., "Methods of Parallel Tangents," Chemical Engineering Program Symposium, Serial No. 50, 1964.

[2] Wilde, D. J., and Beightler, C. S., *Foundations of Optimization*, Prentice Hall, Chapter 2, 1967.

[3] Harkins, A., "The Use of Parallel Tangents in Optimization," Chemical Engineering Program Symposium, Serial No. 50, 1964.

```
      PROGRAM    MASTER (INPUT, OUTPUT)
      COMMON Z(10),D(10),N,B,M,YB,L,DFM(20),JJ    ,KR,RH(9)
      DIMENSION R(10),BC1(10),BC2(10)
      READ    1,E,A,N,L,(Z(I),R(I),I=1,N)
 1    FORMAT (2E10.0,2I10/(2E10.0))
      PRINT    2,N,A,E
 2    FORMAT(*1PARTAN SEARCH IN*I2* VARIABLES*//* WITH RANGE FRACTION A
     1= *E7.0* AND TERMINATION CRITERIA E =*E7.0)
      N1 = N + 1
      M = 1
      KR = 1
 95   J = 0
      DO    3 I = 1,L
 3    DFM(I) = 1.0
      DO 4 I = 1,N
 4    BC1(I) = Z(I)
 90   KR = KR + 1
      IF (KR.GT.N1) STOP
      CALL COSTS(KR,Z,YB,RH)
      IF (YB.GT. 0.0) GO TO 90
      PRINT    6,J,M ,(Z(I),I = 1,N),YB
      B = 0.1
      CALL GRAD(A,R)
      CALL GOLD(E,1)
      J = 2
 5    PRINT    6,J,M ,(Z(I),I = 1,N),YB
 6    FORMAT (*0P*I3*  EVALUATION =*I4/(10E13.3))
      IF (YB.GT. 0.0) GO TO 95
      DO 7 I = 1,N
 7    BC2(I) = Z(I)
      J = J + 1
      CALL GRAD(A,R)
      CALL GOLD(E,1)
      PRINT    6,J,M ,(Z(I),I = 1,N),YB
      IF (YB.GT. 0.0) GO TO 95
      IF (JJ.EQ.0) GO TO 9
      DO 8 I = 1,N
 8    D(I) = (Z(I) - BC1(I))/3.0
      CALL GOLD(E,0)
      IF (YB.GT. 0.0) GO TO 95
      GO TO 10
 9    CALL GRAD(A,R)
      CALL GOLD(E,1)
      IF (YB.GT. 0.0) GO TO 95
 10   DO 11 I = 1,N
 11   BC1(I) = BC2(I)
      J = J + 1
      IF(J.LT. 25) GO TO 5
      STOP
      END
```

```
      IF (B.LT.E) GO TO 17
      GO TO 15
 12   CONTINUE
      DO 125 J = 1,N
      IF (X(J).GT.0.0) GO TO 125
      B = B/1.618034
      IF (B.LT.E) GO TO 17
      GO TO 15
 125  CONTINUE
      YB = Y
      DO 13 I = 1,N
 13   Z(I) = X(I)
      IF (YB .GT. 0.0) RETURN
      GO TO 15
 14   S = - S
 15   KK = KK + 1
      IF(KK.GE.1) RETURN
      DO 16 I = 1,N
 16   D(I) = D(I)/1.618034
      GO TO 10
 17   PRINT 18,KR,(Z(I),I=1,N),YB
 18   FORMAT (* EVALUATION =*I4/(10E13.3))
      STOP
      END
```

```
      SUBROUTINE GRAD(A,R)
      COMMON Z(10),D(10),N,B,M,YB,L,DFM(20),JJ    ,KR,RH(9)
      DIMENSION X(10),DY(10),R(10),DF(10,10)
      DO 1 I = 1,N
1     X(I) = Z(I)
      DO 2 J = 1,KR
2     DF(J,KR) = RH(J)
      DYM = 0.0
      I = 1
3     X(I) = Z(I) + A*R(I)
      CALL COSTS(KR,X,Y,RH)
      DY(I) = Y - YB
      M = M + 1
      DYM = DYM + DY(I)**2
      DO 4 J = 1,KR
      IF (DFM(J).NE.0.0) GO TO 4
      DF(J,I) = RH(J) - DF(J,KR)
4     CONTINUE
      X(I) = Z(I)
      I = I + 1
      IF(I.LE.N) GO TO 3
      DYM = SQRT(DYM)
      DO 5 I = 1,N
5     D(I) = DY(I)/DYM
      DO 8 J = 1,KR
      IF (DFM(J).NE.0.0) GO TO 8
      DO 6 I = 1,N
6     DFM(J) = DFM(J) + DF(J,I)**2
      DFM(J) = SQRT(DFM(J))
      DO 7 I = 1,N
7     D(I) = D(I) + DF(J,I)/DFM(J)
8     CONTINUE
      DO 75 J = 1,N
      IF (DFM(J+7).EQ.0.0) D(J) = D(J) + 1.0
75    CONTINUE
      DO 9 I = 1,N
9     D(I) = B*R(I)*D(I)
      RETURN
      END
```

```
      SUBROUTINE GOLD(E,K)
      COMMON Z(10),D(10),N,B,M,YB,L,DFM(20),JJ    ,KR,RH(9)
      DIMENSION X(10)
      MARK = 0
      KK = -3
      JJ = JK = 1
      S = 0.0
1     DO 2 I = 1,N
2     X(I) = Z(I) + D(I)
      PRINT 102,B
102   FORMAT (20X*B   *E13.5)
      CALL COSTS(KR,X,Y,RH)
      M = M + 1
      IF(Y.LE.YB) GO TO 5
      JK = 1
      DO 3 J = 3,KR
      IF (RH(J).GT.0.0) GO TO 3
      DFM(J) = 0.0
      JJ = JK = 0
3     CONTINUE
      DO 35 J = 1,N
      IF(X(J).GT.0.0) GO TO 35
      DFM(J+7) = 0.0
      JJ = JK = 0
35    CONTINUE
      IF (JK.EQ.0) GO TO 5
      S = 1.0
      YB = Y
      DO 4 I = 1,N
      Z(I) = X(I)
4     D(I) = D(I)*1.618034
      IF (YB .GT. 0.0) RETURN
      IF(MARK.NE.0) GO TO 5
      IF(K.NE.0) B = B*1.618034
      GO TO 1
5     DO 6 I = 1,N
6     D(I) = D(I)/2.618034
      MARK = 1
      IF(S.NE.0.0) GO TO 10
      IF (JK.EQ.0) GO TO 7
      IF(K.EQ.0)  8,9
7     KK = KK + 1
      IF (KK.EQ.0) RETURN
8     B = B/2.618034
      IF (B.LT.E) GO TO 17
      GO TO 1
9     KK = KK + 1
      IF (KK.EQ.0) RETURN
      GO TO 1
10    DO 11 I = 1,N
11    X(I) = Z(I) + S*D(I)
      CALL COSTS(KR,X,Y,RH)
      M = M + 1
      IF(Y.LT.YB) GO TO 14
      DO 12 J = 3,KR
      IF (RH(J).GT.0.0) GO TO 12
      B = B/1.618034
```

```
      SUBROUTINE COSTS(KR,Z,Y,RH)
      DIMENSION Z(7),B(8),T(8),RH(9),R(9,9)
      DATA T/1.0,-4.130E3,1.536E8,-2.422E12,4.221E16,-7.395E20
     1,1.301E25,-2.283E29/,R/81*0.0/
      B(1) = 1.0
      B(2) = Z(1)
      DO 1 K = 3,8
1     B(K) = 0.0
      DO 3 J = 1,3
      J2 = 2*J
      DO 2 I = 1,J2
      K = J2 + 3 - I
2     B(K) = B(K) +B(K-1)*Z(J2) + B(K-2)*Z(J2+1)
3     B(2) = B(2) + Z(J2)
      RR = -B(8)
      DO 45 I = 1,8
      J = 9- I
      BB = 0.0
      DO 4 K = 1,J
      M = J + 1 - K
4     BB = BB + B(K)*T(M)
45    B(J) = BB
      J = - 1
      DO 5 I = 1,4
      J = J + 2
      R(1,I) = B(J)
5     R(2,I) = B(J+1)
      DO 7 J = 2,KR
      LIM = KR + 2 - J
      DO 6 I = 2,LIM
6     R(J+1,I-1) = R(J-1,I) - (R(J-1,1)*R(J,I))/R(J,1)
      IF (R(J+1,1).EQ.0.0) R(J+1,1) = .00001
7     CONTINUE
      DO 8 I = 1,KR
8     RH(I) = R(I,1)
      R(9,1) = RR
      Y = R(KR+1,1)
      RETURN
      END
```

PATTERN SEARCH

Introduction

This report describes the use of an optimizing search procedure for the design of a control system where some of the state variables of the system are unavailable. This design procedure is useful in achieving an approximate closed-loop transfer function rather than obtaining an exact closed-loop transfer function. The advantage of this procedure is that the complexity of the compensation is reduced over the state variable design. This procedure is applied to the design of an inlet control system which minimizes the response of the shock wave position to pressure disturbances at the compressor. The search procedure described starts with an initial choice of parameters and makes small changes in these parameters until an improvement is obtained. Then larger steps are made until no further improvement is obtained. When the larger steps are not fruitful, successively smaller steps are taken. At this point the search procedure has found a local minimum.

The advantage of this type of procedure over a more classical design procedure is its extreme flexibility. The design may include constraints on the parameters, a variety of objectives, and a variety of parameters in the compensator. The performance objectives may include either time domain or frequency domain parameters. Initial design considerations may be used to select an initial choice of parameters. This procedure is therefore more adaptable to practical applications where all the state variables are unavailable or other limitations which make other design procedures impractical.

Pattern Search

The pattern search is based on the following philosophy:

1. If an improvement is made in a given direction, continue to move in that direction.
2. If an improvement is made with a small change in parameters, try a bigger change.

The pattern search has two modes of operation depending on the number of previous successes or failures:

1. Mode 1. Successful Move.

   If a successful move has been made, then try another larger move in the same direction. An additional increment to the left or right is added in if the previous successful move was made by altering the direction of the move. If no success is made, the next mode is used.

2. Mode 2. Local Search.

   Small moves are made in the same direction as the last success and at right angles to this direction. If one or more successes are made after these moves, then the system returns to Mode 1. If no improvement is made, a smaller step is tried. If more than ten reductions in step size are made, the search is terminated on the presumption that a local minimum has been found.

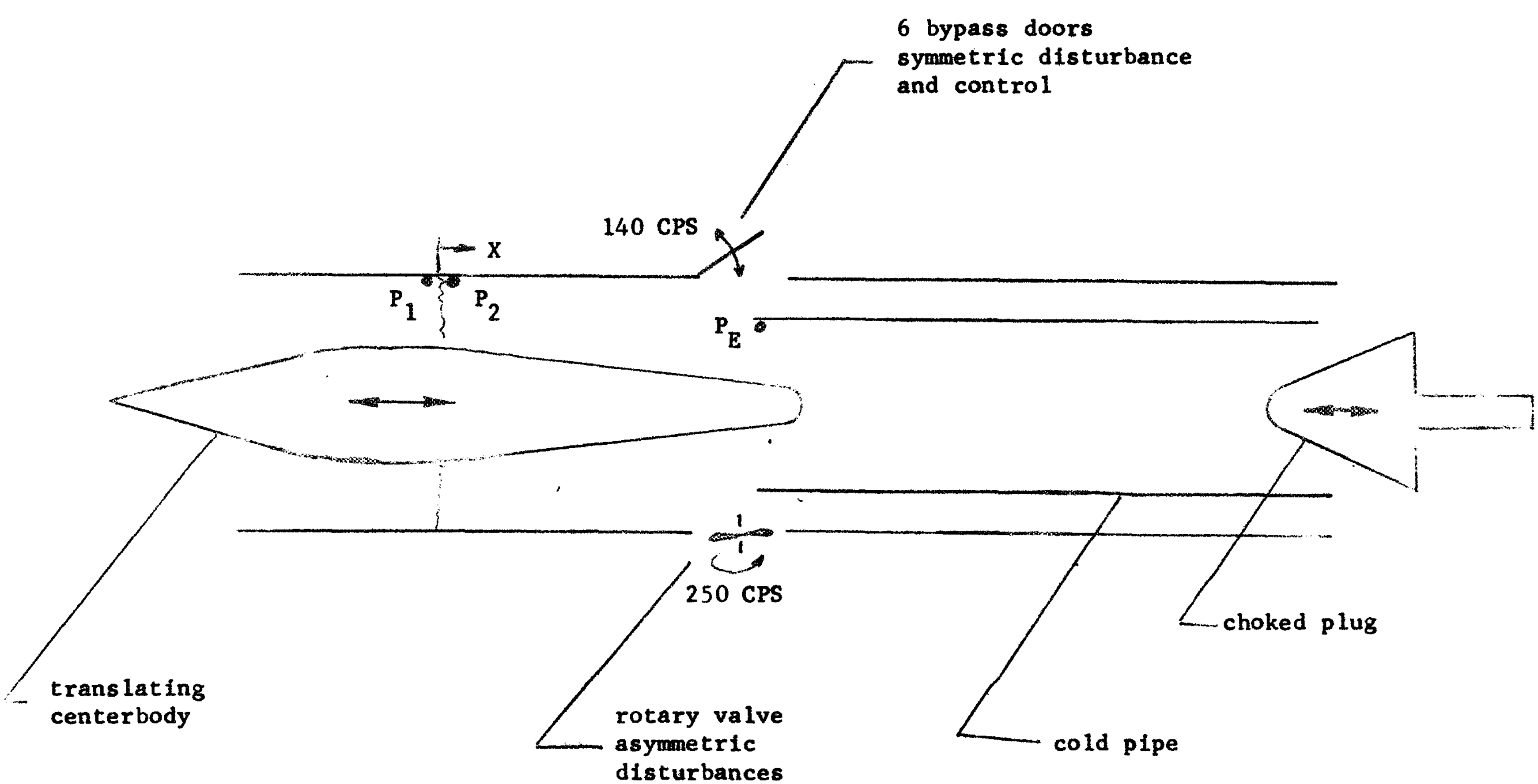


Fig. 1. 40-60 Inlet Diagram.

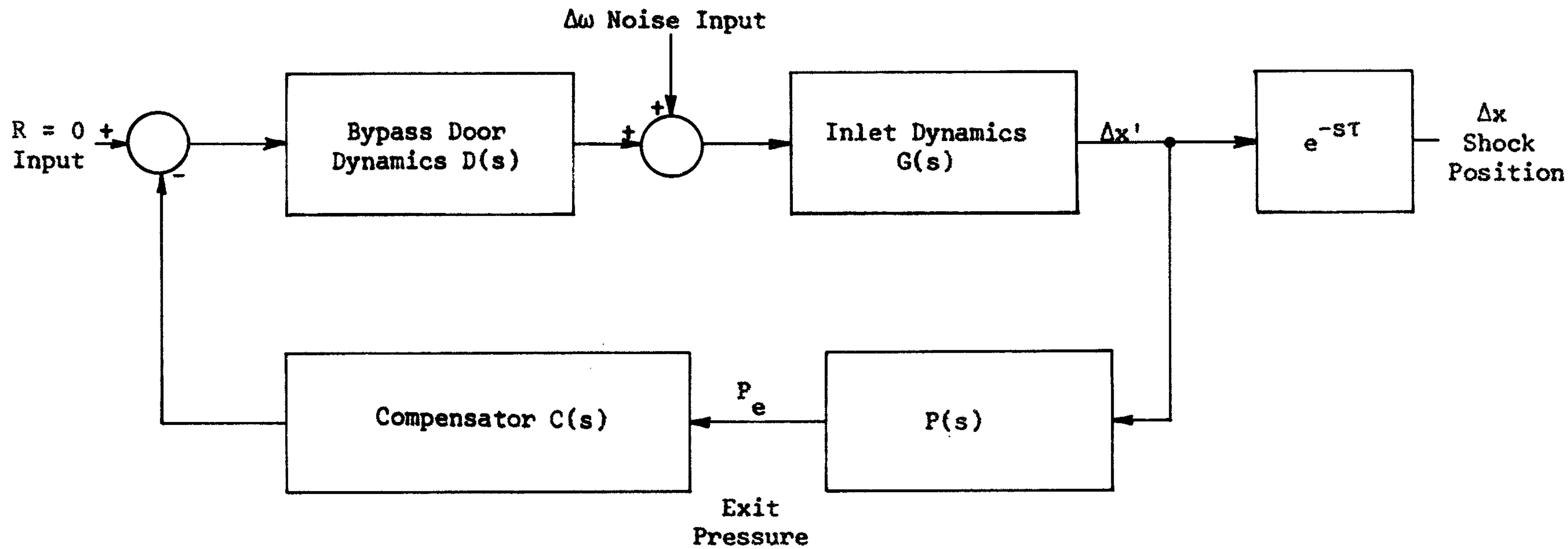


Fig. 2. Inlet Control System.

The following application is to demonstrate the use of this procedure for a jet engine control system. Details of the computer program are given in the last section.

The 40-60 Inlet Control Problem

The outline of the system is shown in Fig. 1, and the 40-60 inlet control system is shown in block diagram form in Fig. 2. It is desired to minimize the response of the shock wave position x to disturbances at the compressor side of the inlet. In the block diagram the noise input occurs in front of G(s) which describes the dynamics of the inlet. The exit pressure P_e is related to the undelayed shock position by the transfer function P(s). The control of the shock position is affected by the shunting of air through six bypass doors whose dynamics are D(s). The response of the system to disturbances when no control is present is shown in Fig. 3. The objective of the system is to provide a compensator whose dynamics are C(s) that will result in a lower response than G(s) shown in Fig. 3.

$$G(s) = \frac{(388.12)(s+80.3\pm j172.20)(s+173.16\pm j332.63)(s+75.71\pm j575.04)}{(s+46)(s+44.58\pm j281.49)(s+84.22\pm j477.63)(s+130.24\pm j738.61)}$$

$$\frac{(s+82.8\pm j946.38)}{(s+191.02\pm j1083.29)}$$

$$P(s) = \frac{s+1010}{1010}$$

$$D(s) = \frac{1.3073\times10^{20}}{s(s+2000)(s+318.5\pm j1899)(s^3+3890.38s^2+2.1038\times10^7 s+1.76187\times10^{10})}$$

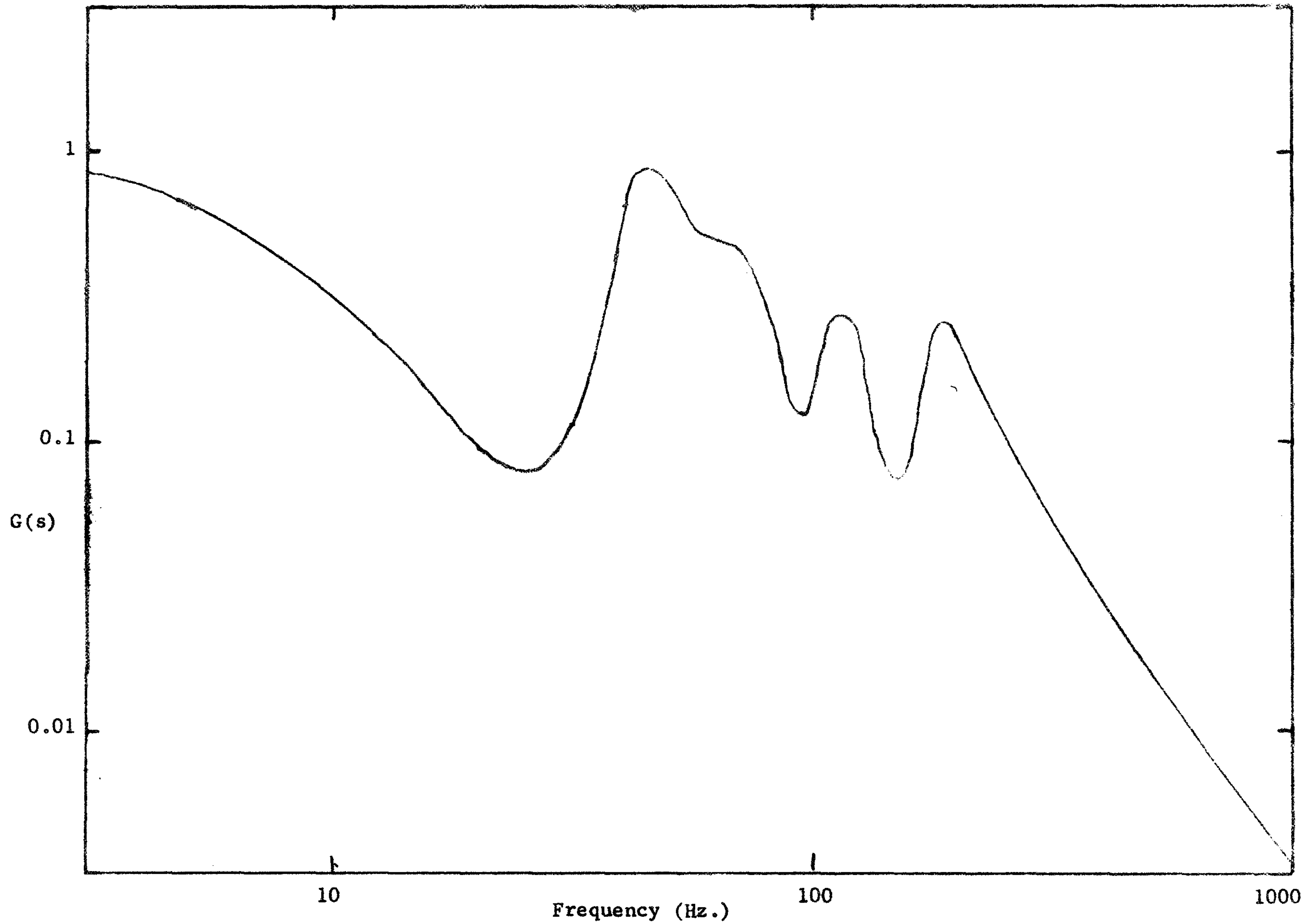


Fig. 3. Inlet Dynamics, G(s).

Performance Specification

A transfer function is specified by the user which determines the desired response of the system. The compensator C(s) which has two poles and two zeros is selected to minimize the mean square difference between the desired transfer function and the actual closed-loop transfer function over all frequencies. The actual transfer function should be stable. The parameters which specify the compensator are constrained so that the system is stable. The subroutine PLANT is used to construct the fixed part of the actual transfer function. The subroutine ERR constructs the difference between the actual transfer function and the desired transfer function for any particular gain, pole position, and zero position of C(s). The subroutine INTSQ evaluates the integral of the magnitude squared of the error over all frequencies. The details of these programs are included in the end of this section.

Computer Results

The desired transfer function, TD(s), was selected to be

$$TD(s) = \frac{388.12s}{(s+30)(s+388.12)}$$

The actual transfer function T(s) between the pressure disturbance and the undelayed shock position is given by

$$T(s) = \frac{G(s)}{1+G(s)P(s)C(s)D(s)}$$

If the numerator of G(s) is written GN and the denominator is written GD, and if the poles of C(s) are selected to be very large, then C(s) is approximately given by the numerator polynomial, C. The doors are given by the ratio DN/DD and the pressure transducer by P. The actual transfer function becomes

$$T(s) = \frac{GN \cdot DD}{D \cdot GD + DN \cdot P \cdot CN \cdot DN}$$

The numerator TN is independent of the choice of the compensator. It is calculated by the subroutine PLANT. Similarly, the following products are calculated in the same subroutine:

$$GP(s) = GN \cdot P \cdot DN$$

$$GD(s) = D \cdot GD$$

Then whenever a new value for C(s) is chosen where

$$C(s) = CN = \frac{X(1)}{X(3)} [X(3) + X(2)s + s^2],$$

the actual transfer function becomes

$$T(s) = \frac{TN}{GD' + GP \cdot CN}.$$

The procedure is carried out in the subroutine ERR. The error is formed from the integral of

$$TD(s) - T(s) = \frac{DN(s)}{DD(s)} - \frac{TN(s)}{TD(s)} = \frac{DN \cdot TD - DD \cdot TN}{DD \cdot TD}$$

The subroutine INTSQ is used to calculate the integral of the squared magnitude of the above polynomial. If the system is unstable, it is detected by the subroutine ROUTH and a large value of the objective function is returned.

The results of the computer program are the compensator whose parameters are

$$C(s) = 1414\,\frac{(s^2 + 0.6 \cdot 1578s + 1578^2)}{1578^2}$$

This compensator yields the closed-loop transfer function shown in Fig. 4. The response is significantly better than the open-loop response at frequencies below 100 H_z but approximately 10 db worse in the unity gain crossover region. The problem is more evident from a study of this result. A large feedback is needed at frequencies below 100 Hz. However, at frequencies above 100 Hz, the doors have a rapidly decreasing response which creates a very large phase shift at the point where the loop gain is unity which results in an unstable system. Either the phase shift must be decreased in magnitude or the gain crossover must be at a lower frequency. If the feedback gain is reduced, the system will respond just as an open-loop system at frequencies above the unity gain point which is not entirely satisfactory. By adding more zeros in the region given by C(s), a better response is possible, as shown in Fig. 5.

The numerator of compensator associated with Fig. 5 is:

$$C(s) = \frac{8{,}010\,(s^2 + 0.8 \cdot 1987s + 1987^2)^2(s + 3964)}{3964(1987)^4}$$

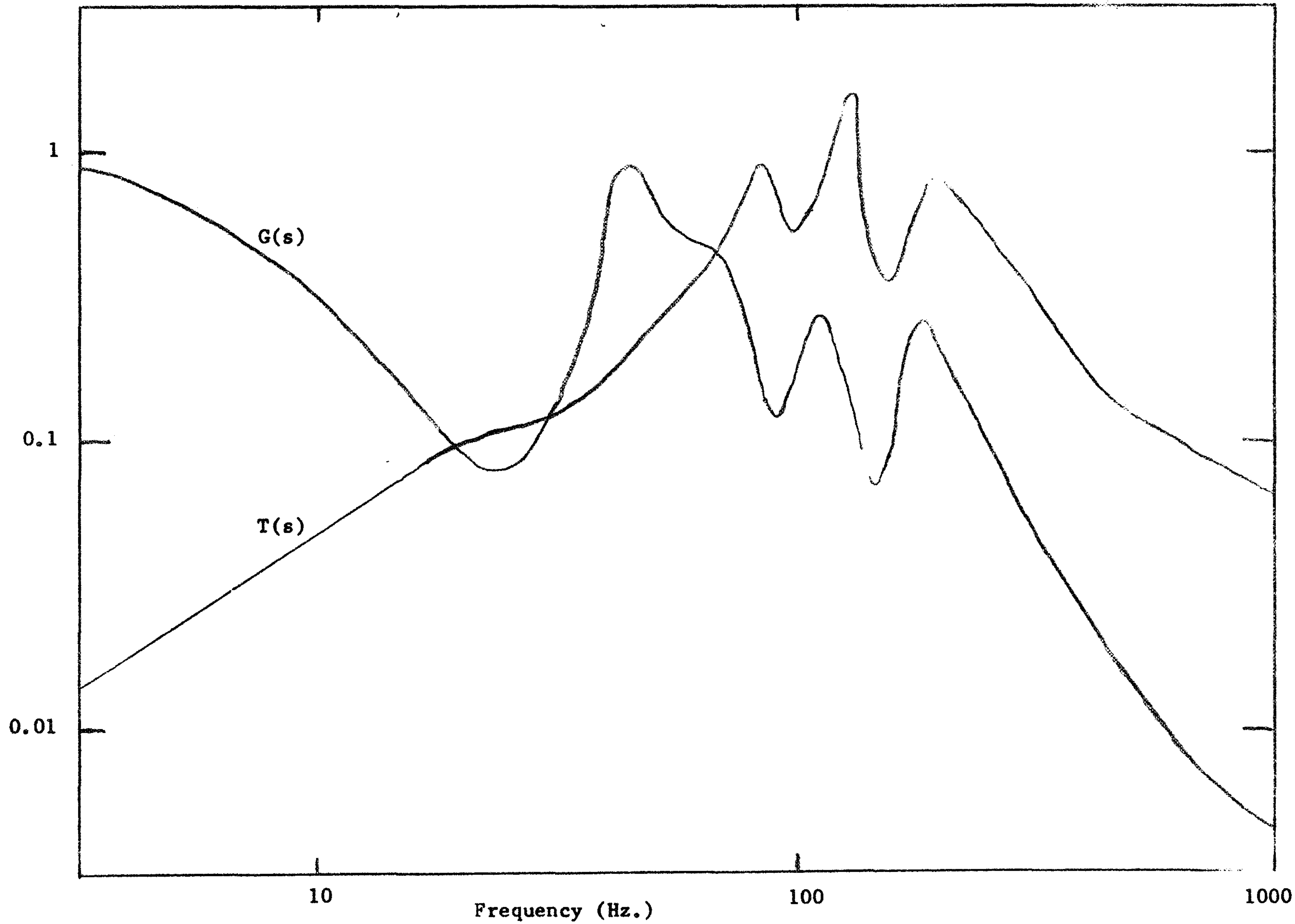


Fig. 4. Closed-Loop Response with Two Zero Compensator.

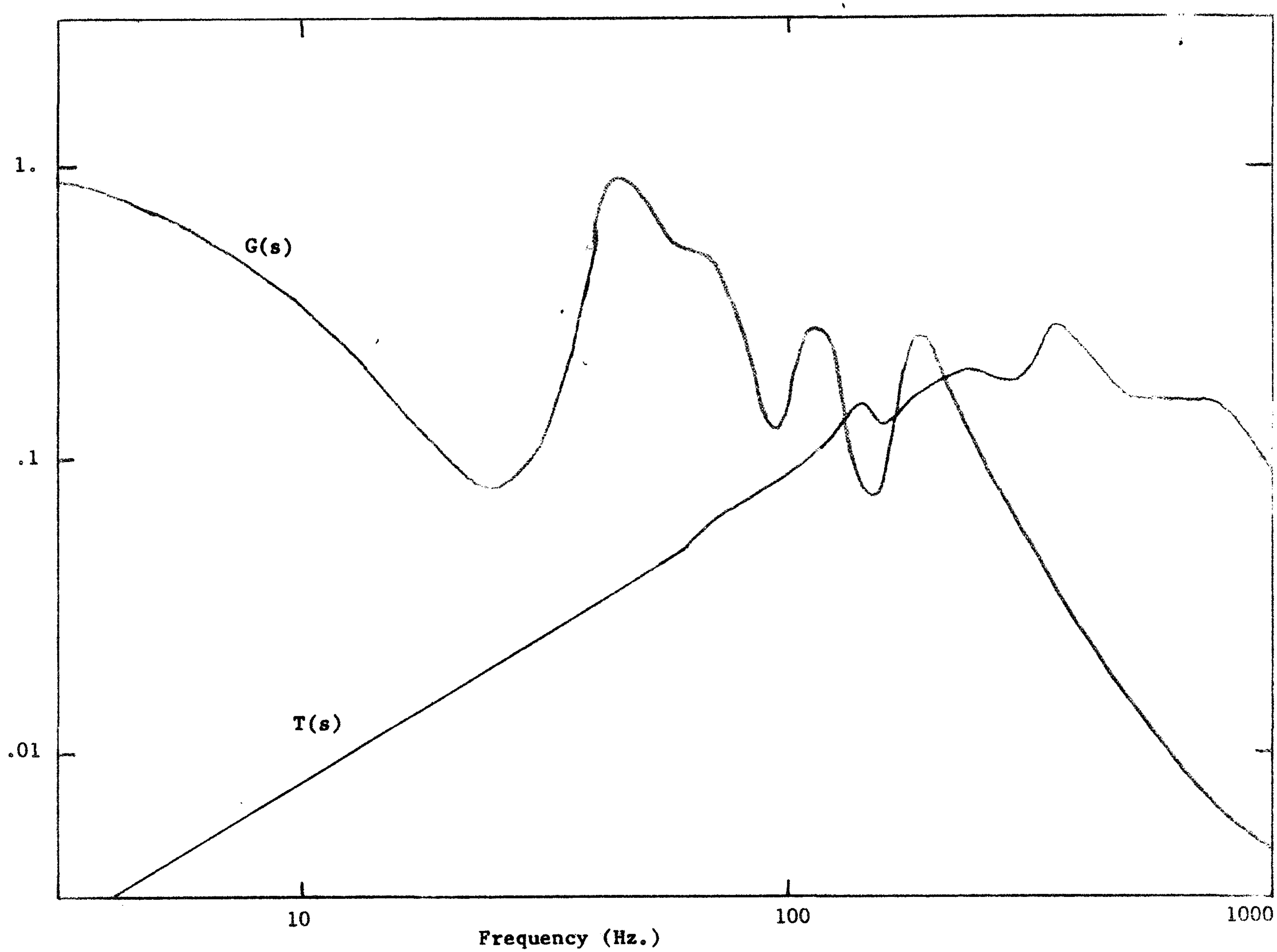


Fig. 5. Closed-Loop Response with Five Zero Compensator.

By adding five zeros, the response is always below -10 db and is smaller than -20 db for all frequencies below 100 Hz.

Computer Program

The relations between the subroutines of this program are shown in Fig. 6. The transfer functions associated with Fig. 2 which are independent of C(s) are computed in the subroutine PLANT for use in calculating T(s). The initial values of the parameters are used to provide a starting point for the pattern search. When a new set of parameters are selected, the subroutine ERR uses these new values to calculate the transfer function T(s). This transfer function is subtracted from the desired transfer function D(s) and the resulting error function E(s) is squared and integrated by the subroutine INTSQ. INTSQ uses the subroutine POLYSQ to calculate the square of the numerator of E(s). If the resulting integral Y is bigger, a local search in the vicinity of the last success is made. Successively smaller steps are taken until no further improvement is made.

A listing of the complete program follows.

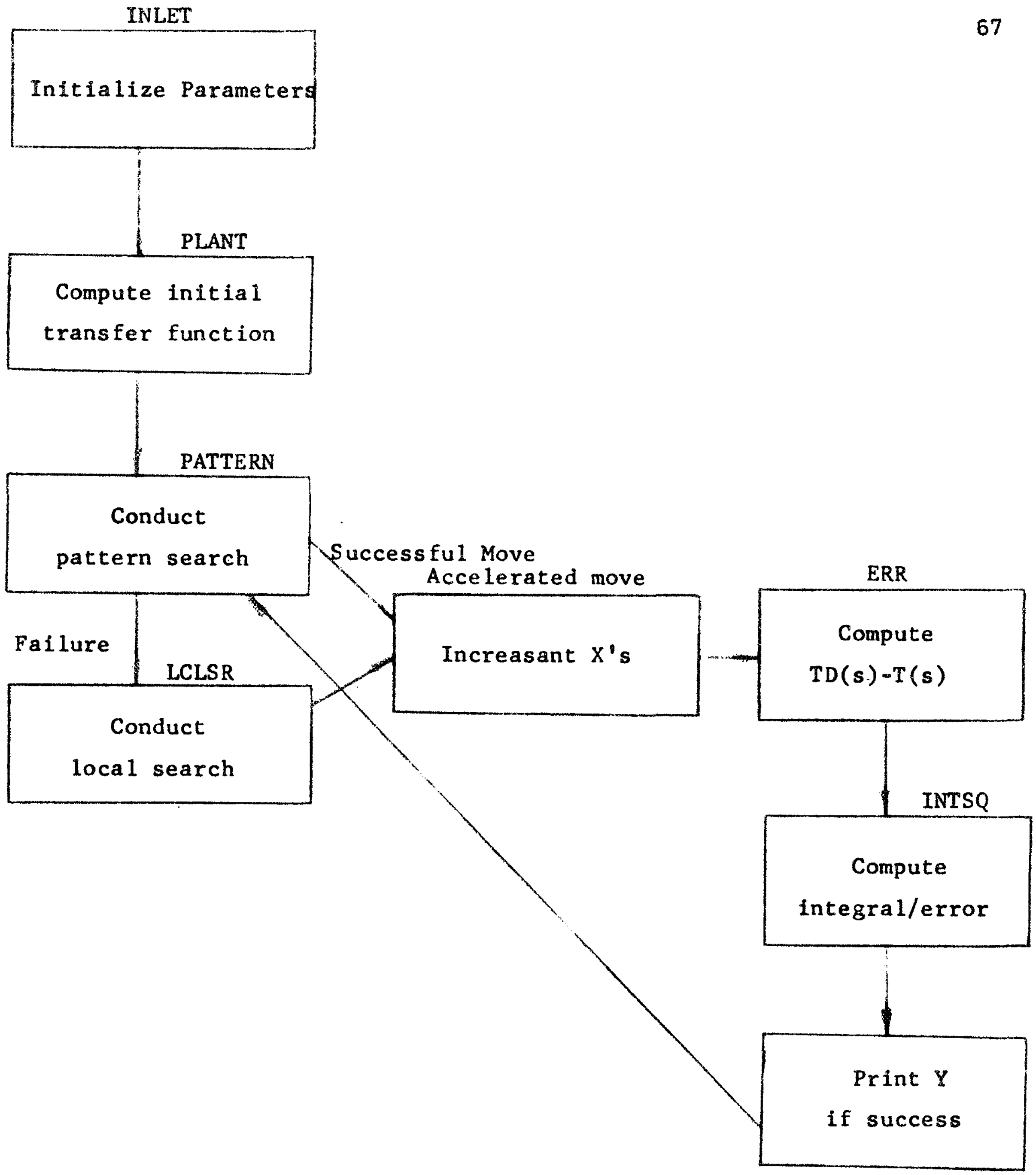


Fig. 6. General Flow Diagram of Pattern Search Program.

```
                    PROGRAM INLET(INPUT,OUTPUT)
000003               COMMON GD(20),GP(20),TN(20),DN(10),DD(10)
000003              DIMENSION BASE(3),HEAD(3)
000003              CALL PLANT
000004              BASE(1)=1414.
000005              BASE(2)=947.
000007               HEAD(1)=1000.
000010              HEAD(2)=1095.
000012              CALL PATERN (2,BASE,HEAD)
000015              STOP
000017              END
```

```
                  SUBROUTINE PLANT
000002             COMMON GD(20),GP(20),TN(20),DN(10),DD(10)
000002            DIMENSION GN1(10),GN2(10),GN(10),GD1(10),GD2(10),P(10),
                 1D1(10),D2(10),D(10)
            C     NUMERATOR OF PRESSURE TRANSDUCER
000002            DN=1.3073E20
000003            P(1)=DN
000005            P(2)=DN/1010.
            C     NUMERATOR OF INLET DYNAMICS
000006            GN1(1)=(172.2**2+80.3**2)*388.12
000012            GN1(2)=80.3*2.*388.12
000013            GN1(3)=388.12
000015            GN2(1)=(173.16**2+332.63**2)
000020            GN2(2)=173.16*2.
000021            GN2(3)=1.
000023            CALL PLYMLT(GN1,3,GN2,3,GN,0)
000027            GN1(1)=75.71**2+575.04**2
000032            GN1(2)=75.71*2.
000033            GN1(3)=1.
000035            CALL PLYMLT(GN1,3,GN,5,GN2,0)
000041            GN1(1)=82.8**2+946.38**2
000044            GN1(2)=82.8*2.
000045            GN1(3)=1.
000047            CALL PLYMLT (GN1,3,GN2,7,GN,0)
            C     DENOMINATOR OF INLET DYNAMICS
000053            GD1(1)=46.
000054            GD1(2)=1.
000056            GD(1)=44.58**2+281.14**2
000061             GD(2)=44.58*2.
000062            GD(3)=1.
000064            CALL PLYMLT(GD1,2,GD,3,GD2,0)
000070            GD1(1)=84.22**2+477.63**2
000073            GD1(2)=84.22*2.
000074            GD1(3)=1.
000076            CALL PLYMLT(GD1,3,GD2,4,GD,0)
000102            GD1(1)=130.24**2+738.61**2
000105            GD1(2)=130.24*2.
000106            GD1(3)=1.
000110            CALL PLYMLT(GD1,3,GD,6,GD2,0 )
000114            GD (1)=191.02**2+1083.29**2
000117            GD (2)=191.02*2.
000120            GD (3)=1.
000122            CALL PLYMLT(GD ,3,GD2,8,GD1,0)
            C     DENOMINATOR OF INLET DOORS
000126            D1(1)=0.
000127            D1(2)=2000.
000130            D1(3)=1.
000132            D(1)=318.5**2+1899.**2
000135            D(2)=318.5*2.
000136            D(3)=1.
000140            CALL PLYMLT (D1,3,D,3,D2,0)
000144            D1(1)=1.76187E10
000145            D1(2)=2.1038E7
000147            D1(3)=3890.38
000150            D1(4)=1.
000152            CALL PLYMLT(D1,4,D2,5,D,0)
            C     DEFINE NUMERATOR OF CLOSED LOOP TRANSFER FUNCTION
```

```
000155          CALL PLYMLT(GN,9,D,8,TN,0)
000162          CALL PLYMLT (GD1,10,D,8,GD,0)
000166          CALL PLYMLT(GN,9,P,2,GP,0)
000172          DO 10 I=11,15
000174       10 GP(I)=0.
000200           DN(1)=0.
000200           DN(2)=388.12
000202           DD(1)=-1164.36
000203           DD(2)=-418.12
000205           DD(3)=-1.
000207          PRINT 200,GD,GP,TN,DN,DD
000224          RETURN
000225      200 FORMAT (1X,10E13.5)
000225          END
```

```
              SUBROUTINE  PLYMLT (A,L,B,M,C,N)
         C
         C        MULTIPLY ONE POLYNOMIAL BY ANOTHER
         C
         C   DEFINITION OF SYMBOLS IN ARGUMENT LIST
         C        A(I),  MULTIPLICAND COEFFICIENTS IN THEORDER A(I) *S**(I-1)
         C        L.     NUMBER OF COEFFICIENTS OF A
         C        B(I),  MULTIPLIER COEFFICIENTS IN THE ORDER B(I)*S**(I-1)
         C        M.     NUMBER OF COEFFICIENTS OF B
         C        C(I),  PRODUCT COEFFICIENTS IN THE ORDER C(I)*S**(I-1)
         C        N,     NUMBER OF COEFFICIENTS OF C
         C
         C   REMARKS
         C        IF N=0, C(I) SET TO ZERO AND PRODUCT FORMED.  OTHERWISE THE PRODU
         C        AND SUM NEWC= OLD C + A*B IS FORMED.
         C
000011            DIMENSION A(10), B(10), C(20)
000011            LPM=L+M-1
000012            IF (N)  10,10,12
000014   10       DO 11  J=1,LPM
000016   11       C(J)=0.0
000022   12        DO 13 J=1,LPM
000024            MAX=MAX0(J+1-M,1)
000030            MIN=MIN0(L,J)
000033            DO 13  I=MAX,MIN
000035   13       C(J)=A(I)*B(J+1-I) + C(J)
000047            RETURN
000047            END
```

```
             SUBROUTINE PATERN (N,BASE,HEAD)
000006       DIMENSION BASE(10),HEAD(10),TEMP(10),CURVE(10)
000006       KPRNT=1
000006       ICUT=0
000007       KFCNS=1
000010       MAXKFN=100
000012       PRINT 200
000015       CALL ERR(HEAD,YH)
000021       GO TO 20
          C  ACCELERATED MOVE
000024    10 MODE=1
000025       ICUT=0
000026     9 DO 11  I=1,N
000030    11 TEMP(I)=2.* HEAD(I)-BASE(I)+CURVE(I)
000040       CALL ERR(TEMP,YT)
000042       KFCNS=KFCNS+1
000044       IF(KPRNT.EQ.1) PRINT 202,MODE,YT,(TEMP(I),I=1,N)
000072       IF(YT-YH)12,20,20
          C  SUCCESSFUL MOVE, SPRING FORWARD
000075    12 DO 13  I=1,N
000077       BASE(I)=HEAD(I)
000101    13 HEAD(I)=TEMP(I)
000105       PRINT 201,KFCNS,MODE,YT,(HEAD(I),I=1,N)
000131       YH=YT
000133       GO TO 9
          C  FAILURE GO TO LOCAL SEARCH
000135    20 SPEED=1.
000136       MODE=2
000140       CALL LCLSR(N,HEAD,BASE,CURVE,SPEED,ISUCC,MODE,KFCNS,YH,KPRNT)
000151        IF(KFCNS.GT.MAXKFN) RETURN
000156       IF(ISUCC)  31,31,10
          C  MANY FAILURES, CUT STEPSIZE
000160    31 SPEED=0.
000161       MODE=4
000162       CALL LCLSR(N,HEAD,BASE,CURVE,SPEED,ISUCC,MODE,KFCNS,YH,KPRNT)
000174       IF(ISUCC) 41,41,10
000200    41 MODE=6
000201       IF(ICUT.GT.10) GO TO 99
000205       ICUT=ICUT+1
000206       DO 42 I=1,N
000207    42 BASE(I) =  .5 * BASE(I)+.5*HEAD(I)
000215       PRINT 201, KFCNS,MODE, Y H   ,(BASE(I),I=1,N)
000241       GO TO 31
000244    99 PRINT 203
000250       RETURN
000251   200 FORMAT (*1 PATTERN SEARCH HAS BEEN CALLED *//* KFCNS MODE* 5X
            1*COST*5X*PARAMETER VALUES*)
000251   201 FORMAT (2I4,10(E12.4))
000251   202 FORMAT(* TEMP*,I4,10(E12.4))
000251   203 FORMAT (* A LOCAL MAXIMUM HAS BEEN REACHED *)
000251       END
```

```
              SUBROUTINE LCLSR (N,HEAD,BASE, CURVE,SPEED,ISUCC,MODE,KFCNS,YH,
             1 KPRNT)
000015        DIMENSION BASE(10),HEAD(10),TEMP(10),PERP(10),CURVE(10)
       C      LOCAL  SEARCH
000015        ISUCC=0
000015        YBEST=YH
       C      TRY FORWARD MOVE
000017        DO 21 I=1,N
000020        CURVE(I)=0.
000021     21 TEMP(I)=HEAD(I)+(.4+SPEED)*(HEAD(I)-BASE(I))
000034        CALL ERR(TEMP,YT)
000036        IF(KPRNT.EQ.1) PRINT 202,MODE,YT,(TEMP(I),I=1,N)
000073        KFCNS=KFCNS+1
000075        IF (YT-YH)    26,22,22
       C      FAILURE ,MOVE BACK
000100     22 DO 23  I=1,N
000102     23 TEMP(I)=HEAD(I)+(SPEED-.4)*(HEAD(I)-BASE(I))
000116        CALL ERR(TEMP,YT)
000120        IF(KPRNT.EQ.1) PRINT 202,MODE,YT,(TEMP(I),I=1,N)
000155        KFCNS=KFCNS+1
000157        IF(YT-YH)26,24,24
000162     24 IF(SPEED.LE.0.) GO  TO 30
       C      ANOTHER FAILURE,MOVE BACK
000164        DO 25  I=1,N
000165     25 TEMP(I)=HEAD(I)+SPEED*(HEAD(I)-BASE(I))
000176        CALL ERR(TEMP,YT)
000200        IF(KPRNT.EQ.1) PRINT 202,MODE,YT,(TEMP(I),I=1,N)
000235        KFCNS=KFCNS+1
000237        IF(YT-YH) 26,30,30
       C      SUCCESS GO TO ORTHOGONAL SEARCH
000242     26 ISUCC=1
000243        YBEST=YT
       C      ORTHOGONAL SEARCH
000245     30 NORTH=1
000246        IF(N.GT.2)  GO TO 32
       C      MOVE LEFT
000251        PERP(1)=(- HEAD(2) + BASE(2))*.4
000254        PERP(2)=  (HEAD(1) - BASE(1)) * .4
000256        NORTH=NORTH+1
000260        GO TO 50
000261     32 RETURN
000262     50 DO 51 I=1,N
000264     51 TEMP(I)= TEMP(I)  +PERP(I)
000271        CM=1.
000272        CALL ERR(TEMP,YT)
000274        IF(KPRNT.EQ.1) PRINT 202,MODE,YT,(TEMP(I),I=1,N)
000331        KFCNS=KFCNS+1
000333        IF (YT     -YBEST  ) 55,52,52
       C      FAILURE,MOVE RIGHT
000336     52 DO 53 I=1,N
000340     53 TEMP(I)=TEMP(I)-2*PERP(I)
000346        CM=-1.
000350        CALL ERR(TEMP,YT)
000352        IF(KPRNT.EQ.1) PRINT 202,MODE,YT,(TEMP(I),I=1,N)
000407        KFCNS=KFCNS+1
000411        IF( YT    -YBEST   ) 55,57,57
000414     55 DO 56  I=1,N
```

```
000416        56 CURVE(I)= CM*PERP(I)  + CURVE(I)
000424           ISUCC=1
000425           YBEST=YT
000427           GO TO 59
            C    FAILURE, RESTORE TEMP
000427        57 DO 58 I=1,N
000431        58 TEMP(I)=TEMP(I)+PERP(I)
000436        59 IF(NORTH.LT.N) GO TO 32
000440        60 IF(ISUCC.EQ.0) GO TO 70
000441        61 DO 62  I=1,N
000443           BASE (I)=HEAD(I)
000446        62 HEAD(I)=TEMP(I)
000452           PRINT 201,( KFCNS,MODE,YBEST,(HEAD(I),I=1,N),(CURVE(I),I=1,N) )
000512            YB=YBEST
000514           RETURN
000515        70 MODE=MODE+1
000517           NROT=1
            C    MOVE HALF LEFT
000520           IF (N.LE.2) GO TO 80
000526        72 RETURN
000527        80 DO 81 I=1,N
000531        81 TEMP(I)=TEMP(I) +(PERP(I)+.4    *(HEAD(I)-BASE(I)))*.707
000544           NROT=NROT+1
000546           CM=1.
000547           CALL ERR(TEMP,YT)
000551           IF(KPRNT.EQ.1) PRINT 202,MODE,YT,(TEMP(I),I=1,N)
000606           KFCNS=KFCNS+1
000610           IF (YT - YBEST)   85,82,82
            C    FAILURE, MOVE HALF RIGHT
000613        82 DO 83 I=1,N
000615        83 TEMP(I)=TEMP(I)-1.414*PERP(I)
000623           CM=-1.
000625           CALL ERR(TEMP,YT)
000627           IF(KPRNT.EQ.1) PRINT 202,MODE,YT,(TEMP(I),I=1,N)
000664           KFCNS=KFCNS+1
000666           IF (YT    -YBEST ) 85,87,87
            C    SUCCESS
000671        85 ISUCC=1
000672           YBEST=YT
000674           DO 86 I=1,N
000675        86 CURVE (I)=CURVE(I)+ CM* (PERP(I) +.2   *(HEAD(I)-BASE(I)))*.707
000711           GO TO 89
            C    RESTORE TEMP
000712        87 DO 88 I=1,N
000714        88 TEMP(I) = TEMP(I) -(-PERP(I)+.2   *(HEAD(I)-BASE(I)))*.707
000727        89 IF (NROT.LT.N) GO TO 72
000731        90 IF(ISUCC) 91,91,61
000733       91  RETURN
000734       201 FORMAT (2I4,10(E12.4))
000734       202 FORMAT(* TEMP*,I4,10(E12.4))
000734           END
```

```
              SUBROUTINE    ERR(X,Y)
000005         COMMON GD(20),GP(20),TN(20),DN(10),DD(10)
000005        DIMENSION    TD(20),C(10),X(10),EN(20),ED(20)
000005        X(3)=2.5E6
000006        C(1)=X(1)
000007        C(2)=X(1)*X(2)/X(3)
000011        C(3)=X(1)/X(3)
000014        CALL PLYMLT(C,3,GP,15,TD,0)
000020        DO 10 I=1,17
000023     10 TD(I)=TD(I)+GD(I)
000030        CALL ROUTH (TD,Y,17)
000032        Y=Y*1.E100
000035        IF (Y.GE.1.0) RETURN
000040         CALL PLYMLT(DN,2,TD,17,EN,0)
000044         CALL PLYMLT(DD,3,TN,16,EN,18)
000050         CALL PLYMLT (DD,3,TD,17,ED,0)
000054        CALL INTSQ(ED,EN,19,Y)
000061        RETURN
000062        END
```

```
                 SUBROUTINE ROUTH (X,Y,L)
000006           DIMENSION X(20),A(20,10)
000006           DATA A/200*0.0/
000006           Y = J = 0
000010           M = L - 1
000012           DO 1 I = 1,M,2
000013           J = J + 1
000015           A(1,J) = X(I)
000021     1     A(2,J) = X(I + 1)
000027           IF (L - 2*J) 3,3,2
000031     2     A(1,J + 1) = X(L)
000036     3     J = 2
000037           N = M = (L + 1)/2.0
000044     4     IF (A(J,1)*A(J-1,1))7,5,5
000050     5     IF (J.EQ.L) RETURN
000052           DO 6 I = 2,M
000054     6     A(J + 1,I - 1) = A(J - 1,I) - (A(J - 1,1)*A(J,I))/A(J,1)
000102           IF (A(J + 1,1).EQ.0.0)A(J + 1,1) = 0.000001
000105           J = J + 1
000107           M = N + 1 - J/2.0
000114           GO TO 4
000114     7     Y = 1.0
000115           RETURN
000116           END
```

```
                SUBROUTINE INTSQ(A,C,N,S)
000007          DIMENSION B(20),A(20),C(20)
          C     RETURNS S=INTEGRAL OF C(S)*C(-S)/A(S)*A(-S) TO MAIN PROGRAM.
          C     B(S) HAS N-1 TERMS, A(S) HAS N TERMS.
          C     B(I) IS COEF. OF S**(2I-2)
          C     C(I) IS COEF. OF S**(I-1)
          C     IF THE LOWER ORDER DEN. AND NUM. COEFF. ARE SMALL (LESS THAN D) THE
          C     DIVIDE BOTH NUM. AND DEN. BY S**2.
000007          K=1
000007          D=1.0E-6
000011    1     IF (ABS(A(1))-D) 2,2,20
000015    2     IF (ABS(C(1))-D) 4,4,20
000021    4     IF (K-1) 5,5,6
000024    5     PRINT 200
000030    6     PRINT 201,A(1), C(1)
000047          K=K+1
000051          N=N-1
000054          DO 10  I=1,N
000056          A(I)=A(I+1)
000060          IF(I-N) 8,10,10
000062    8     C(I)=C(I+1)
000065    10    CONTINUE
000070          GO TO 1
000070    20    CALL PLYSQ (C,N,B)
000072          NM2=N-2
000077          DO 50 K=1,NM2
000100          NK=N-K
000101          BA=B(NK)/A(NK+1)
000104          AA=0.
000104          IF(K.EQ.1) GO TO 40
000106          AA=A(NK+2)/A(NK+1)
000110    40    NMK=(N-K)/2
000112          DO 50 I=1, NMK
000114          NKI=NK-2*I+1
000117          B(NK-I)=B(NK-I)-BA*A(NKI)
000124          A(NKI+1)=A(NKI+1)-AA*A(NKI)
000127    50    CONTINUE
000134          S=ABS(B(1)/(2.*A(2)*A(1)))
000140          RETURN
000141    200   FORMAT (*0 THE FOLLOWING COEFF. WERE FOUND TO BE SMALL AND CANCELL
               1ED IN THE NUMERATOR AND DENOMINATOR*)
000141    201   FORMAT (*  THE DENOMINATOR COEFF.=* E10.3 * THE NUMERATOR COEFF.
               1=* E10.3)
000141          END
```

```
                  SUBROUTINE PLYSQ (C,N,B)
000006            DIMENSION C(40),B(40)
            C     RETURNS B(S) = C(S)*C(-S) TO MAIN PROGRAM.
            C     B(I) IS COEF. OF S**I-1 IN B(S).
            C     C(I) IS COEF. OF S**I-1 IN C(S).
            C     N-1 IS NUMBER OF COEF. IN B(S) AND C(S).
000006            N1=(N-1)/2
000010            DO 20 I=1,N1
000012            MO =-1
000013            B(I)=0.
000014            II = 2*I-1
000016            DO 20 K=1, II
000020            MO = -1*MO
000022      20    B(I)=B(I)+MO*C(K)*C(2*I-K)
000037            NM1=N-1
000040            N2=N1+1
000042            DO 30 I=N2,NM1
000043            IIN=2*I-NM1
000044            B(I)=0.
000046            MO=(-1) ** NM1
000051            DO 30 K=IIN,NM1
000053            MO = -1*MO
000055      30    B(I)=B(I)+MO*C(K)*C(2*I-K)
000071            RETURN
000072            END
```

VARIABLE METRIC

Introduction

This report describes the variable metric optimization scheme (hereafter referred to as variable metrics) and its application to the design problem for an inlet control system of an air-breathing jet engine. The application strategy seeks to force the total closed-loop response of the system to match a desired frequency response over a range of frequencies from 1 cps to 151 cps. Variable metrics is used to minimize the total squared error over this range of frequencies. Successful, physically realizable control of the inlet system is achieved for a variety of parameter values, with system stability checked in each case.

The general type of problem, to which variable metrics is applicable, is the optimization of a scalar-valued function y of N real variables x_j, with respect to the N values of these x_j: an N-dimensional optimization problem. "y" is an arbitrary performance index, descriptive of the effectiveness of a control system. Thus, scalar optimization techniques are of rather general applicability in the design of optimal control systems.

Three distinct advantages of variable metrics over all other optimization techniques, which the author has studied, are:

1. Internal determination of the search path, leading to an optimum.
2. Adaptability to general statistical study of a particular problem's solution properties.

3. Higher resistance to presence of local optima.

Four distinct disadvantages of variable metrics are:

1. Lack of external control over step size.
2. Sensitivity of reliable convergence to values of convergence parameters.
3. Performance dependent on particular one-dimensional search method used. (This is sometimes advantageous.)
4. Storage requirements of two N x N matrices for an N-dimensional optimization problem.

Optimization Procedure

The operation of variable metrics is very simple indeed: One tight computation loop is executed and reiterated until convergence. The purpose of the technique is the minimization of a real-valued function $y(x)$. The point x^*, at which y achieves its optimum value, is called an "optimum". An initial point x_o in the search for x^* is chosen, as an educated guess, at x^* on the part of the user.

Initialization

1. The input starting point x_o in N-space is used to obtain an initial gradient vector $\nabla y_o = \nabla y(x_o)$. Put $H_o = I_N$ for N dimensional identity matrix I_N, and put $j = 1$.

Computation Loop

2. A point x_j is found on the line through x_{j-1} in the direction of $H_{j-1}\nabla y_{j-1}$ by one-dimensional optimization

of the value of $y(x)$ along this line. Upon the determination of $\nabla y_j = \nabla y(x_j)$, the matrix H_j is calculated.

3. Put $j = j + 1$, and reenter again at step 2.

For example, a point x_1 is found on the line through x_o in the direction of $H_o \nabla y_o$ by one-dimensional optimizaiton, where H_1 represents a "metric", which aids reliable convergence, by prohibiting repeated searches along the same direction ("oscillation"). Upon the determination of $\nabla y_1 = \nabla y(x_1)$, matrix H_1 is calculated and the steps are reiterated. Notice that $H_o = I_N$ means that the initial ($j = 1$) one-dimensional optimization is merely a gradient search along the line through x_o in the direction of ∇y_o.

The metric matrix H_j is the sum of two other internal matrices. When computing each of these last two matrices, there is a point at which division occurs. Now, division by zero is not permitted, but convergence of the computations is nevertheless represented by a divisor which is extremely small (numerical zero). Consequently, convergence is checked by the magnitude of the divisors prior to division. Thus, after N iterations, convergence is tested at two places in each additional iteration of the computation loop. The reason for waiting for the completion of N iterations is the prevention of premature convergence to "pseudo-optima", until all different directions of search have been checked at least once.

Here is a summary of the minimization procedure used by variable metrics:

1. Input N-vector x_o, calculate $\nabla y_o = \nabla y(x_o)$, put $H_o = I_N$ for $N \times N$ identity matrix I_N, and set $j = 1$.

2. Search for "minimum" x_j on $x_{j-1} - \alpha H_{j-1} \nabla y_{j-1}$ for $0 < \alpha \leq 1$, calculate $\nabla y_j = \nabla y(x_j)$, compute matrix H_j (while twice checking for convergence), set $j = j + 1$, and repeat 2.

The reader is referred to the book *Foundations of Optimization* by Wilde and Beightler for an equivalent, but different, description of the deflected gradient version of variable metrics popularized by Fletcher and Powell. For a detailed exposition of the variable metric method of optimization, reference can be made to Fletcher and Powell's paper of 1963, or to "GOSPEL" by Dr. Huelsman at the University of Arizona (September 1968).

Discussion of Computer Programs

Subroutine FP represents the digital mechanization of the aforementioned variable metric minimization scheme. As FP is merely a subroutine, input and output data are transferred by means of calling sequence (c.f., computer listing at the end of this section).

In order to function, FP needs only five bits of information and a lot of storage:

N - the number of unknown parameters x_j

XO - the initial estimate vector of unknown parameters x_j

CRIT - convergence criterion (numerical zero)

L - the maximum number of iterations of the variable metric loop

M - the number of iterations per search of the one-dimensional search routine.

The inputs are unaltered by the program, and three outputs are calculated:

X - the final (and optimal) estimate vector of unknown parameters x_j

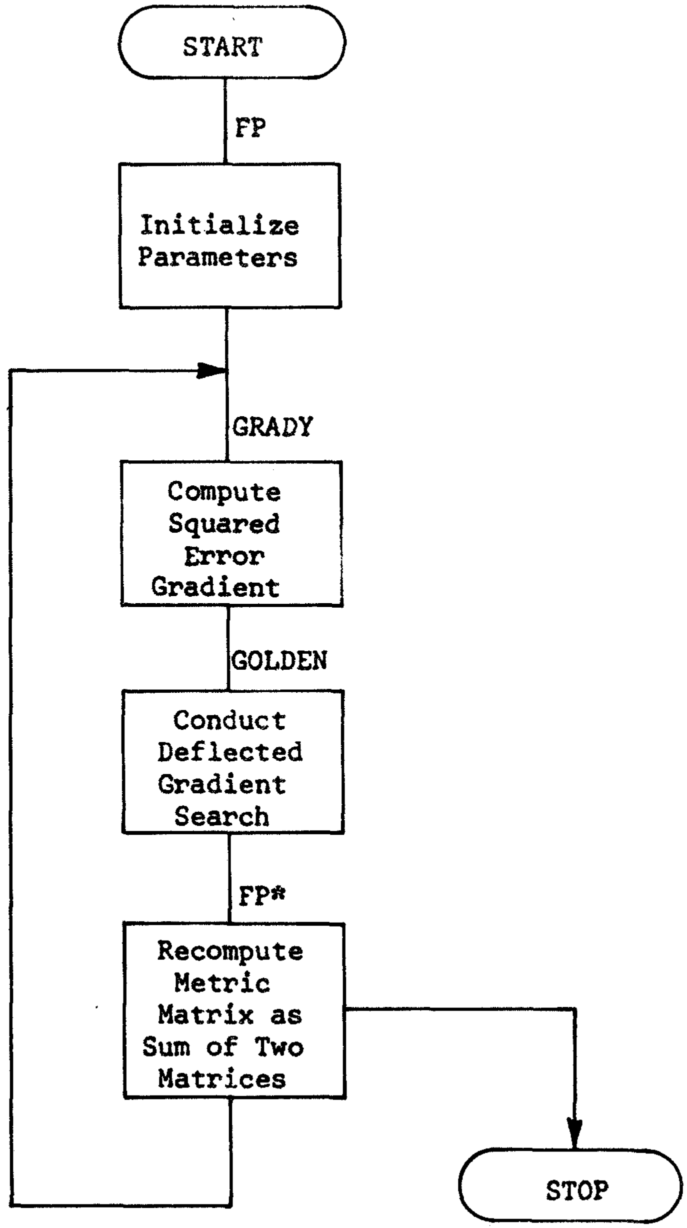
QINV - the estimate of the matrix of second partial derivatives at the optimum. This is the storage area for the metric matrix H.

Y - the optimized value of the scalar function y(x)

Two other variables ICALL and IC are of interest. ICALL is incremented by 1 for each time that FP requires a gradient evaluation. IC is incremented by 1 for each time that the one-dimensional minimization requires a function evaluation. The flow diagram appears in Fig. 1 and is discussed on pages 79 to 80.

The various N-dimensional buffers accomodate current and previous gradients, step sizes, search vectors, and a scratch buffer for matrix manipulations. Two N x N matrices are calculated and added to the previous N x N matrix H_{k-1} to obtain the current matrix H_k. This provides the two opportunities for convergence tests. Storage requirements have been kept to a minimum because all storage buffers must be furnished by the user with FP requiring only enough memory locations to perform multiplications, additions, etc.

Two essential inputs by the user are a function generator along with a gradient generator. Comparative values of y(x) and $\nabla y(x)$ are thus made available to the program. In the current formulation, both of these operations are performed by the subroutine GRADY. The method of Golden Sections is used for one-dimensional minimization. This is mechanized in subroutine GOLDEN. Descriptions of the method of Golden Sections can be found in the aforementioned reference of Wilde and Beightler.



*Check for exit condition of convergence after N iterations.

Fig. 1. Flow Diagram of Variable Metrics Program FP.

Application to Inlet Bypass Door Servo

This particular example system is diagrammed on page 69. Referring to the diagram of this particular problem, the symbols all represent transformed functions of the complex frequency p:

$$P(p) = \frac{p + 1010}{1010}$$

$$C(p) = x_1 \frac{(p + x_3 + jx_2)(p + x_3 - jx_2)}{(p + 4000)^2}$$

$$G(p) = \left(\frac{371.5}{p+46}\right) \frac{(p+80.29+j172.2)(p+80.29-j172.2)(p+173.16+j332.62)}{(p+44.57+j281.49)(p+44.57-j281.49)(p+84.24+j477.63)}$$

$$\frac{(p+173.16-j332.62)(p+75.69+j575.03)(p+75.69-j575.03)}{(p+84.24-j477.63)(p+130.2+j738.6)(p+130.2-j738.6)}$$

$$\frac{(p+82.8+j946.39)(p+82.8-j946.39)}{(p+191.07+j1083.28)(p+191.07-j1083.28)}$$

The closed loop transfer function of the servomechanism is seen from the figure to be

$$T(p) = \frac{G(p)}{1 + G(p)\cdot P(p)\cdot C(p)\cdot D(p)}$$

where the component transfer functions are given below:

$$D(p) = \frac{(1.3073x10^{20})}{p(p+2000)(p+967.5)(p+1461+j4009)(p+1461-j4009)}$$

$$\frac{1}{(p+318.5+j1899)(p+318.5-j1899)}$$

Δw is the frequency spectrum of input additive noise, and $\Delta x'$ is the frequency content of the undelayed shock position to be regulated. G is the fixed inlet dynamics; P is the frequency relation between shock position and exit pressure; C is the compensator of the form

$$K \frac{(p + p_0)(p + \bar{p}_0)}{(p + a)^2} ;$$

and D gives the dynamics of the bypass door. All transfer functions are fixed except C, and the whole problem is the search for three controlling parameters, $K = x_1$, $P_0 = x_3 + jx_2$, and $\bar{P}_0 = x_3 - jx_2$, in order that

$$Y \equiv \sum_{k=1}^{151} \left[T(j2\ k) - \frac{371.5(j2\pi k)^3}{(j2\pi k + 30)(j2\pi k + 37150)} \right]^2$$

be a minimum. Computer results give satisfying verification of predictions.

Numerical Results

Firstly, the frequency response, which it is desired to match, is uniformly 40 db down or more over the range of frequencies between 1 and 151 cycles per second.

Secondly, a chart of results can be given below:

x_1	x_2	x_3	y	Z	Stability
6×10^{-5}	248	142	13.9	0.4	Yes
1.0	248	142	13.9	0.5	Yes
1000	248	142	13.5	2.1	Yes
10024	86.1	778	8.2	4.1	Yes
100007	-192	2435	0.02	34.8	No
10^6	222	551	0.004	37.0	?

The entries beneath the label Z stand for the minimum attenuation in db of the closed-loop frequency response over the range of frequencies from 1 to 121 cycles per second.

Low frequency sensitivity appears to decrease with increasing values of x_1 in the formula

$$C(p) = x_1 \frac{(p + x_3)^2 + x_2{}^2}{(p + 4000)^2}.$$

However, previous studies have demonstrated that it is insufficient to merely have $C(p) = x_1$ and increase x_1 in order to reduce low frequency response because instability occurs as it appears to do in the runs of the table.

Conclusions

This paper has demonstrated the application of a powerful optimization strategy for the case of cost functions, which can be written as the ratio of two polynomials. The application is the discovery of several combinations of acceptable parameter values for a controller, which stabilizes and minimizes the low frequency sensitivity of an Inlet Bypass Door Servo.

In reference to the schematic system diagram on page 69, suitable combinations of x_1, x_2, and x_3 in

$$C(p) = x_1 \frac{(p + x_3 + jx_2)(p + x_3 - jx_2)}{(p + 4000)^2}$$

are $x_1 = 6 \times 10^{-5}$, $x_2 = 248$, $x_3 = 142$, and $x_1 = 10024$, $x_2 = 86.1$, $x_3 = 778$.

The value of the approach taken in this paper is that its success is independent of the particular numbers used to specify the component transfer functions, and it converges rapidly and reliably for lower degree problems while being simultaneously reliable at higher degrees.

References

[1] Wilde, D. J., and Beightler, C. S., *Foundations of Optimization*, Prentice Hall, New Jersey, 1967.

[2] Fletcher and Powell, "A Rapidly Convergent Descent Method for Minimization," *Comp. J., 6,* No. 2, 1963, pp. 163-168.

[3] Huelsman, L. P., "GOSPEL", NASA Contract NGL-03-002-136, University of Arizona, Sept. 1968.

```
      SUBROUTINE FP(X0, N, M, DELY0, DELYN, X1, X, A, XN, DELX, QINV,
     1XP, BUF, Y, ICALL, IC, CRIT, L)
C
C FLETCHER-POWELL MINIMIZATION OF FUNCTION Y=Y(X) OF N VARIABLES X(I) WITH
C  INITIAL POINT X0 OF CONVERGENT SEQUENCE OF X≠S. L IS THE MAXIMUM NUMBER OF
C  ITERATIONS OF FP, M IS THE MAXIMUM NUMBER OF ITERATIONS FOR GOLDEN.
C  L, M, AND N MUST ALL BE POSITIVE INTEGERS.  CRIT IS THE CONVERGENCE
C  CRITERION FOR THE SEQUENCE OF X≠S.
C
C FOR THE FOREGOING INPUT, SUBROUTINE FP RETURNS AN ESTIMATE QINV OF THE INVERSE
C  OF THE HESSIAN MATRIX AND THE MINIMUM VALUE Y OF Y(X), EVALUATED AT THE
C  MINIMUM X.
C
C ICALL COUNTS THE NUMBER OF TIMES FP CALLS GRADY (FOR GRADIENTS).
C IC COUNTS THE NUMBER OF TIMES GOLDEN CALLS GRADY (FOR FUNCTION VALUES).
C
C THE REMAINING UNDESCRIBED PARAMETERS ARE FOR INTERNAL USE ONLY.
C
C MINIMIZE Y(X)=Y0+CT*X +0.5*XT*Q*X (X IN E(N))
C            DELY=C+QX
C           DDELY=Q*DX (BACKWARD DIFFERENCE D)
C              DX=QINV*DDELY
C              DX=-QINV*DELY0 (AT OPTIMUM)
C               Q UNKNOWN
C
C 1. CHOOSE X0 ARBITRARILY WITH DELY0=DELY(X0)
C 2. PERFORM ONE-DIMENSIONAL SEARCH WITH X(1)=X0-MU*DELY0
C              Y(X(1))=INF Y(X(1, MU)) OVER MU
C 3. PUT QINV(0)=I(N)
C 4. COMPUTE X(M)=X(M-1)-MU*QINV(M-1)*DELY(M-1)
C              Y(X(M))=INF Y(X(M, MU))
C 5. COMPUTE QINV(M)=QINV(M-1)+DX(M)*DX(M)T/(DX(M)T*DDELY(M))
C          +QINV(M-1)*DDELY(M)*DDELY(M)T*QINV(M-1)/(DDELY(M)T*QINV(M-1)*DDELY(M))
C THEN, QINV(N) APPROXIMATES QINV, AND X(N) ESTIMATES THE MINIMUM X.
C
C X(N)=X0-QINV(N)*DELY0
C       OR
C X(N)=X0-SUM(MU(M)*QINV(M-1)*DELY(M-1))     (M=1,...,N)
C
C
      DIMENSION X0(N), DELY0(N), DELYN(N), X1(N), X(N), A(N, N), XN(N),
     1DELX(N), QINV(N, N), BUF(N), XP(N)
      IONE=1
      CALL MOVE(XP, X0, N, IONE)
      CALL GRADY(XP, Y, DELY0, ICALL)
      CALL MIDENT(QINV, N)
C
      DO 3 K=1, L
C
      CALL MINTON(QINV, N, N, DELY0, IONE, DELYN)
      CALL CHANGE(DELYN, N, IONE)
      CALL MPLUSN(XP, DELYN, N, IONE, X1)
      CALL GOLDEN(M, N, XP, X1, X, A, DELX, DELYN, BUF, XN, IC)
C
      CALL GRADY(XN, Y, DELYN, ICALL)
      CALL CHANGE(DELY0, N, IONE)
      CALL MPLUSN(DELYN, DELY0, N, IONE, DELY0)
```

```
      CALL MINTON(QINV, N, N, DELYO, IONE, BUF)
      CALL MINTON(BUF, N, IONE, BUF, N, A)
      CALL INNER(DELYO, BUF, N, IONE, IONE, D)
      IF(D.LE.0.0) GOTO 4
      IF(D.GT.CRIT) GOTO 1
      IF(K.GT.N.OR.K.EQ.1) GOTO 4
1     DO 322 I=1, N
      DO 322 J=1, N
322   A(I, J)=-A(I, J)/D
      CALL MPLUSN(QINV, A, N, N, QINV)
      CALL CHANGE(XP, N, IONE)
      CALL MPLUSN(XN, XP, N, IONE, DELX)
      CALL MINTON(DELX, N, IONE, DELX, N, A)
      CALL INNER(DELX, DELYO, N, IONE, IONE, D)
      IF(D.LE.0.0) GOTO 4
      IF(D.GT.CRIT) GOTO 2
      IF(K.GT.N.OR.K.EQ.1) GOTO 4
2     DO 32 I=1, N
      DO 32 J=1, N
32    A(I, J)=A(I, J)/D
      CALL MPLUSN(QINV, A, N, N, QINV)
C
      CALL MOVE(XP, XN, N, IONE)
      CALL MOVE(DELYO, DELYN, N, 1)
C
3     CONTINUE
C
4     CALL MOVE(X, XN, N, IONE)
C
      RETURN
C PREPARED UNDER NASA CONTRACT NGR-03-002-115
      END
```

```
      SUBROUTINE GOLDEN(M, N, X1, X2, X3, X4, X5, X6, X7, X8, IC)
C ONE-DIMENSIONAL MINIMIZATION OF Y=Y(X) BY GOLDEN SECTION WITH N VARIABLES
C  X(I), ENDPOINTS X1, X2 OF SEARCH INTERVAL, AND OPTIMAL POINT X8 OUTPUT
C M IS THE NUMBER OF ITERATIONS PER SEARCH.

      DIMENSION X1(N), X2(N), X3(N), X4(N), X5(N), X6(N), X7(N), X8(N)
      IONE=1
      G=1.618033989
      G1=G-1.0
      G2=2.0-G
      CALL MOVE(X6, X1, N, IONE)
      CALL MOVE(X7, X2, N, IONE)
      CALL MOVE(X8, X6, N, IONE)
      CALL CHANGE(X8, N, IONE)
      CALL MPLUSN(X7, X8, N, IONE, X4)
      DO 1 J=1, N
      X4(J)=G2*X4(J)
      CALL MPLUSN(X6, X4, N, IONE, X3)
      CALL GRADY(X3, Y1, X8, IC)
      CALL CHANGE(X4, N, IONE)
      CALL MPLUSN(X7, X4, N, IONE, X5)
      CALL GRADY(X5, Y2, X8, IC)
      CALL CHANGE(X4, N, IONE)
C Y1=Y(X1+G2*DX)
C Y2=Y(X2-G2*DX)
      DO 3 I=1, M
      DO 31 J=1, N
1     X4(J)=G1*X4(J)
      IF(Y1.LT.Y2) GOTO 33
      CALL MOVE(X6, X3, N, IONE)
      CALL MOVE(X3, X5, N, IONE)
      CALL CHANGE(X4, N, IONE)
      CALL MPLUSN(X7, X4, N, IONE, X5)
      CALL CHANGE(X4, N, IONE)
      Y1=Y2
      CALL GRADY(X5, Y2, X8, IC)
      GOTO 3
3     CALL MOVE(X7, X5, N, IONE)
      CALL MOVE(X5, X3, N, IONE)
      CALL MPLUSN(X6, X4, N, IONE, X3)
      Y2=Y1
      CALL GRADY(X3, Y1, X8, IC)
      CONTINUE
      X=1.0/(2.0*G2)
      DO 5 I=1, N
      X4(I)=X*X4(I)
      CALL MPLUSN(X6, X4, N, IONE, X8)
      RETURN
C PREPARED UNDER NASA CONTRACT NGR-03-002-115
      END
```

```
      SUBROUTINE GRADY(X, Y, DY, IC)
      DIMENSION X(1), DY(1)
      COMPLEX Z, G, D, P, A, B, T, C, S, DC(3)
      Y=0
      CALL MZERO(DY, 3, 1)
      IC=IC+1
      DO 1 I=1, 151, 3
      IF=I
      F=I
      S=CMPLX(0.0, 6.28318531*F)
      CALL GF(IF, G)
      CALL DF(IF, D)
      CALL PF(IF, P)
      CALL CF(IF, X, C)
      CALL TF(IF, T)
      A=P*D*G
      B=1.0+C*A
      Z=G/B-T
      DC(1)=((S+X(3))**2+X(2)**2)/(S+4000.0)**2
      DC(2)=2.0*X(1)*X(2)/(S+4000.0)**2
      DC(3)=2.0*X(1)*(S+X(3))/(S+4000.0)**2
      C=A*G*CONJG(Z)/B**2
      DO 11 J=1, 3
1     DY(J)=DY(J)-2.0*REAL(C*DC(J))
      Z=Z*CONJG(Z)
      Y=Y+Z
      CONTINUE
      RETURN
      END
```

```
      SUBROUTINE GF(T, G)
      COMPLEX G, P
      F=T
      P=CMPLX(0.0, 6.28318531*F)
      G=371.5*((P+80.29)**2+172.2**2)*((P+173.16)**2+332.62**2)
     1        *((P+75.69)**2+575.03**2)*((P+82.8)**2+946.39**2)/
     2  ((P+46.0)*((P+44.57)**2+281.49**2)*((P+84.24)**2+477.63**2)
     3        *((P+130.2)**2+738.6**2)*((P+191.07)**2+1083.28**2))
      RETURN
      END
```

```
      SUBROUTINE DF(I, D)
      COMPLEX D, P
      F=I
      P=CMPLX(0.0, 6.28318531*F)
      D=1.3073*10.0**20/(P*(P+2000.0)*(P+967.5)*((P+1461.0)**2+4009.0**2)
     1   )*((P+318.5)**2+1899.0**2))
      RETURN
      END
```

```
      SUBROUTINE PF(I, PP)
      COMPLEX PP, P
      F=I
      P=CMPLX(0.0, 6.28318531*F)
      PP=(P+1010.0)/1010.0
      RETURN
      END
```

```
      SUBROUTINE CF(I, X, C)
      DIMENSION X(1)
      COMPLEX C, P
      F=I
      P=CMPLX(0.0, 6.28318531*F)
      C=X*((P+X(3))**2+X(2)**2)/(P+4000.0)**2
      RETURN
      END
```

```
      SUBROUTINE TF(I, T)
      COMPLEX T, P
      F=I
      P=CMPLX(0.0, 6.28318531*F)
      T=371.5*P/((P+30.0)*(P+37150.0))
      RETURN
      END
```