DETERMINATION OF OPTIMAL PARAMETERS FOR DYNAMICAL SYSTEMS

By

DENNIS RAY UNRUH

Bachelor of Science
Kansas State University
Manhattan, Kansas
1966

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1968

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
May, 1970

Thesis
1970 D
L59d
cop.2

OKLAHOMA STATE UNIVERSITY LIBRARY
OCT 15 1963

DETERMINATION OF OPTIMAL PARAMETERS

FOR DYNAMICAL SYSTEMS

Thesis Approved:

O. Fitch
Thesis Adviser

Henry R Seberta

J. E Bose

Charles M. Bacon

D. Durham
Dean of the Graduate College

762837

ACKNOWLEDGMENT

I thank my wife, Kathy, for the love and patience that she has given me during my graduate program. A man in the midst of a graduate studies program is sometimes not a very congenial husband.

I thank my parents, Harvey and Geneva Unruh, for the financial support which they provided during my undergraduate schooling, but most of all I thank them for the encouragement which they have always provided.

In addition, I thank the following people and organizations for the support and encouragement which they supplied during my doctoral program:

Dr. E. C. Fitch, my graduate committee chairman, provided valuable general guidance and criticism throughout my doctoral work. I received financial support from the Basic Fluid Power Research Program which is being carried on at Oklahoma State University. Without the dynamic leadership of Dr. Fitch, this program would likely have never been started.

Dr. H. R. Sebesta, Dr. J. E. Bose, and Dr. C. M. Bacon were the other members of my graduate committee. They provided many hours of help throughout the preparation of my thesis.

The Federal Government helped provide my financial support through a National Defense Educational Act Fellowship.

Miss Velda Davis is responsible for the final typing of this manuscript as well as the final steps in completion of this program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION AND LITERATURE SURVEY

Introduction

What is System Identification?

System identification is the process of determining a suitable mathematical model for a dynamic system. A schematic flow diagram is given in Figure 1. In order to explain this diagram in complete detail, it will be hypothesized that the system shown schematically in Figure 2 has been built and is undergoing experimental testing in the laboratory. The previous statement implies that the physical system must be subjected to some experimental runs in the laboratory before system identification can be performed.

In the system shown in Figure 2, it is assumed that the hydraulic ram has reached the end of its stroke and that a constant flow pump is still delivering oil to the system. Hence, the relief valve must open and allow the oil to return to tank. In this system, there are two variables which might conveniently be measured. The first is the movement, as a function of time, of the stem of the poppet in the poppet valve, and the second is the variation of the

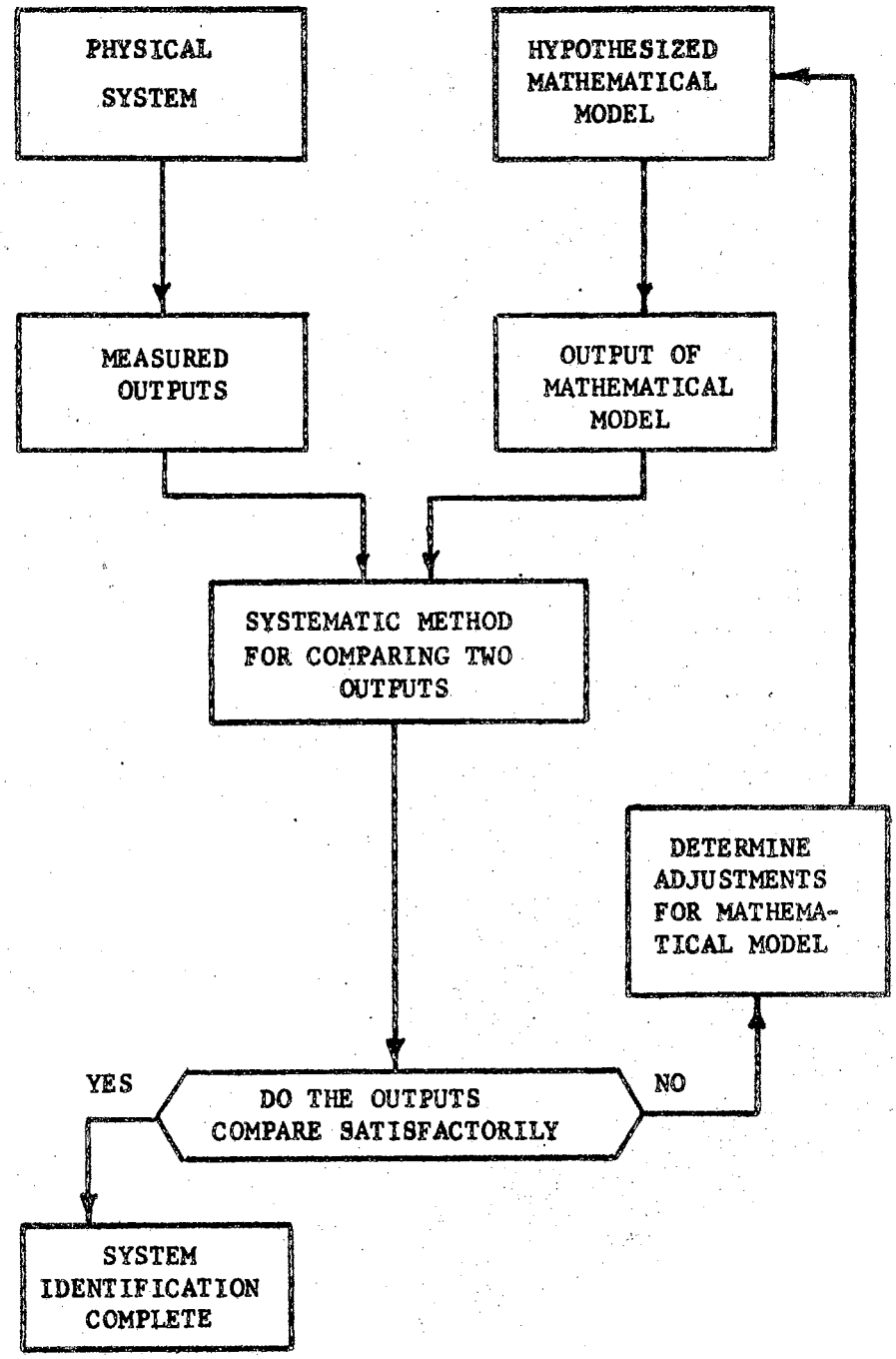


Figure 1. System Identification

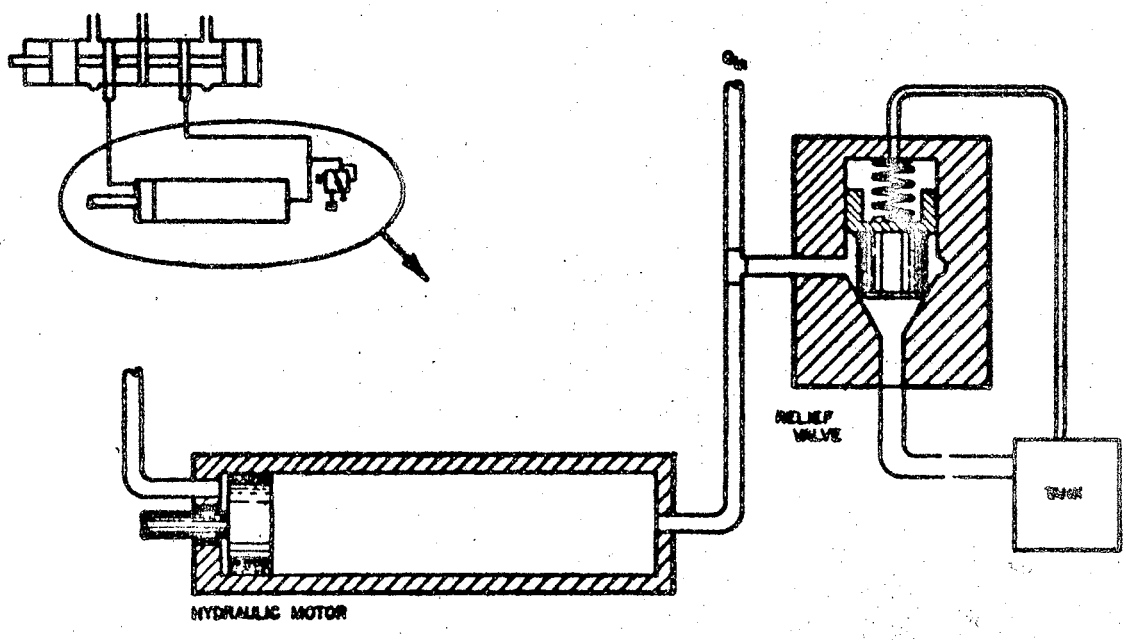


Figure 2. Hydraulic System

pressure in the inlet chamber of the poppet valve. If the valve were performing improperly, typical response plots might appear as shown in Figure 3.

The oscillating nature of the pressure and stem position as indicated in Figure 3 is undesirable, and it would be to the manufacturer's advantage if he could determine the cause of this improper performance. This is the point at which system identification enters into the problem. If a satisfactory mathematical model can be found which will yield a response which is identical to the response shown in Figure 3, the engineer can probably determine how to adjust the physical system to improve its responses by adjusting the mathematical model and observing how the mathematical model response varies as the model is varied.

At this point it is now possible to return to Figure 1 and explain the basic flow diagram for system identification in greater detail. First of all a physical system must be built and subjected to experimental tests in the laboratory. This yields the information in the upper left-hand portion of the flow diagram. Next, the user must hypothesize a mathematical model for the system, and this model is then used to simulate the response of the system. This provides the information in the upper right-hand portion of the flow diagram. The response of the mathematical model and the actual system must then be compared, and if they do not satisfactorily agree, the mathematical model must be adjusted until they do agree satisfactorily. There are various

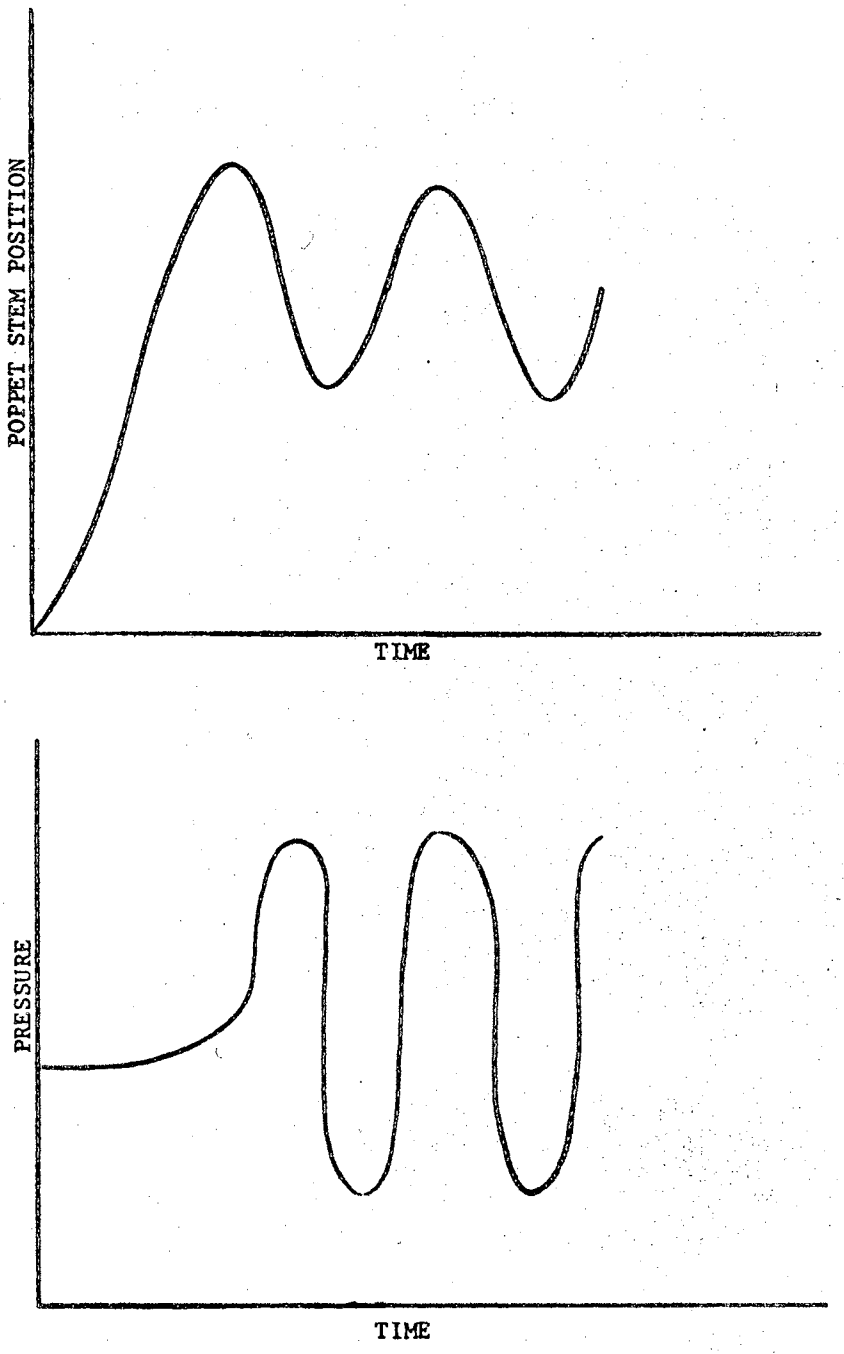


Figure 3. Initial Response of System of Figure 2

philosophies as to what is the best way of comparing the system response with the mathematical response and how the mathematical model should be varied if they do not agree satisfactorily.

As stated earlier, the user must hypothesize the basic form of the mathematical model which is to be used in the identification procedure. Hence, depending upon what the mathematical model is to be used for, the user may be interested in either a "black box" model or a "parameter estimation" model. An example of a situation in which the user might be interested in a "black box" mathematical model is that of determining the dynamic response of an internal combustion engine which is being used to supply power for a dynamic process. Since the user is sure in this situation that the structure of the internal combustion engine is not going to be varied during the experiments to be performed, he will be satisfied with any mathematical model which will predict the response speed of the engine versus its dynamical load. In other words, the parameters in the mathematical model need not have any relationship to the physical components of the engine.

The previous example of the hydraulic relief valve is a situation in which the user would be interested in a "parameter estimation" model. This is due to the fact that after the system has been satisfactorily identified, the user will want to adjust the mathematical model to improve the predicted response. If the user cannot, at this point,

relate the adjustments on the mathematical model to physical components in the system, his work cannot be used to its greatest advantage.

What is System Design?

One way of looking at a system design is to consider it as the process of determining the proper sizes of physical components in a system from its mathematical model. A schematic flow diagram for this view of system design is shown in Figure 4. It is interesting to note that in system design the first thing that must be specified is the desired system response. For the poppet relief valve example of the previous section, typical desired responses might be as shown in Figure 5.

The next step is to determine a relevant mathematical model for "parameter estimation". It must be a "parameter estimation" mathematical model since in the design problem you must be able to relate the parameters of the mathematical model to the physical components of the system. Once the mathematical model has been established, a systematic method for comparing the results of the response of the mathematical model with the desired response must be formulated. If the mathematical response does not compare favorably with the desired response, the mathematical model must be adjusted. When the model has been satisfactorily adjusted, the physical sizes of the components can then be determined.

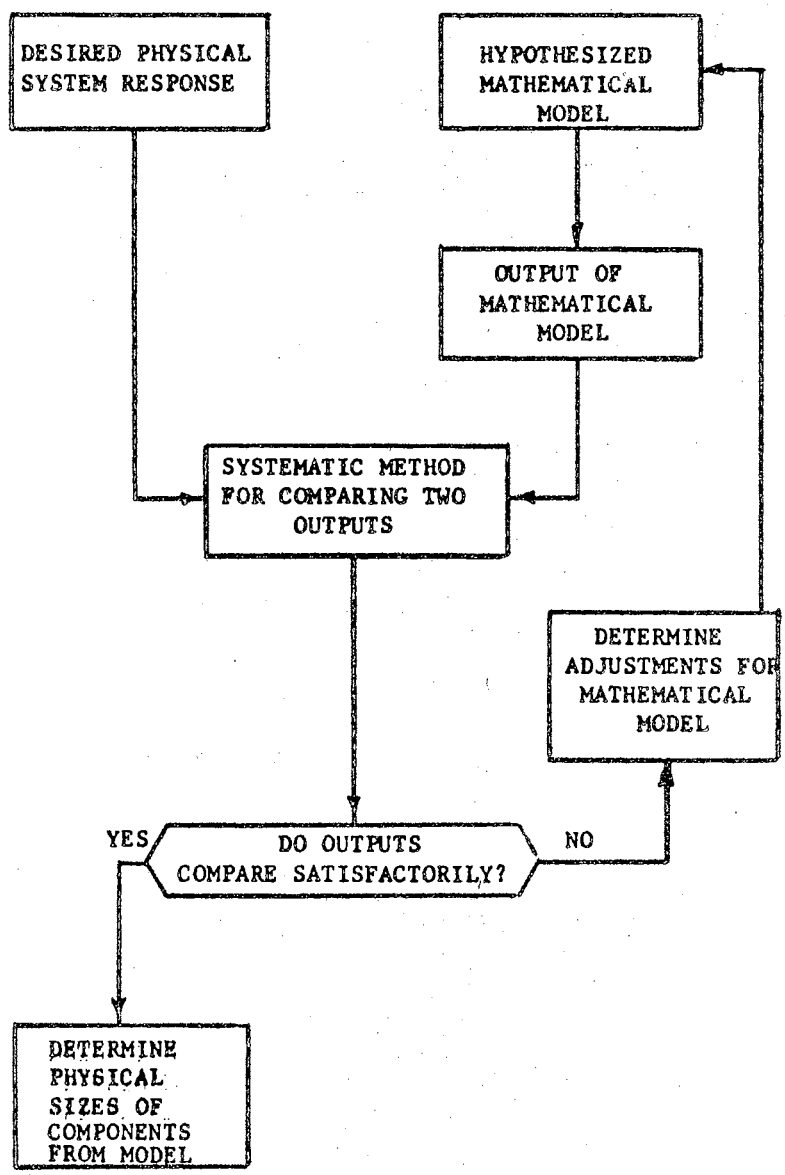


Figure 4. System Design

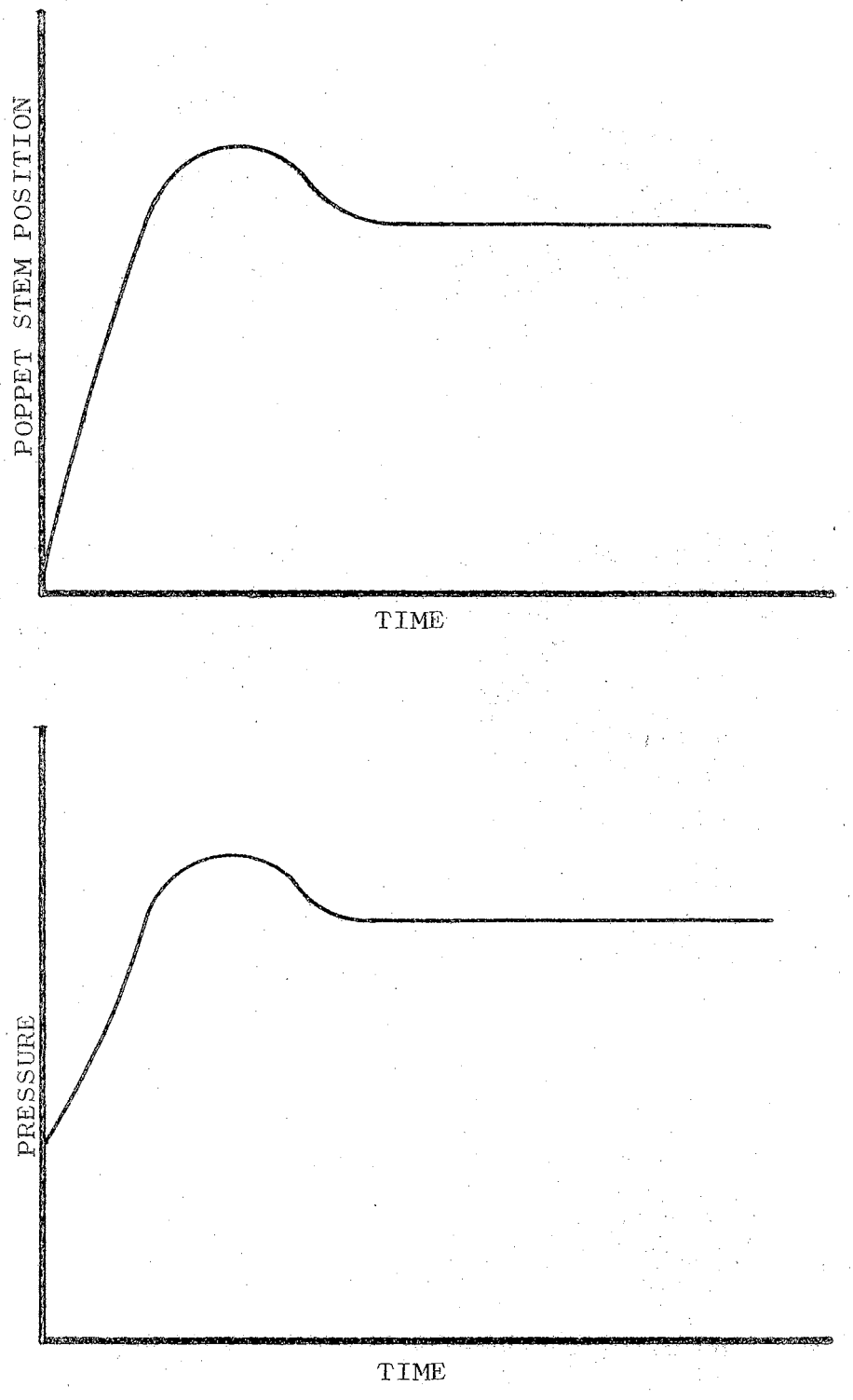


Figure 5. Desired Response of System of Figure 2

At this point it is very informative to compare the flow diagrams for System Design (Figure 4) and System Identification (Figure 1). The two diagrams are quite similar. Hence, it is not surprising that any method which can be used for system identification can also be used for system design.

Method Presented in This Thesis

The System Design and System Identification problem can be approached by either classical, statistical, or non-statistical methods. All three methods are discussed later in this chapter. Within this thesis a new non-statistical method is presented which is called the method of seeking principal planes.

In Chapter II the method of seeking principal planes is completely developed for the problem of determining the minimum of a function of several variables. Chapter II might be called a "capital goods expenditure" since the algebraic minimization techniques are not directly applicable to System Design and System Identification problems.

But in Chapter III it is demonstrated that optimal control theory can be used to convert a system design or identification problem into a two-point boundary value problem. The two-point boundary value problem can in turn be reformulated into an algebraic minimization problem, and the method of seeking principal planes becomes applicable.

If enough care is taken, the System Design or

Identification problem can be directly approached as an algebraic minimization problem. In Chapter IV the method of seeking principal planes is applied directly to System Design and Identification problems.

Chapter V presents some problems which arise when the dynamical systems which are being considered have certain types of discontinuities. Several approaches for resolving some of these difficulties are presented.

In Appendix A a description of a computational algorithm which is capable of analytically differentiating most mathematical expressions which can be written in the Fortran IV computer language is given. This algorithm was used extensively when programs were written to implement the method of seeking principal planes.

The analytical differentiation computational program relieves the user of the responsibility of determining the partial derivitives of the dynamical system equations. This eliminates most of the hand manipulations of the equations which the user must perform in order to use contemporary system identification techniques.

The rest of this introductory chapter presents the various methods available for solving System Design or Identification problems, a precise mathematical statement of the problem, and a comparison of presently available methods.

Classical Methods

The classical methods of system identification are based on s-plane analysis. The methods are applicable only to linear systems and, in general, only to systems which have one input and one output. The essential final result of s-plane analysis is that the characteristic roots are placed in positions such that the system is both stable and properly damped.

The use of classical methods of system identification requires that the input must be either sinusoidal, step, or impulse. When a sinusoidal input is used, the frequency of the input is varied and the amplitude of the response is noted. A log-magnitude (or Bode) versus frequency plot is then drawn and the systems is identified as well as possible from the shape of the plot.

Detailed discussion of the procedures of using frequency response methods for determining models is found in most texts on automatic control.

The main shortcomings of classical methods are: (1) they are applicable only to linear systems, (2) they are applicable only to systems with one input and one output, and (3) the system must be subjected to special inputs.

Statistical Methods

The three basic statistical methods used for system identification as presented by Aoki (1) are Least Squares Estimation, Maximum Likelihood Estimation, and Bayesian

Estimation. For linear systems with Gaussian (normal distribution) random noise, these three methods are equivalent. In order to estimate the parameters by Least Squares Estimation, nothing needs to be known about the various probability distributions. In order to use Maximum Likelihood Estimation or Bayesian Estimation, further statements concerning various probability distributions must be made.

Non-Statistical Methods

In the literature survey which was performed for this report, there were five non-statistical methods of system identification found. They are: (1) Optimal Control, (2) Differential Approximation, (3) Bacon's Method, (4) Parameter Influence Coefficients, and (5) Quasilinearization. A discussion of the basic philosophy of each of these methods will be presented later in this chapter.

Statement of Problem

Mathematical Model

In this section the identification and modeling problem will be stated in precise mathematical notation. The first thing which is necessary is for the user to supply the basic form of the differential equations in state variable notation.

$$\dot{X} = F(X, P, t). \qquad (1\text{-}1)$$

In the above equation, X is the N dimensional state

variable vector, P is an M dimensional vector of unknown parameters, F is the functional form for the derivatives of the state variables, and t is the independent variable, time.

The precise values for the parameters in Equation (1-1) are not known, and the values for some of the state variable initial conditions may also not be known.

Specified or Measured Inputs and Responses

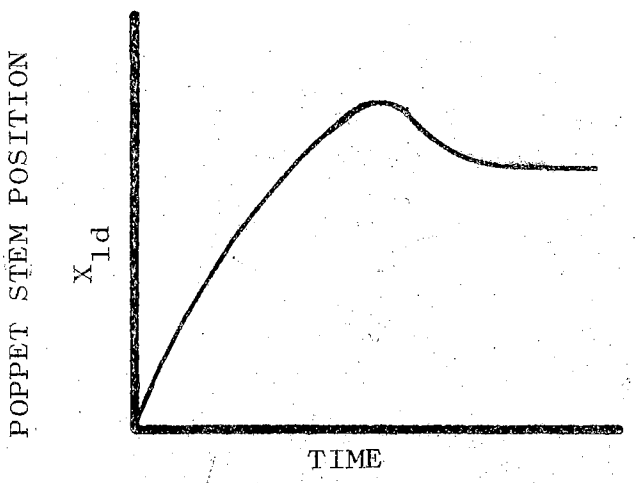
The inputs to Equation (1-1) need to be specified as functions of time. If the problem is an identification problem, the inputs to the physical system would be measured in the laboratory. If the problem is a system design problem, the input to the physical system and, hence, for the mathematical model must be specified.

The measured or desired responses for the mathematical equation must then be specified. For the poppet valve example, typical desired responses for state variables 1 and 3 (poppet position and chamber pressure) are shown in Figure 6.

Performance Index

Some measure must be set up to describe how "close" the response of the mathematical equation (1-1) is to conforming to the specified or measured response.

One standard performance index which is commonly used is the integral square error (ISE) which is defined as



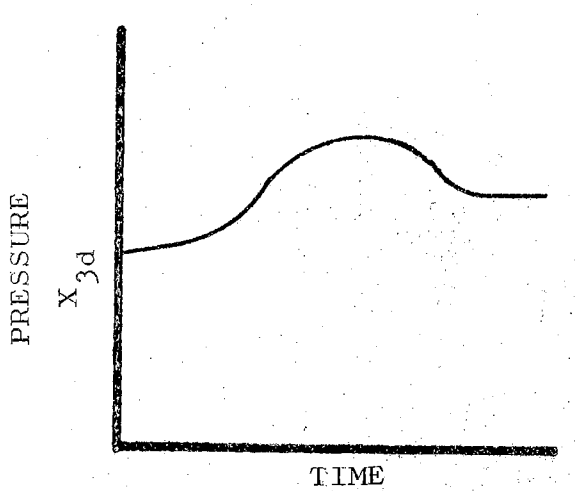


Figure 6. Typical Desired Response

follows for the poppet valve example:

$$PI = \int_{t_o}^{t_f} W_1 (X_1 - X_1 d)^2 + W_3 (X_3 - X_{3d})^2 dt. \qquad (1\text{-}2)$$

In the above equation, W_1 and W_3 are weighting factors, X_1 and X_3 are the responses of state variables 1 and 3 in Equation (1-1), and X_{1d} and X_{3d} are the desired state variable responses as shown in Figure 6.

The weighting factors, W_1 and W_3, must be specified by the user. If for instance, the weighting factors are chosen as $W_1 = 1.0$ and $W_3 = 0.0$, then the performance index depends only on the error which is present due to the amount that the mathematically predicted position of the poppet stem deviates from the desired position. If $W_1 = 0.0$ and $W_3 = 1.0$, then the performance index depends only on the error which is present due to the pressure predicted by the mathematical equation not agreeing with the desired pressure response. As a general rule, the weighting factors will be chosen so that their sum is unity. Hence, for this example,

$$W_1 + W_3 = 1.0. \qquad (1\text{-}3)$$

The other general rule to be used when choosing weighting factors is that the ratio of any two weighting factors should be inversely proportional to the square of the maximum expected error associated with each weighting factor. Hence, if the maximum expected errors in this example were $(X_1 - X_{1d})_{max} = .001$ and $(X_3 - X_{3d})_{max} = 10$,

the weighting factors should satisfy

$$\frac{W_1}{W_3} = \frac{(X_3 - X_{3d})^2_{max}}{(X_1 - X_{1d})^2_{max}} = \frac{(10)^2}{(.001)^2} = 10^8. \qquad (1\text{-}4)$$

If Equations (1-3) and (1-4) are solved simultaneously, the following values are obtained:

$$W_1 = .999$$

$$W_3 = 10^{-8}. \qquad (1\text{-}5)$$

Determination of Parameters

The system identification and modeling problem can now be summarized in the following short paragraph.

The parameter vector, P, and unknown initial conditions are to be determined so that the response of the system equation (1-1) will minimize the specified performance index. When these parameters and initial conditions have been determined, they represent the "best" values for the system equation according to the specified performance index. These parameters and initial conditions then represent the optimal "identification of the system" or the optimal values for system design.

Comparison of Non-Statistical Methods

In this section the following five methods of system identification are discussed: (1) Parameter Influence Coefficients, (2) Optimal Control, (3) Differential

Approximation, (4) Bacon's Method, (5) Quasilinearization, and (6) the method of seeking principal planes.

All of these methods, excepting Bacon's Method, determine the parameters for the state model, Equation (1-6), which will minimize a specified performance index, Equation (1-7).

$$\dot{X} = F(X, P, t) \tag{1-6}$$

$$PI = \int_{t_o}^{t_f} G(X, P, t)dt. \tag{1-7}$$

A special case of the above performance index is the integral error squared.

$$PI = \int_{t_o}^{t_f} (X_1 - X_{1d})^2 \, dt. \tag{1-8}$$

The term, X_{1d}, is the desired response for the first state variable.

Parameter Influence Coefficients

Meissenger (2) presents the basis for this powerful technique of parameter identification. It is based on the philosophy that if the gradient of the performance index with respect to the parameter vector can be determined, it will indicate the proper direction in which the parameter vector must be moved in order to reduce the performance index. Upon calculation of the gradient of the performance

index, it is realized that the parameter influence coefficients which are defined below are quite important.

$$\text{PARAMETER INFLUENCE COEFFICIENTS} = \frac{\partial X_i}{\partial P_j} \quad \begin{matrix} i = 1, 2, \ldots N \\ j = 1, 2, \ldots M \end{matrix} . \tag{1-9}$$

These parameter influence coefficients can be obtained by performing some mathematical manipulations and expanding the original state model.

The solution procedure is an iterative technique which utilizes the steepest descent. Each iteration requires the solution of 1+N+M+NM first order differential equations where N is the order of the state model and M is the number of parameters being optimized. This requires considerable computational time per iterative step.

Optimal Control Theory

Optimal control theory can be applied in a straightforward manner, to yield a set of 2(N+M) ordinary differential equations with split boundary conditions. These equations represent a set of necessary conditions which the optimal parameter vector must satisfy.

$$\dot{X} = F(X, P, t) \tag{1-10}$$

$$\dot{P} = 0$$

$$\dot{\lambda}_K = \frac{-\partial G}{\partial X_K} - \sum_{j=1}^{N} \lambda_j \frac{\partial F_j}{\partial X_K} \qquad K = 1, 2, \ldots N$$

$$\dot{\mu}_K = \frac{-\partial G}{\partial P_K} - \sum_{j=1}^{N} \lambda_j \frac{\partial F_j}{\partial P_K} \qquad K = 1, 2, \ldots M.$$

The boundary conditions which the above equations must satisfy are,

$$\begin{aligned} X(0) &= X_o \\ \lambda(T) &= 0 \\ \mu(0) &= 0 \\ \mu(T) &= 0. \end{aligned} \tag{1-11}$$

Any method such as gradient techniques, quasilinearization, or dynamical programming can be used to solve the set of differential equations. This is not a simple problem and considerable effort is necessary for finding a solution.

Optimal Parameters for Linear Systems

Bacon (3, 4) has presented a method which is applicable to constant coefficient linear systems

$$\begin{aligned} \dot{X} &= [A]X + [B]U \\ Y &= [C]X + [D]U. \end{aligned} \tag{1-12}$$

In the above equation [A], [B], [C], and [D] are constant coefficient matrices. However, each individual element of these matrices can be a function of the parameters, and the procedure which Bacon develops optimizes the parameters and not the coefficients. This is a definite

advantage since it is quite common for one parameter to be involved in several coefficients.

The principle handicap of Bacon's procedure is that the desired response, Y, and system input, U, must be specified as linear combinations of analytic functions. These analytic functions must be of the class which is obtainable from the solution of a linear constant coefficient homogeneous state model. This means that the inputs and outputs obtained from measured data must be fitted with analytic functions.

The iterative procedure which determines the optimal parameters does not require that the state model, Equation (1-12), be solved over a time interval. Thus, this method requires relatively little computational effort as compared to other methods.

Differential Approximations

The method of differential approximation as presented by Bellman, Kalaba, and Sridhar (5) utilizes the fact that Equation (1-1) can be rewritten as,

$$\dot{X} - F(X, P, t) = 0. \tag{1-13}$$

From this simple observation, it is evident that if $\mathring{X}$, and X, are known, then the correct parameter vector must minimize the following performance index.

$$PI = \int_0^T \langle \dot{X} - F(X, P, t), \dot{X} - F(X, P, t) \rangle \, dt. \tag{1-14}$$

In the above equation the symbols $<, >$ stand for the dot product of the two vectors.

The fact that the gradient of the performance index with respect to the parameter vector must be zero at the minimum point is utilized to obtain a general equation for determining the optimal coefficients. Bose (6) has developed this procedure for use with a discrete performance index. This is a powerful technique which does not require an iterative procedure for determining the optimal coefficients.

However, the differential approximation method requires that the entire state trajectory and the derivative of the state trajectory be known. This is an extremely large amount of information which is often quite difficult to obtain. This method requires the solution of a set of nonlinear algebraic equations if optimal parameters are to be determined. It often occurs that a change in one parameter will cause a change in several of the coefficients.

Quasilinearization

This approach as presented by Bellman, Kalaba, and Sridhar (5), and Allison (7) can be used to determine the parameters of the state model (1-1) which will minimize a general integral error squared performance index (1-2). This is accomplished by solving a sequence of linear models which approximate the original system. If this sequence of linear models converges to a final solution, the resulting

parameters are the optimal parameters for the original system model.

The major weakness of the method of quasilinearization is that the method may diverge rather than converge to the desired solution. A proof which is presented in Appendix C can be used to show that the method of quasilinearization may diverge from a desired parameter point regardless of how close to that point one starts if the mathematical response does not fit the desired response exactly.

Method of Seeking Principal Planes

This is a new method which is developed in this thesis. It is based upon the Taylor series expansion which is truncated after the second-order terms.

Comparison of Methods

Among the qualities which are of interest in comparing these methods are:

1. Is the method restricted to linear systems?
2. Does the method require knowledge of the entire state vector of the physical system, or is knowledge of just one of the system outputs sufficient?
3. Does the method require that a set of differential equations be integrated for each iterative step?

4. Can the method be used with data in graphical form?
5. Does the method require the solution of a system of differential equations with split boundary conditions?
6. Is the user free to specify the criteria which are used to determine the "goodness" of the parameters which are selected by the method?
7. Does the method determine the desired physical parameters or a set of coefficients for the mathematical equations?
8. Can limits be placed on the parameter values which the method selects?
9. Does the method converge from even a poor original guess of the actual system parameters?
10. Does the method converge when the parameter guesses become close to the optimal parameters?

The comparison of the various methods can most easily be presented in the form of a table which is given in Figure 7. An X has been placed over each undesirable characteristic.

From Figure 7 it is seen that the method of seeking principal planes has only one major weakness: It requires the solution of the entire state model over the time period of interest for each iterative step.

However, it has very good convergence characteristics. In Chapter II this method is developed for the problem

	Parameter Influence Coefficients	Optimal Control	Differential Approximation	Bacon's Method	Quasilinearization Method	Method of Seeking Principal Planes
Convergence characteristics when parameter guess is close to the optimal	~~POOR~~	Solution method unspecified	Excellent	Reasonable	?	Excellent
Stability problems if initial parameter guess is not close enough	NO	Solution method unspecified	NO	~~YES~~	~~YES~~	NO
Limits can be placed on parameter values	YES	~~NO~~	YES	YES	~~NO~~	~~NO~~
Can adjust parameters rather than coefficients	YES	YES	YES	YES	YES	YES
Type of performance index allowed	GENERAL	GENERAL	~~SPECIAL~~	~~SPECIAL~~	~~SPECIAL~~	GENERAL
Requires solution of nonlinear two-point boundary value problem	NO	~~YES~~	NO	NO	NO	NO
Desired response and input can be in graphical form	YES	YES	YES	~~NO~~	YES	YES
Requires solution of entire state model for each iterative step	~~YES~~	Solution method unspecified	NO	NO	~~YES~~	~~YES~~
Requires knowledge of entire state vector of physical system	NO	NO	~~YES~~	NO	NO	NO
Restricted to linear systems	NO	NO	NO	~~YES~~	NO	NO

Figure 7. Comparison of the Various Methods of Parameter Identification
(An X has been placed over each of the undesirable characteristics.)

of determining the minimum of a function of several variables. Chapter III expands the method so it can be used to work two-point boundary value problems. Chapter IV uses the method of seeking principal planes directly on the problem of system design or identification, and Chapter V presents several approaches for minimizing problems which arise when the dynamical system is discontinuous.

CHAPTER II

MINIMIZATION OF A FUNCTION OF SEVERAL VARIABLES

There are two basic methods for attacking the problem of determining optimal parameters for a dynamical system. The first approach which is called the direct method adjusts the free parameters until a minimum of the performance index is obtained. The second method which is called the indirect method and utilizes the fact that the gradient vector of the performance index must be zero at an extremum point. This provides a set of necessary conditions which must be satisfied. Indirect solution approaches then attempt to satisfy these necessary conditions.

Both the direct and indirect methods of approach for determining optimal parameters of a dynamical system can be restated as a problem of minimization of a function of several variables. Hence, it is seen that the problem of determining the minimum of a function of several variables is important. Chapter III discusses the indirect method; Chapter IV discusses direct methods.

The first section of this chapter summarizes briefly the standard approaches for the minimization problem. Next,

a new procedure which is called the method of seeking principal planes is presented. The third portion of this chapter is devoted to a discussion of methods for solving systems of algebraic equations which is an inherent part of the indirect procedures. Finally, several example problems which have been solved by the method of seeking principal planes are presented and compared with results obtained by other methods to illustrate the method's versatility.

Standard Methods for Determining a Minimum of a Function of Several Variables

An important problem both from a mathematical and practical viewpoint is to determine the minimum of a scalar function of several variables. The independent variables will be denoted by the vector X which is of dimension M, and the function to be minimized is

$$Y(X). \tag{2-1}$$

Extensive work has been performed to determine methods for minimizing a scalar function. There are many methods which have been devised for special scalar functions, but three methods of particular interest which handle general scalar functions are: (1) gradient techniques, (2) the Newton-Raphson method, and (3) the conjugate gradient method.

Gradient techniques normally proceed in the negative gradient direction. The distance traveled in the negative

gradient direction depends upon which method is being used, but it is quite common to travel in the negative gradient direction until a local minimum is reached. The gradient is then reevaluated and one proceeds in the new negative gradient direction to a local minimum. This is continued until one is sufficiently close to the desired minimum point. The principle shortcoming of gradient techniques is that they may be very slow in converging to a desired minimum. Another problem associated with gradient techniques is the determination of the initial estimate of how far one should proceed in the negative gradient direction as one searches for the local minimum.

The Newton-Raphson method as is described by Wilde and Beightler (10) utilizes the expansion of the scalar function in a Taylor series about some initial point, X.

The gradient and matrix of second partial derivatives of the scalar function are defined as

$$Y_X = \left[\frac{\partial Y}{\partial X_1}, \frac{\partial Y}{\partial X_2}, \cdots, \frac{\partial Y}{\partial X_M}\right]$$

$$Y_{XX} = \begin{bmatrix} \partial\dfrac{\partial Y}{\partial X_1}\Big/\dfrac{\partial X_1}{\partial X_1}, & \partial\dfrac{\partial Y}{\partial X_1}\Big/\dfrac{\partial X_1}{\partial X_2}, & \cdots, & \partial\dfrac{\partial Y}{\partial X_1}\Big/\dfrac{\partial X_1}{\partial X_M} \\ \vdots & & & \\ \partial\dfrac{\partial Y}{\partial X_M}\Big/\dfrac{\partial X_M}{\partial X_1}, & \partial\dfrac{\partial Y}{\partial X_M}\Big/\dfrac{\partial X_M}{\partial X_2}, & \cdots, & \partial\dfrac{\partial Y}{\partial X_M}\Big/\dfrac{\partial X_M}{\partial X_M} \end{bmatrix}.$$

The Taylor series expansion then becomes

$$Y(X) = Y(X_0) + Y_X(X_0)[X - X_0] + \frac{1}{2!}[X - X_0]^T$$

$$Y_{XX}(X_0)[X - X_0] + \ldots \qquad (2\text{-}2)$$

It is assumed that the new function which is obtained by truncating the Taylor series after the first two terms describes the scalar function accurately. Then one looks at the required conditions for extremizing the local function which has been obtained by truncating the Taylor series. The result is that the new estimate for the minimum of the scalar function is

$$X_{new} = X_0 - Y_{XX}(X_0)^{-1}Y_X(X_0)^T. \qquad (2\text{-}3)$$

In Equation (2-3) the superscripts -1 and T, respectively, represent the matrix inverse and matrix transpose.

The assumption that the truncated Taylor series expansion describes the scalar function accurately, in general, is not valid globally. This implies that if one is not close to the desired minimum, then the new point which is predicted by Equation (2-3) may not be a good estimate. The Newton-Raphson method will often diverge if one is not close enough to a desired minimum.

Another shortcoming of the Newton-Raphson method is that it requires evaluation of the gradient of the function and the matrix of second partial derivatives. This can amount to significant computational effort.

The feature of the Newton-Raphson method which is very desirable is that if one is sufficiently close to the desired minimum, then the method is quadratic in convergence.

Another method for minimizing a scalar function which has received recent attention is called the conjugate gradient method. Fletcher and Reeves (8) have demonstrated the computational feasibility of this method.

In the conjugate gradient method, the first step is normally taken in the negative gradient direction. It is required that a local minimum along the negative gradient direction be located. Then, a new direction is calculated which might be termed a deflected negative gradient direction. This deflected gradient direction is not the same as the true gradient direction, and this is the primary difference between conjugate gradient and normal gradient procedures.

The deflected negative gradient direction is such that if you proceed a small enough distance in this direction, you will obtain a smaller value for the scalar function which is to be minimized.

The next step of the procedure is to determine a local minimum along the deflected negative gradient direction. The iterative procedure of determining (1) a new deflected gradient direction and (2) a minimum along this direction is continued until one is sufficiently close to a desired minimum of the scalar function.

The basic reason that this method is so powerful is that it is quadratic in convergence. This means that once you are close to a desired minimum value, the method will converge very rapidly. Hence, the method of conjugate gradients eliminates the principal fault of gradient techniques, but it still does not remedy the selection of an initial estimate of the distance to be traveled along the selected direction.

Method of Seeking Principal Planes

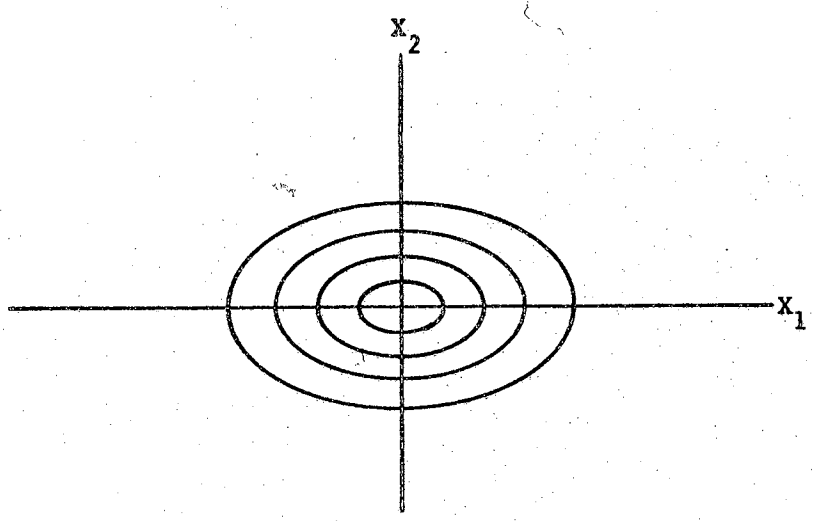
A good amount of insight into the scalar minimization problem can be obtained by investigating the scalar function

$$Y = C_1X^2_1 + C_2X^2_2$$

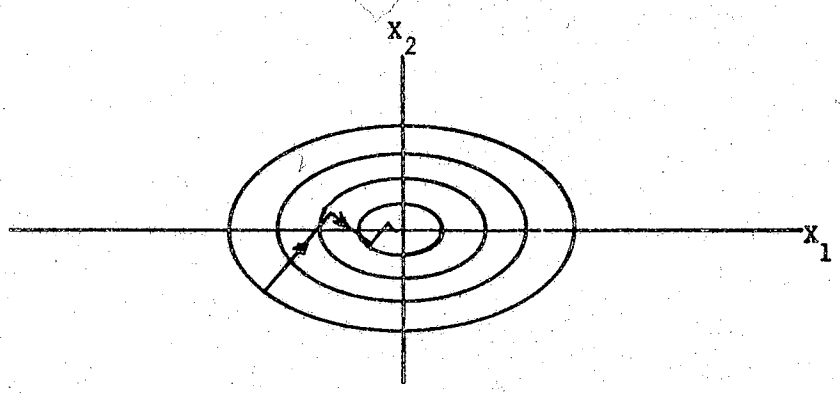
where C_1 and C_2 are both positive numbers. In Figure 8 contours of constant value for the scalar function have been drawn. They are concentric ellipses, and the amount of eccentricity is governed by the relative magnitudes of C_1 and C_2.

It is informative to investigate how it would be possible to reach the exact minimum of this function by going successively in negative gradient directions.

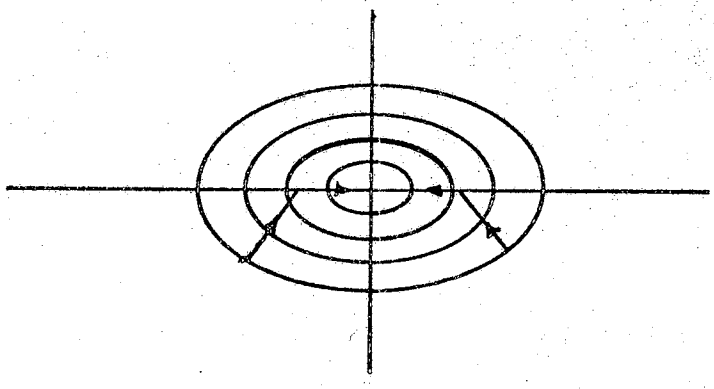
It is first noted that the normal gradient technique will never reach the minimum of the scalar function exactly. Starting from an arbitrary point, the standard gradient technique will zigzag back and forth about one of the principal axes, but it will never reach the exact



A. Contours of Constant Value for Y



B. Typical Trajectory for Gradient Technique



C. Trajectories Which Proceed Sequentially in Negative Gradient Direction and Which Reach the Exact Minimum

Figure 8. Minimization of $Y = C_1 X^2_1 + C_2 X^2_2$

minimum point. This is shown in part B of Figure 8.

In part C of Figure 8, several trajectories which proceed sequentially in negative gradient directions and which reach the exact minimum of the scalar function are shown. In general, two steps are required to reach the minimum for this particular problem.

The minimization procedure presented in this section is called the method of seeking principal planes. For the two-dimensional minimzation problem in Figure 8, it is evident that the points at which the negative gradient direction intersects the axes are important; but the figure does not indicate why, in general, the points at which the negative gradient direction intersects the principal planes are important.

The three-dimensional minimization problems given by

$$Y = C_1X^2_1 + C_2X^2_2 + C_3X^2_3,$$

will exemplify why the points of intersection of the principal planes are important. C_1, C_2, and C_3 are positive numbers in the preceding equation.

A contour of constant value for Y is shown in Figure 9. In general, constant value contours for Y will be ellipsoids, and the amount of eccentricity is governed by the relative magnitudes of C_1, C_2, and C_3.

It will now be demonstrated that by proceeding three steps in the negative gradient direction the exact minimum point can be reached.

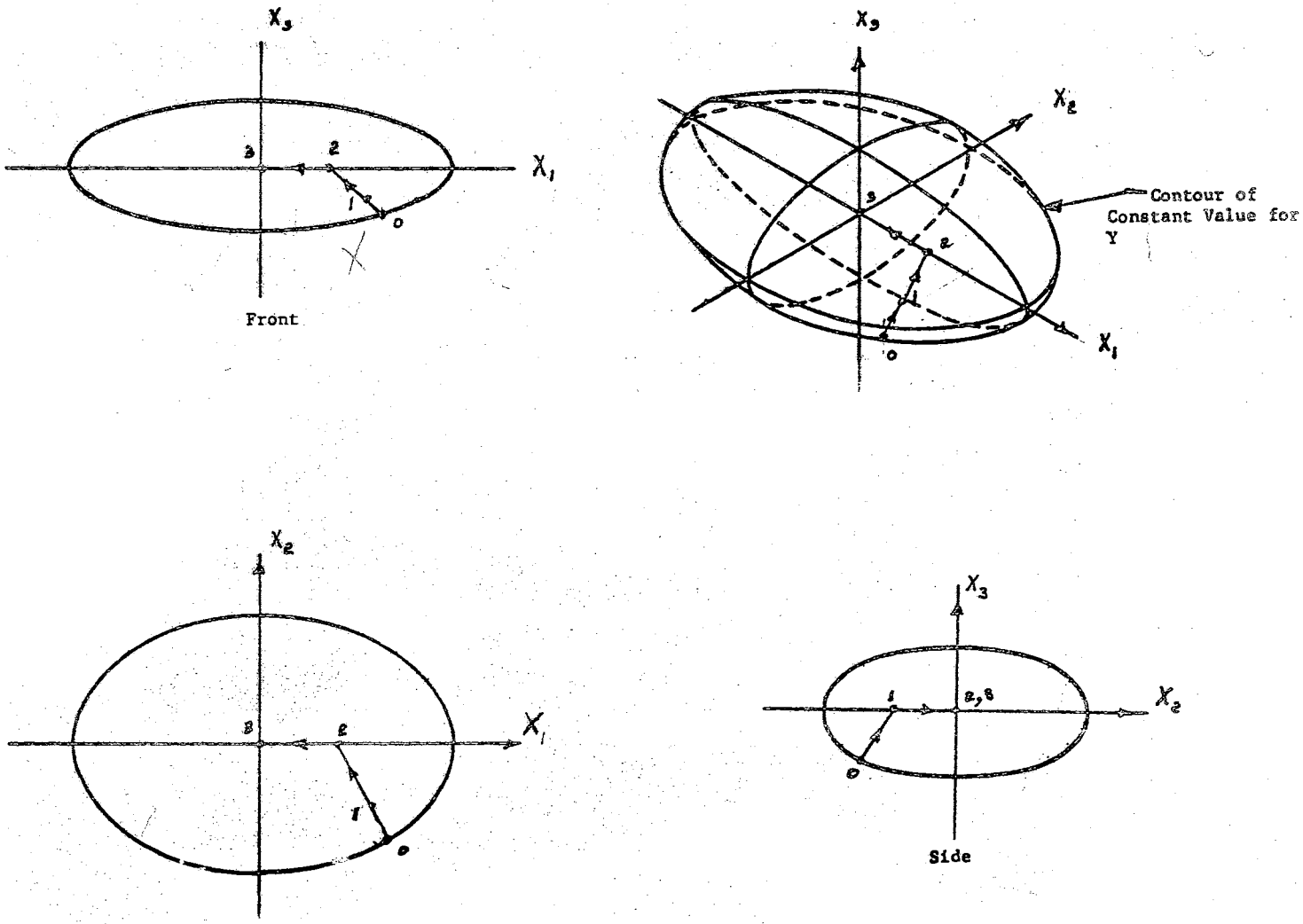


Figure 9. Minimization of $Y = C_1 X^2_1 + C_2 X^2_2 + C_3 X^2_3$ by Method of Seeking Principal Planes

An arbitary point denoted by 0 in Figure 9 will be used as the starting point. The point 1 of Figure 9 is the point at which the negative gradient direction from point 0 intersects the plane which contains the axes X_1 and X_2. The point 2 of Figure 9 is the point at which the negative gradient direction from point 1 intersects the plane which contains axes X_1 and X_3. Similarly, point 3 of Figure 9 is the point at which the negative gradient direction from point 2 intersects the plan which contains axes X_2 and X_3.

The mathematical development which indicates the distance to the intersection of the principal planes in negative gradient direction will now be presented.

The first step of the mathematical development is to expand the scalar function in a Taylor series such as in Equation (2-2).

Throughout the rest of this development, it will be assumed that all of the third partial derivatives of the scalar function are continuous in the neighborhood of X_0. This implies that the matrix of second partial derivatives will be symmetric, and the scalar function can be approximated locally by truncating the Taylor series as

$$Y_{loc}(X) = Y(X_0) + Y_X(X_0)[X - X_0] + \frac{1}{2!}[X - X_0]^T Y_{XX}(X_0)[X - X_0]. \qquad (2\text{-}4)$$

The gradient vector of the local approximation to the scalar function is

$$Y_{X_{loc}}(X) = Y_X(X_0) + [X - X_0]^T Y_{XX}(X_0). \qquad (2\text{-}5)$$

The question which must now be answered is: How far in the negative gradient direction should one travel in proceeding from the present position of the independent variable, X_i, to the new independent variable, X_{i+1}. Note that the present position, X_i, will be equal to X_0 on the very first step, but in general X_i will not be the same as X_0. Also, the subscript refers to the iteration number; that is, X_i implies the X vector at the i^{th} iteration.

In mathematical terms, this amounts to determining the magnitude of K in the following equation:

$$X_{i+1} = X_i - K\, Y_X^T (X_i). \qquad (2\text{-}6)$$

An excellent approximation of $Y_X(X_i)$ is $Y_{X_{loc}}(X_i)$ which is given by Equation (2-5) provided that X_i is sufficiently close to X_0. If it is assumed that they are exactly the same, Equation (2-5) can be substituted into Equation (2-6); and the resulting expression is

$$X_{i+1} = X_i - K[Y_X(X_0) + (X_i - X_0)^T Y_{XX}(X_0)]^T. \qquad (2\text{-}7)$$

The predicted difference between the value of the scalar function at the point X_{i+1} and the point X_i provides valuable information for determining the magnitude of K in Equation (2-7). This predicted difference can be obtained from Equations (2-7) and (2-4), and the resulting

equations are

$$X_{i+1} = X_i = -K[Y_X(X_0) + [X_i - X_0]^T Y_{XX}(X_0)]^T \qquad (2\text{-}8)$$

and

$$Y_{loc}(X_{i+1}) - Y_{loc}(X_i) = Y_X(X_0)\,[X_{i+1} - X_i]$$

$$+ \tfrac{1}{2}[X_{i+1} - X_0]^T Y_{XX}(X_0)[X_{i+1} - X_0] \qquad (2\text{-}9)$$

$$- \tfrac{1}{2}[X_i - X_0]^T Y_{XX}(X_0)[X_i - X_0].$$

Several coordinate transformations need to be performed so that Equations (2-8) and (2-9) can be better interpreted. These coordinate transformations involve the eigenvalues and eigenvectors of the matrix of second partial derivatives. Athans and Falb (9) show that all of the eigenvalues will be real numbers since the matrix of second partial derivatives is real and symmetric.

In general, there will be P positive eigenvalues, N negative eigenvalues, and Z zero eigenvalues. The eigenvalue matrix, λ, is arranged so that the positive eigenvalues are the first to appear on the diagonal, the negative eigenvalues appear next on the diagonal, and the zero eigenvalues appear last on the diagonal. Further, it is required that $\lambda_1 > \lambda_2 > \ldots \lambda_P$.

Throughout the rest of this paper, λ_i will mean the i^{th} eigenvalue which is located on the diagonal of the eigenvalue matrix,

$$\lambda = \begin{bmatrix} \lambda_1 & & & & & & \\ & \ddots & & & & & \\ & & \lambda_P & & & & \\ & & & \lambda_{P+1} & & & \\ & & & & \ddots & & \\ & & & & & \lambda_{P+N} & \\ & & & & & & 0 \\ & & & & & & & \ddots \\ & & & & & & & & 0 \end{bmatrix}, \qquad (2\text{-}10)$$

The eigenvector matrix, E, which corresponds to the eigenvalue matrix will be partitioned as

$$E = [E_1, \ldots, E_P, E_{P+1}, \ldots, E_{P+N}, E_{P+N+1}, \ldots E_M]. \qquad (2\text{-}11)$$

In Equation (2-11) and throughout the rest of this chapter, E_i refers to the i^{th} column of the eigenvector matrix, E. Athans and Falb (9) demonstrate that the eigenvector matrix satisfies the following relationship:

$$E^T E = I \qquad (2\text{-}12)$$

and

$$E^T Y_{XX} (X_0) E = \lambda. \qquad (2\text{-}13)$$

I is the identity matrix in Equation (2-12).

The first coordinate transformation which will be performed is a translation of the coordinate system by the amount Z,

$$W = X - Z, \qquad (2\text{-}14)$$

The vector Z is defined by the following set of equations:

$$C_j = E_j^T \, Y_X^T(X_O) \qquad j = 1, 2, \ldots, p + N \qquad (2\text{-}15)$$

$$C_j = 0 \qquad j = p+N+1, \ldots, M \qquad (2\text{-}16)$$

$$Z = -EC + X_O. \qquad (2\text{-}17)$$

The amount of the coordinate translation which is given by Equation (2-14) at first appears to be quite confusing, but it should be noted that if the matrix of second partial derivatives has no zero eigenvalues, then the resulting translation is to the extremum point of the local approximation for the scalar function, Equation (2-4). This will now be demonstrated.

If Equation (2-13) is premultiplied by E, post multiplied by E^T, and then inverted, the equation which results is

$$Y_{XX} \, (X_O)^{-1} = E \, \lambda^{-1} \, E^T. \qquad (2\text{-}18)$$

Equations (2-17) and (2-18) can be combined and simplified to

$$Z = -Y_{XX} \, (X_O)^{-1} \, Y_X(X_O)^T + X_O$$

which is the stationary point of Equation (2-4).

A new coordinate system will now be introduced by rotating the W coordinate. The resulting coordinate system e is defined as

$$e = E^T W = E^T[X - X_0] + C. \tag{2-19}$$

If the following definitions are made,

$$e_0 = e\ (X_0) = C, \tag{2-20}$$

$$e_i = e\ (X_i) = E^T[X_i - X_0] + C \tag{2-21}$$

and

$$e_{i+1} = e\ (X_{i+1}) = E^T[X_{i+1} - X_0] + C; \tag{2-22}$$

then it follows that

$$X_i - X_0 = E[e_i - C] \tag{2-23}$$

$$X_{i+1} - X_0 = E[e_{i+1} - C] \tag{2-24}$$

and

$$X_{i+1} - X_i = E[e_{i+1} - e_i]. \tag{2-25}$$

The simplification of the equation which results from the substitution of Equations (2-23) through (2-25) into Equation (2-9) yields

$$Y_{loc}(X_{i+1}) - Y_{loc}(X_i) = [Y_X(X_0)E - C^T\lambda]$$
$$[e_{i+1} - e_i] + \tfrac{1}{2}\, e_{i+1}^T \lambda\ e_{i+1} - \tfrac{1}{2}\, e_i^T \lambda\ e_i. \tag{2-26}$$

The new value in the e coordinate system as predicted by going in the negative gradient direction can be found by substituting Equations (2-23) and (2-25) into Equation (2-8).

The change of each of the individual components of the e coordinate system can be written by using a double subscript notation. The first subscript indicates the iteration number, and the second subscript indicates the component of the e coordinate vector. The resulting equations are:

$$e_{i+1,j} = [1 - K \lambda_j] e_{i,j} \quad ; \quad j = 1, 2, \ldots, P+N \tag{2-27}$$

and

$$e_{i+1,j} = e_{i,j} - K E_j^T Y_X(X_0) \quad ; \quad j = P+N+1, \ldots, M. \tag{2-28}$$

The substitution of Equations (2-27), (2-28), (2-15), and (2-16) into Equation (2-26) will yield the final desired form which predicts the resulting magnitude change of the scalar function in going from the i^{th} iteration point, X_i, to the point X_{i+1} in the negative gradient direction. The resulting equation is

$$Y_{loc}(X_{i+1}) - Y_{loc}(X_i) = - \frac{K}{2} \sum_{j=1}^{P+N} [(2 - K\lambda_j) \lambda^2_j \, e^2_{i,j}] - K \sum_{j=P+N+1}^{M} [Y_X(X_0) E_j]^2 . \tag{2-29}$$

The systematic investigation of Equations (2-27), (2-28), and (2-29) enables one to envision an algebraic minimization procedure which is very general.

From Equation (2-27), it is noted that the first P + N components of the e coordinate vector will never move from the value of zero once they become zero. This is true

regardless of the distance traveled in the gradient direction.

The method utilized for proceeding to the minimum of an algebraic function should be such that after each step in the negative gradient direction one will be at a smaller value for the scalar function. Hence, the magnitude chosen for K should be such that the predicted change of the scalar function, Equation (2-29), will be negative.

On the very first step in the negative gradient direction X_i will equal X_0, the point at which the gradient vector and the matrix of second partial derivatives was evaluated. By setting $K = 1/\lambda_1$, the change predicted by Equation (2-29) is negative since each of the terms $(2 - K\lambda_j)$ will be positive. Equation (2-27) states that the first component of the e coordinate should become equal to zero on this first iterative step.

On the second iterative step, one proceeds in the predicted negative gradient direction, Equation (2-5), with the magnitude of K set at $K = 1/\lambda_2$. The first component of the e coordinate system was set equal to zero on the first iterative step, and Equation (2-27) states that the second component of e should become equal to zero on the second iterative step. Equation (2-29) predicts a negative change of the scalar function on this second iterative step since $(2 - K\lambda_j)$ is positive for $j = 2, 3, \ldots, P$ and since component one of the e vector should be zero.

By proceeding in the general manner outlined in the

previous paragraphs, the first P-1 components of the e vector will be zero on the P^{th} iteration, and Equation (2-29) predicts a negative change of the scalar function on the P^{th} iteration since $(2-K\lambda_j)$ is positive for $j = P, \ldots, P+N$.

After P iterations the first P components of the e coordinate system should be zero. Hence, on the $P+1^{th}$ iteration, K should be chosen as any positive value instead of $1/\lambda_{P+1}$ which would be a negative value. Equation (2-29) indicates that the magnitude of the predicted change of the scalar function will continue to increase negatively as K becomes a larger positive value. This due to the fact that the quantity $(2-K\lambda_j)$ will be positive for $j = P+1, \ldots, P+N$, since all of the corresponding values for the λ_j will be negative. Hence, on the $P+1^{th}$ iteration, the value of K will be increased until the actual value of scalar function ceases to decrease. This will probably occur due to the fact that the change as predicted by Equation (2-29) is based upon the truncated Taylor series approximation for the function in question. In general, this approximation is not valid in a global sense. It is, however, a good approximation in the vicinity of X_0.

Before presenting a flow chart of a minimization procedure based upon the equations that have thus far been derived, there are still two issues which should be clarified. First, it is desirable to understand the significance of the transformations which were performed in order to

obtain Equation (2-19). Second, it is informative to demonstrate that the method of principal planes will tend to follow a valley to the desired minimum. The second statement will be verified by the solution of the example problem called Rosenbrock's curved valley in the example problem section.

To better understand the coordinate transformations, it will be informative to substitute Equations (2-15), (2-16), and (2-23) into Equation (2-4). The local approximation for the scalar function becomes

$$Y_{loc}(X_i) = Y(X_0) - \tfrac{1}{2}\sum_{j=1}^{P+N} \frac{(Y_X(X_0)E_j)^2}{\lambda_j} + \sum_{j=P+N+1}^{M} Y_X E_j e_{i,j} + \tfrac{1}{2}\sum_{j=1}^{P+N} \lambda_j e^2_{i,j}. \tag{2-30}$$

If, in addition, the point, X_0, is such that the second partial matrix has all positive eigenvalues, then the inverse of Y_{XX} exists and Equation (2-30) will simplify to

$$Y_{loc}(X_i) = Y_{loc}(X_{ext}) + \tfrac{1}{2}\sum_{j=1}^{P} \lambda_j e^2_{i,j}. \tag{2-31}$$

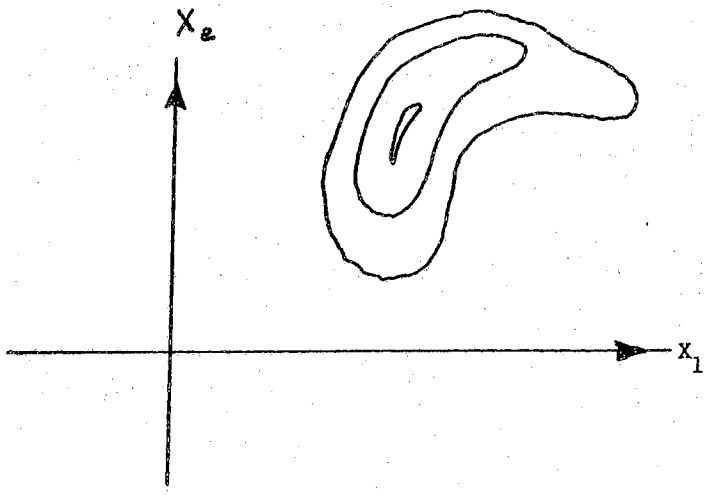
In Equation (2-31), the term $Y_{loc}(X_{ext})$ is the value of Equation (2-4) evaluated at its extremum point. Hence, under the assumption that all of the eigenvalues are positive, the truncated Taylor series approximation for the scalar

function in the vicinity of X_0 is an ellipsoid in M dimensional space. This is shown in Figure 10 for the case of M = 2.

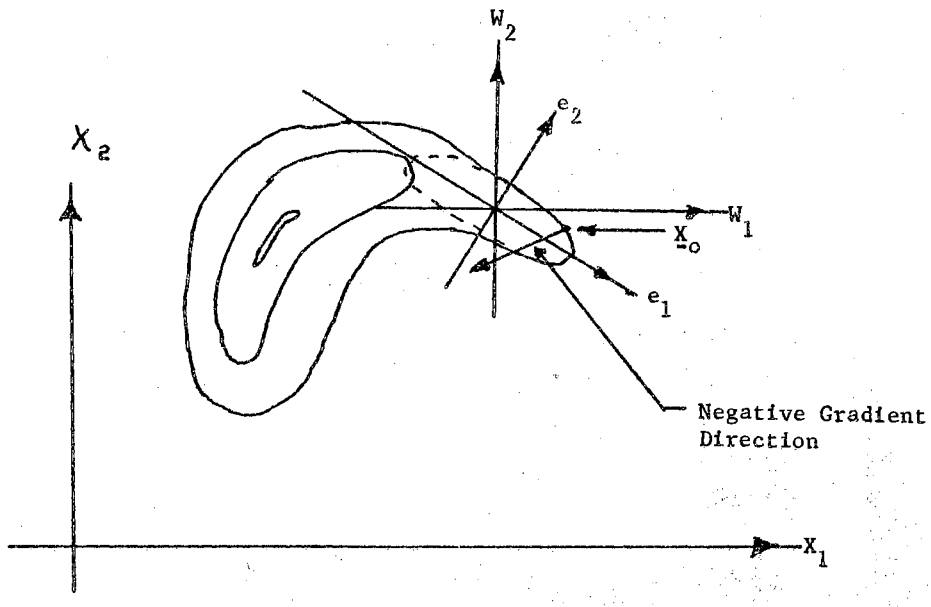
In part A of Figure 10, lines of constant contours for the hypothetical function under consideration have been drawn. The scalar function has a valley which curves from the extremum point upward and to the right. In part B of Figure 10, a line of constant countour that is predicted by the truncated Taylor series is shown as a dotted line. The magnitude predicted for the scalar function along this line of constant contour is $Y(X_0)$, and the predicted constant contour line is an ellipse.

The first coordinate transformation that was performed, Equation (2-14), was a transformation to the center of the ellipse which approximates the constant contour line at the point X_0. These axis are denoted by W_1 and W_2 in part B of Figure 10. The next coordinate transformation which was performed was to rotate the axes as indicated by Equation (2-19). From Equation (2-31), it is known that the e coordinate system corresponds to the major axes of the ellipsoids which approximate the lines of constant contour for the scalar function. In part B of Figure 10, the e_1 and e_2 coordinates represent this new coordinate system and they correspond to the major and minor axes of the ellipse.

It is noted that the constant contour line predicted by the Taylor series in part B of Figure 9 is a good approximation in the vicinity of X_0. However, the approximation



A. Contours of Constant Magnitude for Hypothetical Function



B. Diagram showing the co-ordinate transformations. The dotted line is the truncated Taylor series approximation for the constant contour line passing through X_o.

Figure 10. Figure Demonstrating Meaning of the Coordinate Transformations

for the scalar function is not valid globally since the extremum point as predicted by the approximation is at the point where the axes e_1 and e_2 intersect. The true extremum point for the hypothetical function is located to the left and below this point.

The e_1 axis runs essentially parallel to the valley of the hypothetical function in the neighborhood of X_0. For this particular problem, if one proceeds in the negative gradient direction to the point of intersection of the e_1 axis, one will be located in the immediate neighborhood of the true valley.

A desirable characteristic of a scalar minimization computational procedure is that it be able to locate a valley of the scalar function and follow the valley to the minimum. This seems to be the charactersitic of the method of seeking principal planes.

The flow chart for the method of seeking principal planes is presented in Figure 11. It indicates that the first step in the minimization problem is to determine the value of the scalar function, the gradient vector, and the matrix of second partial derivatives at the initial point, X_0.

The eigenvalues of the matrix of second partial derivatives are then determined.

Equation (2-7) gives the values of the independent variables by sequentially setting $K = 1/\lambda_i$, $i = 1, \ldots, P$. It is necessary after each of these steps in the local

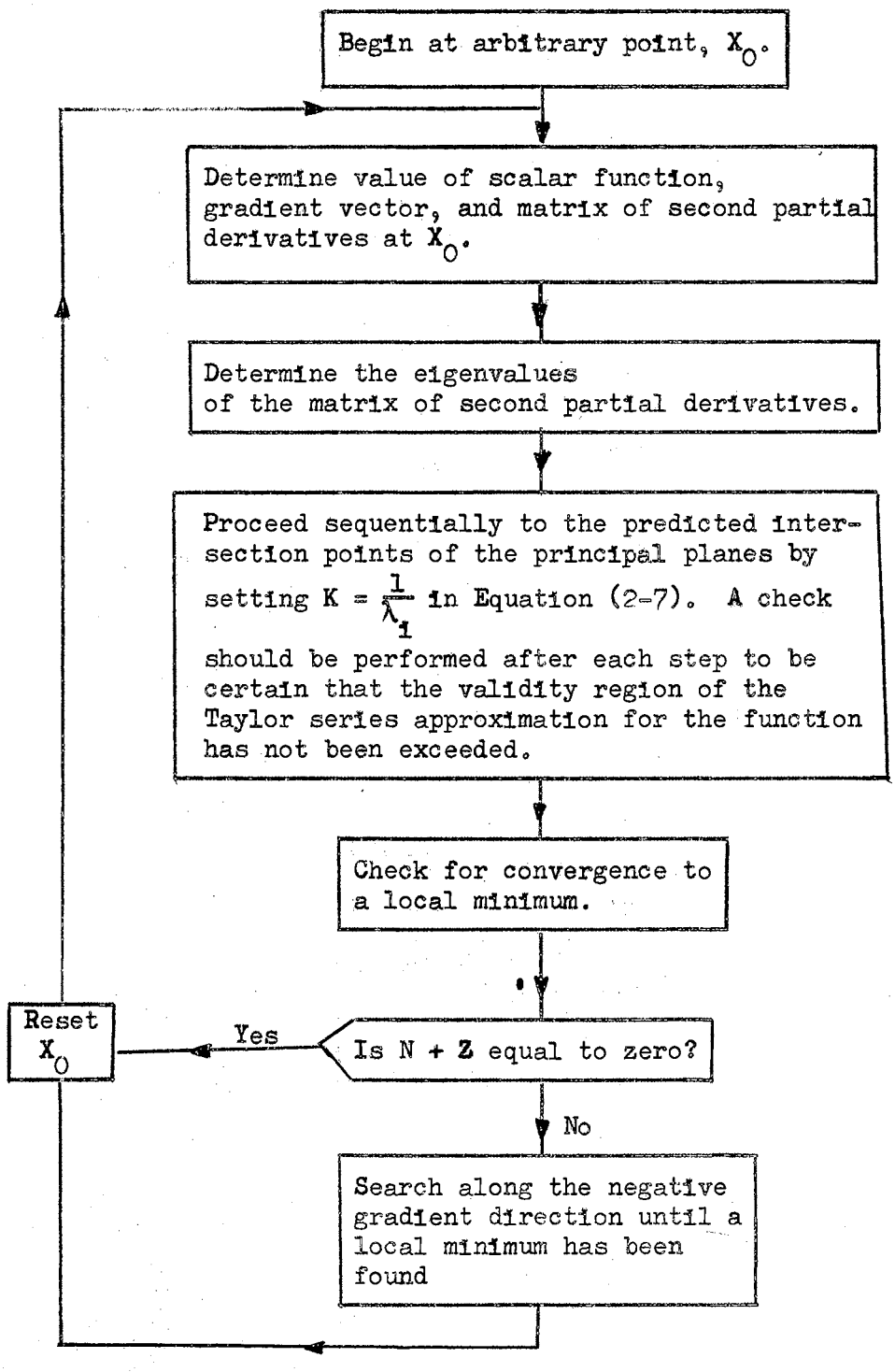


Figure 11. Flow Chart for the Minimization of a Scalar Function by the Method of Seeking Principal Planes

negative gradient direction to determine if the region of validity of the Taylor series approximation has been exceeded. If the scalar function increases after one of the steps or if the actual change of the scalar function is significantly different from the predicted change of the scalar function, the validity region has been exceeded.

A check for convergence is performed after having successfully proceeded P sequential steps in the negative gradient direction. A standard method for checking for convergence to a local minimum to determine if the magnitude of the gradient vector is equal to zero.

The next block in Figure 11 indicates a logic check to determine if the matrix of second partial derivatives has some zero or negative eigenvalues. If N plus Z is equal to zero, X_0 is reset to the present value of the independent vector and the minimization algorithm is reinitiated. When N plus Z is not equal to zero, a search is performed along the negative gradient direction until a local minimum is located. The value of X_0 is reset to the value of the independent variable at the local minimum, and the minimization procedure is reinitiated.

The method of seeking principal planes can be used for solving a system of algebraic equations. The insight obtained by studying this problem helps to explain why existing methods for solving two-point boundary value problems and for identifying the parameters of dynamical systems have not been completely satisfactory. Chapters III and IV will

refer to some of the results presented in the following section.

Solution of Systems of Algebraic Equations

The simplest set of algebraic equations which exist is the linear problem given by

$$AX = b. \tag{2-32}$$

In the above equation, A is an M x M matrix of constants; X is a vector of independent variables of dimension M; and b is a vector of constants of dimension M.

In order to formulate Equation (2-32) as a minimization problem, the following scalar function is defined:

$$Y(X) = \tfrac{1}{2}[AX - b]^T W[AX - b]. \tag{2-33}$$

In the above equation, W is a symmetric positive definite M x M matrix. It is not uncommon for W to be chosen as the identity matrix.

Since Equation (2-33) is a quadratic equation, the Taylor series which utilizes the first two terms will describe it exactly. This means that the method of principal planes as described in the previous section will be an exact method for the solution of the minimum of Equation (2-28) if roundoff error within the digital computer is ignored.

The equation for the gradient and matrix of second partial derivatives of Equation (2-33) are:

$$Y_X(X) = -b^T WA + X^T A^T W A \qquad (2\text{-}34)$$

and

$$Y_{XX}(X) = A^T W A. \qquad (2\text{-}35)$$

One good characteristic of the method of seeking principal planes is that if a solution for Equation (2-33) exists it will be found even in the situation where the matrix A is singular. If the matrix A is singular, there exists either no solution or an infinite number of solutions to Equation (2-33). When the solution is not unique, the solution which is found by the method of seeking principal planes depends upon the initial guess which the user provides for independent variable vector, X. It should be noted that if the user does not know a good approximation for the solution to Equation (2-33), the initial guess of X = 0 can always be used. In the situation where A is singular and Equation (2-33) does not have an exact solution, the method of seeking principal planes will provide a solution which minimizes Equation (2-33). If W is the identity matrix, this will be the best fit in the least square error sense.

Since the method of seeking principal planes is able to provide a solution even in the situation where A is singular, it is reasonable that this method should be a good solution technique for cases in which A is ill conditioned.

The independent variable which minimizes Equation (2-33) provides an approximate solution for Equation (2-32)

in the overconstrained case. This is the situation where A is a R X M constant matrix; X is the independent vector of M dimension; b is a constant vector of dimension R; and R is larger than M. This set of equations does not have an exact solution except in special cases. If the matrix W is the identity matrix, the solution yielded by the method of seeking principal planes will be a "best" solution in the least square error sense.

In engineering and mathematical analysis, the need for solving a set of simultaneous nonlinear algebraic equations often arises. In mathematical terms, this means that it is necessary to determine a M dimensional independent vector, X, such that the M dimensional vector valued function, G, becomes equal to the zero vector.

$$G(X) = 0. \tag{2-36}$$

This problem can be treated in a manner similar to that used for the solution of linear algebraic equations. Hence, the scalar function,

$$Y(X) = \frac{1}{2} G(X)^T G(X), \tag{2-37}$$

is formed.

Since Equation (2-37) can never be less than zero, and the only points at which it can be exactly equal to zero are solution points, a minimization procedure can be used to determine the solution for Equation (2-36).

The first and second partial derivatives for the scalar

function need to be formulated. To write these equations, the following definitions are needed.

$$G_X(X) = \begin{bmatrix} \frac{\partial G_1}{\partial X_1} & \cdots & \frac{\partial G_1}{\partial X_M} \\ \vdots & & \vdots \\ \frac{\partial G_M}{\partial X_1} & \cdots & \frac{\partial G_M}{\partial X_M} \end{bmatrix} \tag{2-38}$$

$$B_j = \left[\frac{\partial}{\partial X_j} G_X(X)^T\right] G; \quad j = 1, 2, \ldots, M \tag{2-39}$$

$$B = [B_1, \ldots, B_M]. \tag{2-40}$$

The gradient vector and matrix of second partial derivatives then become

$$Y_X(X) = G(X)^T G_X(X) \tag{2-41}$$

and,

$$Y_{XX}(X) = G_X(X)^T G_X(X) + B. \tag{2-42}$$

Equations (2-37), (2-41), and (2-42) are the equations which are needed to use the method of seeking principal planes.

From Equation (2-39), it is noted that if one is quite close to the solution for Equation (2-36), the second half of Equation (2-42) will approach the zero matrix. Hence, in the neighborhood of the solution for the nonlinear set of

algebraic equations, the second partial matrix can be approximate by the simpler expression,

$$Y_{XX}\ \underset{\text{Approx}}{} = G_X^T\ G_X. \qquad (2\text{-}43)$$

The reason for formulating Equation (2-43) is to demonstrate how the method of seeking principal planes is related to Gauss's method for least squares analysis as presented in Wilde and Beightler (10), and how it is related to the Newton-Raphson method as presented by Bellman and Kalaba (11).

The method of seeking principle planes will reach the extremum of the truncated Taylor series approximation for the scalar function in M iterative steps. The value for this extremum point can be approximated by substituting Equation (2-41) and (2-43) into Equation (2-2).

$$X = X_0 - [G_X(X_0)^T\ G_X(X_0)]^{-1}\ G_X(X_0)^T\ G(X_0). \qquad (2\text{-}44)$$

If the Jacobian matrix, $G_X(X_0)$, is not singular, it can be shown that

$$[G_X(X_0)^T\ G_X(X_0)]^{-1} = G_X(X_0)^{-1}\ [G_X(X_0)^T]^{-1}. \qquad (2\text{-}45)$$

Upon the substitution of Equation (2-45) into (2-44), the following equation is obtained:

$$X = X_0 - G_X(X_0)^{-1}\ G(X_0). \qquad (2\text{-}46)$$

Equation (2-46) is the same as the Newton-Raphson method as presented by Bellman and Kalaba (11). Hence, in final convergence, the method of seeking principal planes behaves in the same manner as the Newton-Raphson technique for the solution of a set of nonlinear algebraic equations. This is a desirable characteristic since the Newton-Raphson method is very rapid in final convergence.

In Gauss's method for least squares analysis, the dimension of the vector G in Equation (2-36) will be larger than the dimension of the independent variable X. Hence, in general, it is not possible to find a solution such that Equation (2-36) will be satisfied exactly. A vector X can be determined such that the set of algebraic equations are satisfied in the least square error sense by minimizing Equation (2-37).

The iterative equation which Gauss uses for determining the minimum of Equation (2-37) is given by Equation (2-44). Equation (2-44) was obtained by assuming that the matrix of second partial derivatives as given by Equation (2-42) could be approximated by neglecting the term given by matrix B. For the situation where the system of equations can be satisfied exactly, Equation (2-39) indicates that matrix B will be the zero matrix at the minimum point; and it is legitimate within the neighborhood of a solution to approximate the matrix of second partial derivatives by neglecting the B matrix.

However, when the vector G is of larger dimension than

X, Equation (2-36) cannot, in general, be satisfied exactly. If the point which minimizes Equation (2-37) is actually a very poor solution for the original set of nonlinear equations, the matrix B may be a very significant portion of the matrix of second partial derivatives even at the exact location of a minimum point. In Appendix C, a mathematical proof is presented which demonstrates that Gauss's method as given by Equation (2-44) will under certain circumstances not converge to a minimum point regardless of how close to the minimum point one is. An example problem is given in the following section which demonstrates this characteristic.

A direct comparison of the computational effort required for all three methods is presented in Table I. The method of seeking principal planes required more computational time than the DFMFP and DFMCG methods. This was primarily due to using the analytic differentiation computation algorithm which is given in Appendix A within the SEEK algorithm. The equations generated by the analytic differentiation algorithm execute slower than standard Fortran IV statements.

Example Problems

Within this section, five example problems will be worked using the computational algorithm presented in Appendix B. The example problems have been specially chosen to demonstrate the widely varying capabilities of the method of seeking principal planes (SEEK).

TABLE I

COMPARISONS OF COMPUTATIONAL EFFORT REQUIRED
FOR EACH OF THE EXAMPLE PROBLEMS

		Analytical Minimum of Function	Value of Function at Termination of Program	Number of Function Evaluations	Number of Gradient Evaluations	Number of Matrix of Second Partial Derivatives Evaluations
Example 1	SEEK	0.0	0.0	2	2	1
	DFMFP		$.197 \times 10^{-29}$	17	17	None
	DFMCG		$.739 \times 10^{-30}$	25	25	None
Example 2	SEEK	0.0	$.426 \times 10^{-5}$	44	23	18
	DFMCG	0.0	$.312 \times 10^{-26}$	73	73	None
	DFMFP	0.0	$.818 \times 10^{-31}$	176	176	None
Example 3	SEEK	0.0	$.450 \times 10^{-14}$	37	15	11
	DFMCG	0.0	$.170 \times 10^{-27}$	107	107	None
	DFMFP	0.0	$.194 \times 10^{-69}$	87	87	None
Example 4	SEEK	0.0	$.211 \times 10^{-8}$	69	21	21
	DFMCG	0.0	$.534 \times 10^{-23}$	413	413	None
	DFMFP	0.0	$.260 \times 10^{-28}$	395	395	None
Example 5	SEEK		1.767037	7	3	3
	DFMCG		1.767037	37	37	None
	DFMFP		1.767037	183	183	None

The method of conjugate gradients is also a very powerful minimization technique. IBM has supplied two minimization programs in their Scientific Subroutine Package which are based on the conjugate gradient method as presented by Fletcher and Reeves (8). These programs are called DFMFP and DFMCG, and they require that the user provide an estimate for the minimum of the function (EST) and a value called EPS which is used to determine if final convergence has been obtained. The values of EST = 0.0 and EPS = 10.0^{-15} were used to generate the data which are presented in Table I. The SEEK algorithm is considered to have converged when the absolute value of each component of the gradient vector is less than 10^{-8}.

The first example problem is that of seeking the solution for a set of linear algebraic equations. The equations were specially chosen so that the matrix would be singular. The matrix A and the vector b were defined as

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \qquad b = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}. \tag{2-47}$$

Each of the algorithms determined an exact solution for the linear equation by minimizing

$$Y = [AX - b]^T [AX - b]. \tag{2-48}$$

By starting from the point X = (0,0,0,0), the solution X = (-.05, .025, .1, .175) was found.

Table I indicates that the SEEK algorithm had to

evaluate the scalar function three times, the gradient of the scalar function twice, and the matrix of second partial derivatives once in order to determine a minimum point for Equation (2-48).

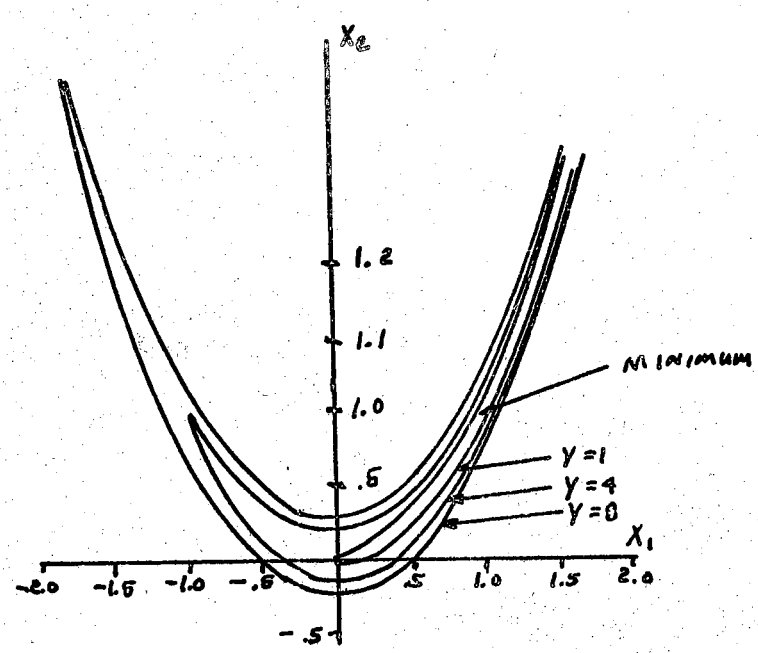
The computational effort which was required for the DFMCG and DFMFP algorithms is also given in Table I. This information is perhaps a little misleading since both DFMCG and DFMFP had reached the exact solution in four iterations. However, their convergence criteria did not terminate the program until after 17 and 25 iterations, respectively.

The second example which was worked is referred to by Fletcher and Reeves (8) as Rosenbrocks curved valley and its equation is

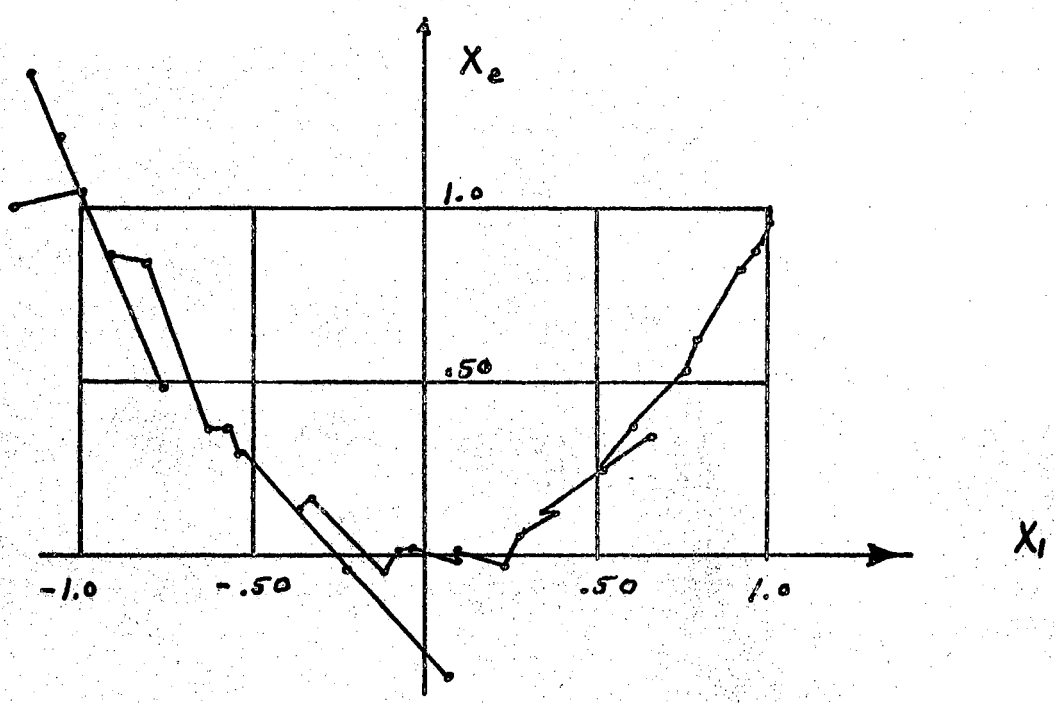
$$Y = 100.0 * (X_2 - X_1^2)^2 + (1 - X_1)^2. \qquad (2\text{-}49)$$

In part A of Figure 12 a rough sketch of some lines of constant contour for Equation (2-49) are given. It can be seen that the function has a "valley" which curves from the point (-1, 1) through the origin to the point (1, 1). In part B of Figure 12 the locus of points which the SEEK algorithm traversed as it proceeded to the minimum point (1.0, 1.0) from the point (-1.2, 1.0) is given. It can be seen that the SEEK algorithm follows the "valley" to the minimum.

The third example Fletcher and Reeves (8) refer to as the helical curved valley, and its equation is



Part A Sketch of lines of constant contour for Rosenbrocks curved valley



Part B Locus of points which the SEEK followed as it went from (-1.2, 1.0) to (1.0, 1.0)

Figure 12. Solution for Rosenbrocks Curved Valley Problem

$$Y = 100.0 * (X_3 - 10.0\ TAN^{-1}(X_2/X_1)/2\pi)^2 + \qquad (2\text{-}50)$$

$$100.0 * (\sqrt{X^2_1 + X^2_2} - 1.0)^2 + X^2_3 \quad \text{if} \quad X_1 > 0$$

$$Y = 100.0 * (X_3 - 10.0\ (TAN^{-1}(X_2/X_1)/2\pi + .50))^2 +$$

$$100.0 * (\sqrt{X^2_1 + X^2_2} - 1.0)^2 + X^2_3 \quad \text{if} \quad X_1 < 0.$$

The information given in Table I was generated by starting at the point (-1.0, 0.0, 0.0).

The fourth example worked was

$$Y = (X_1 + 10\ X_2)^2 + 5(X_3 - X_4)^2 + (X_2 - 2\ X_3)^4 +$$

$$+ 10(X_1 - X_4)^4. \qquad (2\text{-}51)$$

The information given in Table I was generated by starting at the point (3.0, -1.0, 0.0, 1.0).

The fifth example which was worked was that of determining the "best" solution of the following set of algebraic equations.

$$G_1 = X^2_1 + X^2_2 + .8786271 = 0$$

$$G_2 = .5^2\ X^2_1 + X^2_2 + .8786271 = 0 \qquad (2\text{-}52)$$

$$G_3 = X_1 - X_2 + .6207711 = 0.$$

A "best" solution in the least square error sense was determined by minimizing the function

$$Y = G^2_1 + G^2_2 + G^2_3. \qquad (2\text{-}53)$$

It was found that the point (-.159, .100) provides a local minimum for Equation (2-53). The information given in Table I was generated by starting from (-.10, .05).

The eigenvalues for the matrix $Q^T Q$ when evaluated at the point (-.159, .100) are 4.596×10^4 and 1.562. The matrix Q is defined in Appendix C. Since both of these eigenvalues are greater than one, Gauss's method will not converge to the solution point (-.159, .10) regardless of how close to that solution point one starts. This example will be discussed in more detail in Chapter IV.

From the results which are presented in Table I, it is noted that deflected gradient methods often require many more evaluations of the function and its gradient than the SEEK algorithm. However, the SEEK algorithm requires the computation of the matrix of second partial derivatives.

In the final analysis, it is debatable if the method of seeking principal planes requires less computational effort than the DFMCG and DFMFP algorithms. However, it provides valuable additional information since the eigenvalues of the matrix of second partial derivatives are computed. If all of the eigenvalues are positive when a converged solution is obtained, then Athans and Falb (9) state on page 227 that this provides both the necessary and sufficient conditions for a local minimum of the scalar function.

This chapter completely developed the method of seeking principal planes for the problem of determining the minimum of a function of several variables. In the next chapter,

the method of seeking principal planes is applied to two-point boundary value problems. This is a significant topic since optimal control theory can be used to convert a system design or identification problem into a two-point boundary value problem.

CHAPTER III

PARAMETER IDENTIFICATION FORMULATED AS A TWO-POINT BOUNDARY VALUE PROBLEM

The problem of parameter identification of dynamical systems can be reformulated as a two-point boundary value problem through the use of optimal control theory. In general, the first necessary conditions of optimal control theory for dynamical systems result in a two-point boundary value problem. An efficient solution technique for the two-point boundary value problem is necessary.

At present the basic methods for the solution of two-point boundary value problems are perturbation, quasilinearization, and deflected gradient. These are iterative procedures in which the missing initial conditions are guessed, and the differential equations are integrated out to the final time. On the basis of the information that has been obtained by integrating the differential equations, the initial conditions are changed. The amount and manner in which these initial conditions are changed separate the procedures from one another.

Lewallen, Tapley, and Williams (12) have investigated the quasilinearization and perturbation procedures in considerable detail. After reviewing their paper, it was

decided that it would be desirable to have a gradient technique in which the size of the step in the gradient direction was determined by some systematic method.

The method of seeking principal planes as was presented in Chapter II for the solution of algebraic minimization problems does have the desirable characteristic of a systematic step selection in the negative gradient direction. In this chapter, it will be shown that the two-point boundary value problem can be reformulated into a scalar minimization problem. Hence, the method of seeking principal planes can be used for the solution of two-point boundary value problems.

Statement of Parameter Identification Problems

It is assumed that a mathematical model for the system has been formulated and written in state variable notation,

$$\dot{X} = F(X, P, t). \tag{3-1}$$

In the above equation, X is the N dimensional state variable, P is an M dimensional parameter vector, and t is the independent variable, time.

The initial condition vector is partitioned as

$$X(t_0) = \begin{bmatrix} \psi \\ X_0 \end{bmatrix} \begin{matrix} \text{unknown} \\ \text{known.} \end{matrix} \tag{3-2}$$

In the above equation, ψ is a K dimensional vector of

unknown initial conditions, and X_0 is an N-K dimensional vector of the known initial conditions.

The entire parameter vector is unknown.

$$P = \text{unknown} \tag{3-3}$$

Values for the parameter vector, P, and the unknown initial conditions, ψ, need to be determined which will provide a local minimum for the performance index.

$$PI = \int_{t_0}^{t_f} G(X, t)\, dt. \tag{3-4}$$

Optimal control theory is presented by Athans and Falb (9) reformulates the problem presented by Equations (3-1) through (3-4) as a two-point boundary value problem. The dynamical equations which must be satisfied are:

$$\dot{X} = F(X, P, t) \tag{3-5}$$

$$\dot{P} = 0 \tag{3-6}$$

$$\dot{\lambda} = -G_X^T - F_X^T \lambda \tag{3-7}$$

$$\dot{\mu} = \quad - F_P \lambda. \tag{3-8}$$

λ is an N dimensional vector; μ is a K dimensional vector; and G_X, G_P, F_X, and F_P are defined as:

$$G_X = \left[\frac{\partial G}{\partial X_1}, \ \ldots, \ \frac{\partial G}{\partial X_M}\right], \tag{3-9}$$

$$F_X = \begin{bmatrix} \frac{\partial F_1}{\partial X_1} & \cdots & \frac{\partial F_1}{\partial X_N} \\ \vdots & & \\ \frac{\partial F_N}{\partial X_1} & \cdots & \frac{\partial F_N}{\partial X_N} \end{bmatrix}, \tag{3-11}$$

and

$$F_P = \begin{bmatrix} \frac{\partial F_1}{\partial P_1} & \cdots & \frac{\partial F_1}{\partial P_M} \\ \vdots & & \\ \frac{\partial F_N}{\partial P_1} & \cdots & \frac{\partial F_N}{\partial P_M} \end{bmatrix}. \tag{3-12}$$

The boundary conditions which Equations (3-5) through (3-8) must satisfy are:

$$X(t_0) = \begin{bmatrix} \psi \\ X_0 \end{bmatrix} \begin{matrix} \text{- free} \\ \text{- specified} \end{matrix} \qquad \begin{matrix} X(t_f) = \text{- free} \\ P(t_f) = \text{- free} \end{matrix} \tag{3-13}$$

$$P(t_0) = \text{free}$$

$$\lambda(t_0) = \begin{bmatrix} 0 \\ \lambda_{K+1}(t_0) \\ \vdots \\ \lambda_N(t_0) \end{bmatrix} \begin{matrix} \text{- specified} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{free} \end{matrix} \qquad \begin{matrix} \lambda(t_f) = 0 \text{ - specified} \\ \mu(t_f) = 0 \text{ - specified} \end{matrix}$$

$$\mu(t_0) = 0 \qquad \text{- specified}$$

The parameter identification problem has now been restated as a two-point boundary value problem. Equations (3-5) through (3-8) are the 2(N+M) first order dynamical equations which must be satisfied, and Equation (3-13) provides the set of 2(N+M) boundary conditions.

The following section outlines a method for solving a general two-point boundary value problem. The special two-point boundary value problem presented in this section falls within the framework of the material presented in the next section.

Statement of Two-Point Boundary Value Problem

The dynamical equations for the system are:

$$\dot{X} = F(X, t). \tag{3-14}$$

In the above equation, X is an N dimensional vector of dependent variables, and F is an N dimensional vector which is a function of the dependent variables and the independent variable, t.

The initial condition vector is partitioned as

$$X(t_0) = \begin{bmatrix} \psi \\ X_0 \end{bmatrix} \begin{matrix} \text{- unknown} \\ \text{- known.} \end{matrix} \tag{3-15}$$

In the above equation, ψ is a K dimensional vector of unknown initial condition, and X_0 is an N-K dimensional vector of known initial conditions.

The desired final boundary condition vector is

$$H(X)_{t_f} = 0. \tag{3-16}$$

H is a K dimensional vector. Each of the individual components of H can be written explicitly as a function of the components of the state vector evaluated at the specified final time, t_f.

Equations (3-14), (3-15), and (3-16) are a precise statement of the problem which needs now to be solved.

The relation between the problem statement in this and the previous one is clarified by noting that:

$$X_{\text{of this section}} = \begin{bmatrix} X_1 \\ \vdots \\ X_K \\ \lambda_{K+1} \\ \vdots \\ \lambda_N \\ P \\ \lambda_1 \\ \vdots \\ \lambda_K \\ X_{K+1} \\ \vdots \\ X_N \\ \mu \end{bmatrix} \text{ of the previous section}$$

$$
F_{\substack{\text{of this} \\ \text{section}}} = \left[\begin{array}{l}
F_1 \\
\vdots \\
F_K \\
-\dfrac{\partial G}{\partial X_{K+1}} - F_{K+1_X}\ \lambda \\
\vdots \\
-\dfrac{\partial G}{\partial X_N} - F_{N_X}\ \lambda \\
0 \\
-\dfrac{\partial G}{\partial X_1} - F_{1_X}\ \lambda \\
\vdots \\
-\dfrac{\partial G}{\partial X_K} - F_{K_X}\ \lambda \\
F_{K+1} \\
\vdots \\
F_N \\
\qquad - F_P{}^T\ \lambda
\end{array}\right] \quad \text{of the previous section}
$$

$$H_{\text{of this section}} = \begin{bmatrix} X_{K+1} \\ \vdots \\ X_N \\ X_{N+K+1} \\ \vdots \\ X_{N+2K} \\ X_{N+1} \\ \vdots \\ X_{N+K} \\ X_{N+2K+1} \\ \vdots \\ X_{2N+2K} \end{bmatrix} = \begin{bmatrix} \lambda \\ \mu \end{bmatrix} = 0 \quad \begin{matrix} \text{of the previous} \\ \text{section} \end{matrix}$$

$$\text{of this section}$$

$$N_{\text{of this section}} = 2(N + K) \quad \text{of the previous section}$$

$$K_{\text{of this section}} = 2K \quad \text{of the previous section.}$$

Sometimes the final boundary condition which results from optimal control theory is not precisely of the form given by Equation (3-16). This occurs when the final time is not specified in advance. Then, the final boundary condition is of the form

$$H(X, t) = 0. \tag{3-17}$$

To handle this problem two new state variables and a new independent variable, τ, are now introduced. Their

interrelations are given by Equations (3-18) through (3-23).

$$X_{N+1} = t - t_0 \tag{3-18}$$

$$X_{N+2} = \text{constant} \tag{3-19}$$

$$\tau = X_{N+1}/X_{N+2} = (t - t_0)/X_{N+2} \tag{3-20}$$

$$dt = X_{N+2}\, d\tau \tag{3-21}$$

$$dX_{N+1}/d\tau = X_{N+2} \tag{3-22}$$

$$dX_{N+2}/d\tau = 0. \tag{3-23}$$

Equation (3-24) results from substituting Equations (3-21) and (3-18) into Equation (3-23). Equations (3-22) and (3-23) are rewritten immediately below Equation (3-24) to provide the following complete set of dynamical equations.

$$dX/d\tau = X_{N+2}\, F(X, X_{N+1} + t_0) \tag{3-24}$$

$$dX_{N+1}/d\tau = X_{N+2} \tag{3-25}$$

$$dX_{N+2}/d\tau = 0. \tag{3-26}$$

The substitution of Equation (3-18) into (3-17) yields

$$H(X, X_{N+1} + t_0). \tag{3-27}$$

Equations (3-24) through (3-27) are equivalent in form to Equations (3-14) and (3-16). Hence, any method which

will solve the two-point boundary value problem presented at the start of this section can also be used to solve problems which have final boundary conditions of the form given by Equation (3-17).

Solution of Two-Point Boundary Value Problem by Method of Seeking Principal Planes

The method of seeking principal planes was developed in Chapter II for the minimization of a scalar function. The two-point boundary value problem can be converted into an algebraic minimization problem by defining the following scalar performance function.

$$Y(X(\psi)) = \frac{1}{2} H^T(X(\psi))\ H(X(\psi))_{t_f}. \qquad (3\text{-}28)$$

The scalar quantity, Y, is explicitly a function of the state vector, X, and implicitly a function of the unknown initial conditions, ψ.

This is more easily understood by studying the first two blocks of Figure 13. As the firgure indicates the first step in the procedure for determining the value of the scalar function is to make a guess for the unknown initial conditions. The second step of the procedure is to numerically integrate the dynamical equations from the initial time to the final time, t_f. Hence, the value of the state vector at the final time is a function of ψ. Once the value of the state vector at the final time is known, the value for the scalar performance function can be

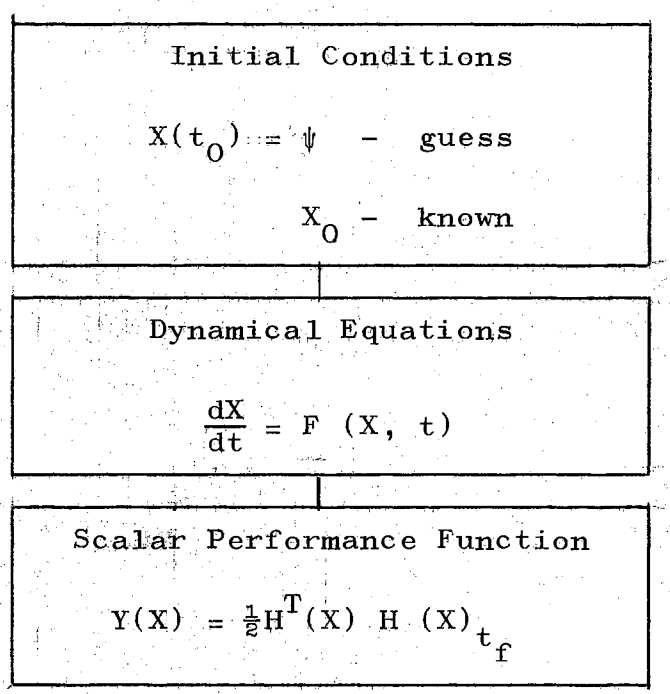


Figure 13. Equations Required to Determine the Value of the Scalar Performance Function

evaluated as indicated in block three of Figure 13. The equation for the scalar performance function is an explicit function of the state vector, X. But since the state vector is in turn a function of the unknown initial condition, ψ, the scalar performance function is implicitly a function of ψ.

The value of the scalar performance function must always be greater than or equal to zero since it is constructed as the sum of the components of the final boundary condition squared. If the scalar performance function equals zero, a solution for the two-point boundary value problem has been obtained.

The method of seeking principal planes requires computation of the gradient and the matrix of second partial derivatives of the scalar performance function.

In Figure 14 the necessary equations for determining the gradient of the scalar performance function are given. The notation used is

$$X_{\psi_i} = \frac{\partial X}{\partial \psi_i} \tag{3-29}$$

$$F_X = \begin{bmatrix} \frac{\partial F_1}{\partial X_1} & \cdots & \frac{\partial F_1}{\partial X_N} \\ \vdots & & \vdots \\ \frac{\partial F_N}{\partial X_1} & \cdots & \frac{\partial F_N}{\partial X_N} \end{bmatrix} \tag{3-30}$$

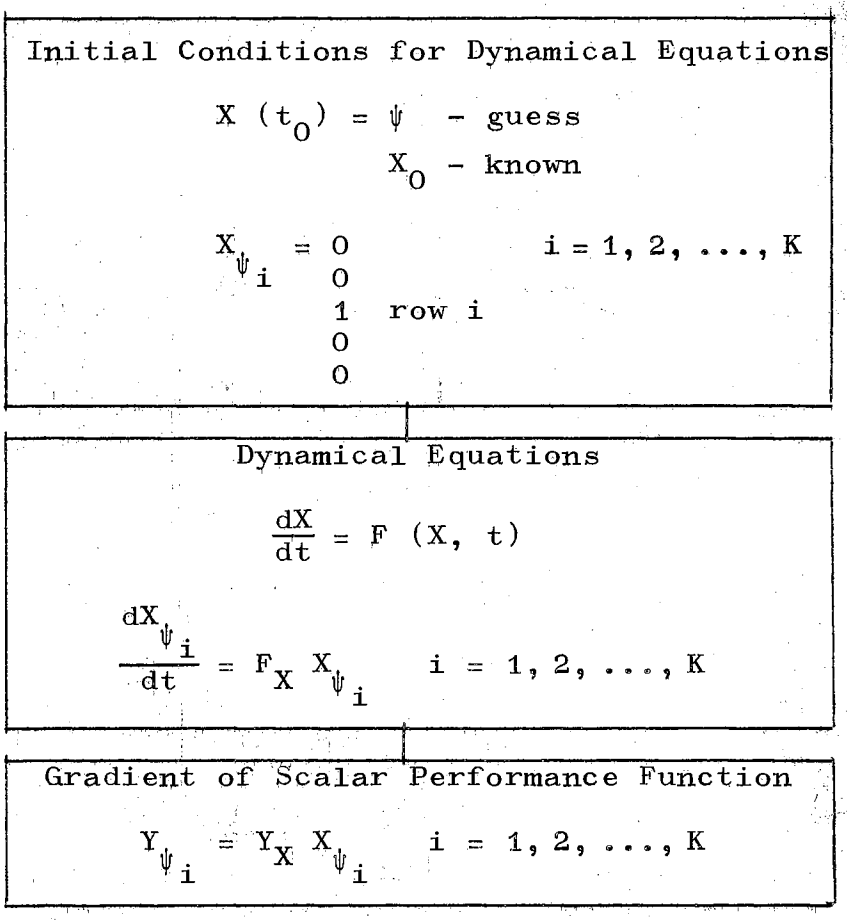


Figure 14. Equations Required to Determine the Values for the Gradient of the Scalar Function

$$Y_{\psi_i} = \frac{\partial Y}{\partial \psi_i} \tag{3-31}$$

$$Y_{X_i} = \frac{\partial Y}{\partial X_i} \tag{3-32}$$

$$Y_X = \left[Y_{X_1}, \quad \ldots, \quad Y_{X_N} \right]. \tag{3-33}$$

The first block of Figure 14 gives the proper initial conditions for the dynamical equations. The second block contains the dynamical equations which must be solved, and the third block has the equation for each of the components of the gradient vector.

The sensitivity of the state vector with respect to initial condition ψ_i is X_{ψ_i}. This is determined as a function of time by solving the set of homogeneous differential equations given in the second block. These equations are solved simultaneously with the equations for the basic dynamical system. Their mathematical development is given in Appendix D.

In Figure 15 the necessary equations for determining the elements of the matrix of second partial derivatives of the scalar performance function are given. The additional notation used is

$$X_{\psi_j \psi_i} = \frac{\partial X_{\psi_i}}{\partial \psi_j} \tag{3-34}$$

$$Y_{\psi_j \psi_i} = \frac{\partial Y_{\psi_i}}{\partial \psi_j} \tag{3-35}$$

Initial Conditions for Dynamical Equations

$$X(t_0) = \begin{bmatrix} \psi \\ X_0 \end{bmatrix} \begin{matrix} - \text{ guess} \\ - \text{ known} \end{matrix}$$

$$X_{\psi_i} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{ row } i \qquad i = 1, 2, \ldots, K$$

$$X_{\psi_j \psi_i} = 0 \qquad \begin{matrix} i = 1, 2, \ldots, K \\ j = 1, 2, \ldots, K \end{matrix}$$

Dynamical Equations

$$\frac{dX}{dt} = F(X, t)$$

$$\frac{dX_{\psi_i}}{dt} = F_X X_{\psi_i} \qquad i = 1, 2, \ldots, K$$

$$\frac{dX_{\psi_j \psi_i}}{dt} = F_X X_{\psi_j \psi_i} + \sum_{L=1}^{N} \left(\frac{\partial (F_X)}{\partial X_L} \frac{\partial X_L}{\partial \psi_j} \right) X_{\psi_i} \qquad \begin{matrix} i = 1, 2, \ldots, K \\ j = 1, 2, \ldots, K \end{matrix}$$

Elements of Matrix of Second Partial Derivatives of Scalar Performance Function

$$Y_{\psi_j \psi_i} = X^T_{\psi_i} Y_{XX} X_{\psi_j} + Y_X X_{\psi_j \psi_i} \qquad \begin{matrix} i = 1, 2, \ldots, K \\ j = 1, 2, \ldots, K \end{matrix}$$

Figure 15. Equations Required to Determine the Values for the Elements of the Matrix of Second Partial Derivatives of the Scalar Performance Function

$$Y_{X_i X_i} = \frac{\partial Y_{X_j}}{\partial X_i} \tag{3-36}$$

$$Y_{XX} = \begin{bmatrix} Y_{X_1 X_1} & \cdots & Y_{X_N X_1} \\ \vdots & & \vdots \\ Y_{X_1 X_N} & \cdots & Y_{X_N X_N} \end{bmatrix}. \tag{3-37}$$

The sensitivity of X_{ψ_i} with respect to initial ψ_j is given by $X_{\psi_j \psi_i}$. This is determined as a function of time by solving the set of coupled dynamical equations given in the second block of Figure 15. The mathematical development of these equations is given in Appendix D.

The method of seeking principal planes can determine a solution for the two-point boundary value problems posed by Equations (3-14) through (3-16) by minimizing Equation (3-28). Figures 13, 14, and 15 provide the necessary equations for the method of seeking principal planes procedure presented in Chapter II.

Relation to Quasilinearization Procedures

In this section it will be demonstrated that the quasilinearization methods for solving two-point boundary value problems are really just partial second-order method. A true second-order method for determining the minimum of a scalar function is the Newton-Raphson method which is given by Equation (2-3). However, the computational effort required for determining the matrix of second partial

derivatives is sometimes very considerable. Hence, it is very tempting to try to find a satisfactory approximation for the matrix of second partial derivatives.

Figure 15 provides the following equation for the elements of the matrix of second partial derivatives of Equation (3-28).

$$Y_{\psi_j \psi_i} = X^T_{\psi_i} Y_{XX} X_{\psi_j} + Y_X Y_{\psi_j \psi_i}. \tag{3-38}$$

Equation (3-28) has the following partial derivatives:

$$Y_{X_i} = H^T_{X_i} H, \tag{3-39}$$

and

$$Y_{X_j X_i} = H^T_{X_j X_i} H + H^T_{X_i} H_{X_j}; \tag{3-40}$$

where

$$H_{X_i} = \frac{\partial H}{\partial X_i} \tag{3-41}$$

and

$$H_{X_j X_i} = \frac{\partial H_{X_i}}{\partial X}. \tag{3-42}$$

But, when one is close to the desired solution for the two-point boundary problem, it can be seen from Equation (3-16) that H should be approximately zero. Hence, Equation (3-39) and the last term of Equation (3-40) should be approximately zero. This in turn implies that a good approximation for Equation (3-38) is

$$Y_{\psi_j \psi_i} = X^T_{\psi_i} D^T D X_{\psi_j}, \quad \text{Approx.} \tag{3-43}$$

where D is a K X N matrix defined as

$$D = \left[H_{X_1}, \ \ldots, \ H_{X_N} \right]. \tag{3-44}$$

The following equation is equivalent to the Newton-Raphson correction procedure given by Equation (2-3).

$$\psi_{new} = \psi_{old} - Y^{-1}_{\psi\psi} \ Y^T_{\psi}. \tag{3-45}$$

In Equation (3-45), the following notation is used:

$$Y_{\psi\psi} = \begin{bmatrix} Y_{\psi_1 \psi_1} & \cdots & Y_{\psi_N \psi_1} \\ \vdots & & \vdots \\ Y_{\psi_1 \psi_N} & & Y_{\psi_N \psi_N} \end{bmatrix} \tag{3-46}$$

$$Y_{\psi} = \left[Y_{\psi_1}, \ \ldots, \ Y_{\psi_K} \right]. \tag{3-47}$$

In Figure 14 the equation for the gradient vector is given as

$$Y_{\psi} = Y_X \ X_{\psi} \tag{3-48}$$

where

$$Y_X = \left[Y_{X_1}, \ \ldots, \ Y_{X_N} \right]. \tag{3-49}$$

and

$$X_{\psi} = \left[X_{\psi_1}, \ldots, X_{\psi_K} \right]. \tag{3-50}$$

If Equations (3-48) and (3-39) are combined, and the notation of Equation (3-44) is used, the result is

$$Y_{\psi} = H^T D X_{\psi}. \tag{3-51}$$

If the element of the matrix of second partial derivatives are approximated by Equation (3-43), the following equation results

$$Y_{\psi_{\psi}} = X_{\psi}^T D^T D X_{\psi}. \tag{3-52}$$
Approx.

When Equation (3-51) and the approximation given in Equation (3-52) are used, the Newton-Raphson correction method given by Equation (3-45) becomes

$$\psi_{new} = \psi_{old} - \left[X_{\psi}^T D^T D X_{\psi} \right]^{-1} X_{\psi}^T D^T H. \tag{3-53}$$

If the matrix DX_{ψ} is not singular, the above equation simplifies to

$$\psi_{new} = \psi_{old} - \left[D X_{\psi} \right]^{-1} H. \tag{3-54}$$

Equation (3-54) is the basic equation which is used for the quasilinearization solution methods for two-point boundary value problems. Since it was obtained by approximating the matrix of second partial derivatives, it is only an approximate second-order method.

The Newton-Raphson method for minimizing a scalar function will sometimes diverge from a solution if the original independent vector guess is not close enough to the desired solution. Since the quasilinearization method given by Equation (3-54) is only an approximate Newton-Raphson procedure, it is not surprising that the quasilinearization procedure will sometimes diverge if the original guess for the unknown initial conditions is not close enough to a desired solution.

In Appendix C it is proved that if too much of the matrix of second partial derivatives is truncated, a modified Newton-Raphson method will not converge. This problem is not encountered by the quasilinearization method provided one is sufficiently close to the desired solution. This is true since the portion of the matrix of second partial derivatives which was truncated becomes the zero matrix at the solution point. However, when the starting point is not close to the desired solution, the portion of the matrix of second partial derivatives which is truncated may be very significant.

Example five in Chapter II demonstrates the importance of not truncating too much of the matrix of second partial derivatives. Since truncating a portion of the matrix of second partial derivatives can cause difficulties for relatively simple algebraic minimization problems, one should be careful when a solution technique for two-point boundary value problems is being used which utilizes an approximation

for the matrix of second partial derivatives.

The method of seeking principal planes is a true second-order method. A number of example problems have been solved by using this method. The results which are presented in the following section are encouraging.

Examples

The computational algorithm which was used for working these example problems is presented in Appendix E. The algorithm is very versatile, and it is designed so that the user need only supply the basic information.

The algorithm makes extensive use of the digital computer's capability to generate partial derivatives, and this relieves the user of the responsibility for providing the equations for the partial derivatives.

Linear Example

The first example which will be presented is that of solving a simple linear two-point boundary value problem. The dynamical equations used were

$$\begin{aligned} \dot{X}_1 &= X_2 \\ \dot{X}_2 &= -X_1 - X_2 + 1.0. \end{aligned} \tag{3-55}$$

Both of the initial boundary conditions were unspecified, and the desired final boundary conditions were

$$X_1(4.0) = 1.2$$

$$X_2(4.0) = 0.0.$$

The method of seeking principal planes determined that the initial conditions of

$$X_1(0) = -.12688$$

$$X_2(0) = -.54238$$

would cause the dynamical equations to reach the desired final boundary conditions. The original guess for the initial condition vector was $X_1(0) = 0.0$ and $X_2(0) = 0.0$. Eight steps were used in the numerical integration of the dynamical equations by a Runga-Kutta algorithm.

The method of seeking principal planes converged after two evaluations of the performance index, one evaluation of the gradient vector, and one evaluation of the matrix of second partial derivatives.

This information is summarized in Figure 16, and for the subsequent examples all of the information will be presented in a figure of the same format.

The second-order Taylor series describes a linear two-point boundary value problem exactly. Hence, if a linear two-point boundary value problem has K unknown initial conditions, the method of seeking principal planes will converge after K evaluations of the scalar performance index, one evaluation of the gradient vector, and one evaluation of the matrix of second partial derivatives. This first

Dynamical Equations

$$\dot{X}_1 = X_2$$

$$\dot{X}_2 = -X_1 - X_2 + 1.0$$

Scalar Performance Index

$$Y = (X_1 - 1.2)^2 + X_2^2$$

Initial Condition Guess

$$X_1 = 0.0 \quad X_2 = 0.0$$

Converged Initial Condition

$$X_1 = -.12688 \quad X_2 = -.54238$$

Initial Time = 0.0

Final Time = 4.0

Number of Steps = 8

Converged After:

2 Evaluations of performance index
1 Evaluation of the gradient vector
1 Evaluation of matrix of second partial derivatives

Figure 16. Linear Two-Point Boundary Value Problem

example provided a demonstration of this point.

Drag Racer Example

A very crude dynamical model for a drag racing car is

$$\dot{X}_1 = X_2$$

$$\dot{X}_2 = P_1 + P_2 X^2_2 . \qquad (3\text{-}56)$$

The above model is obtained by assuming that the coefficient of friction between the wheels and the pavement remains constant, and that the aerodynamic forces are proportional to the velocity of the vehicle squared. In January of 1967, an AA fuel drag car was able to travel the quarter of a mile in 6.95 seconds reaching a speed of 221 mph.

It would be interesting to determine the parameters P_1 and P_2 such that the mathematical model, Equation (3-56), would hit the point

$$X_1 (6.9) = 1/4 \text{ mile} = 1320 \text{ ft.}$$

$$X_2 (6.9) = 221 \text{ mph} \approx 310 \text{ ft/sec.}$$

when started with the initial condition

$$X_1(0) = 0.0$$

$$X_2(0) = 0.0.$$

If time is nondimensionalized by 10, and distance is nondimensionalized by 1000, Equation (3-56) becomes

$$\begin{aligned} \dot{X}_1 &= X_2 \\ \dot{X}_2 &= X_3 + X_4 X_2^2 \\ \dot{X}_3 &= 0 \\ \dot{X}_4 &= 0 \end{aligned} \tag{3-57}$$

where $X_3 = P_1/10$ and $X_4 = 1000\ P_2$. The initial conditions become

$$\begin{aligned} X_1(0) &= 0.0 \\ X_2(0) &= 0.0 \\ X_3(0) &= ? \quad \text{(To be determined)} \\ X_4(0) &= ? \quad \text{(To be determined)} \end{aligned} \tag{3-58}$$

and the nondimensionalized desired final conditions become

$$\begin{aligned} X_1(.69) &= 1.32 \\ X_2(.69) &= 3.10. \end{aligned} \tag{3-59}$$

The results for the parameter identification problem given by Equation (3-57) and boundary conditions (3-58) and (3-59) are presented in Figure 17.

The converged value for the initial condition on X_3 is 7.0826. This means that P_1 in Equation (3-56) equals 70.826 ft/sec^2, which is slightly greater than twice the acceleration of gravity. Hence, the crude mathematical model predicts that as a drag racer leaves the starting line it should accelerate at more than twice the acceleration of gravity. If you are unfamiliar with the drag racing sport,

Dynamical Equations

$$\dot{X}_1 = X_2$$

$$\dot{X}_2 = X_3 + X_4 X_2^2$$

$$\dot{X}_3 = 0$$

$$\dot{X}_4 = 0$$

Scalar Performance Index

$$Y = (X_1 - 1.32)^2 + (X_2 - 3.10)^2$$

Known Initial Condition

$$X_1 = 0 \quad X_2 = 0$$

Initial Condition Guess

$$X_3 = 6.0 \quad X_4 = -.05$$

Converged Initial Condtion

$$X_3 = 7.0826 \quad X_4 = -.57584$$

Initial Time = 0.0
Final Time = .69
Number of Steps = 10

Converged After:

16 Evaluations of performance index
5 Evaluations of gradient vector
7 Evaluations of matrix of second partial derivatives

Figure 17. Drag Racer Parameter Identification Example

this may be a surprising prediction. But, among drag racing fans, it is a known fact that such high accelerations do occur.

Birta and Trushel Example

The dynamical equations used for this example were

$$\begin{aligned}
\dot{X}_1 &= (1.0 - X_2^2)\,X_1 - X_2 - X_3/2.0 \\
\dot{X}_2 &= X_1 \\
\dot{X}_3 &= -2.0\,X_1 - X_3 + X_3X_2^2 - X_4 \\
\dot{X}_4 &= -2.0\,X_2 + 2.0\,X_1X_2X_3 + X_3.
\end{aligned} \tag{3-60}$$

Equations (3-60) are given by Birta and Trushel (13).

The method of seeking principle planes was used to solve the two-point boundary value problem, and the results are presented in Figure 18.

Birta and Trushel reported that after 11 iterations by the conjugate gradient method, final convergence for the problem presented in Figure 18 was obtained. Each iteration of a deflected gradient method often requires at least four evaluations of the performance index and four evaluations of the gradient vector.

Schlichting Example

On page 332 of Schlichting (14) a two-point boundary value problem which arises when considering natural flow in thermal boundary layers is presented. This two-point

Dynamical Equations

$$\dot{X}_1 = (1.0 - X_2^2) X_1 - X_2 - X_3/2.0$$

$$\dot{X}_2 = X_1$$

$$\dot{X}_3 = -2.0X_1 - X_3 + X_3 X_2^2 - X_4$$

$$\dot{X}_4 = -2.0 X_2 + 2.0 X_1 X_2 X_3 + X_3$$

Scalar Performance Index

$$Y = X_3^2 + X_4^2$$

Known Initial Condition

$$X_1 = 0.0 \quad X_2 = .10$$

Initial Condition Guess

$$X_3 = 0.0 \quad X_4 = 0.0$$

Converged Initial Condition

$$X_3 = .081176 \quad X_4 = .66941$$

Initial Time = 0.0
Final Time = 4.0
Number of Steps = 40

Converged After:

10 Evaluations of the performance index
2 Evaluations of the gradient vector
7 Evaluations of the matrix of second partial derivatives

Figure 18. Results for the Birta and Trushel Example

boundary value problem was worked, and the results are presented in Figure 19.

Lewallen, Tapley, and Williams Example

The final two-point boundary value problem to be presented has been extensively studied by Lewallen, Tapley, and Williams (12). In their paper, they reported regions of convergence for several two-point boundary value solution techniques.

The method of seeking principal planes was able to solve the two-point boundary value problem from a poor initial condition guess. The initial condition guess used was in error by 100 per cent for each of the free adjoint variables. Lewallen, Tapley, and Williams were unable to obtain convergence from this poor initial condition guess without resorting to solution techniques which were problem oriented.

The results for the Lewallen, Tapley, and Williams example are presented in Figure 20.

Some of the versatility of the method of seeking principal planes has been shown by the two-point boundary value examples worked in this chapter. The results of the Lewallen, Tapley, and Williams example were especially encouraging since the method of seeking principal planes is not a problem oriented technique.

In Chapter II, the method of seeking principal planes was developed for determining the minimum of a function of

Dynamical Equations

$$\dot{X}_1 = X_2$$

$$\dot{X}_2 = X_4$$

$$\dot{X}_3 = X_5$$

$$\dot{X}_4 = -3.0\, X_1\, X_4 + 2.0\, X_2\, X_2 - X_3$$

Scalar Performance Index

$$Y = X^2_2 + X^2_3$$

Known Initial Condition

$$X_1 = 0.0 \quad X_2 = 0.0 \quad X_3 = 1.0$$

Initial Condition Guess

$$X_4 = .60 \quad X_5 = -.60$$

Converged Initial Condition

$$X_4 = .64189 \quad X_5 = -.56743$$

Initial Time = 0.0
Final Time = 10.0
Number of Steps = 20

Converged After:

15 Evaluations of performance index
7 Evaluations of gradient vector
8 Evaluations of matrix of second partial derivatives

Figure 19. Results of Schlichting Example

Dynamical Equations

$$\dot{X}_1 = X_7 \left(X^2_2/X_3 - 1.0/X^2_3 - .1405\, X_5/\right.$$
$$\left.((1.0 - .07487\, T\, X_7)\sqrt{X^2_5 + X^2_6})\right)$$

$$\dot{X}_2 = X_7 \left(-X_1 X_2/X_3 - .1405\, X_6/\right.$$
$$\left.((1.0 - .07487\, T\, X_7)\sqrt{X^2_5 + X^2_6})\right)$$

$$\dot{X}_3 = X_7 X_1$$

$$\dot{X}_4 = X_7 \left(X^2_2\, X_5/X^2_3 - 2.0\, X_5/X^3_3 - X_1\, X_2\, X_6/X^2_3\right)$$

$$\dot{X}_5 = X_7 \left(X_2\, X_6/X_3 - X_4\right)$$

$$\dot{X}_6 = X_7 \left(-2.0\, X_2\, X_5/X_3 + X_1\, X_6/X_3\right)$$

$$\dot{X}_7 = 0$$

Scalar Performance Function

$$Y = X^2_1 + (X_2 - .8098)^2 + (X_3 - 1.525)^2$$

Known Initial Condition

$$X_1 = 0.0 \quad X_2 = 1.0 \quad X_3 = 1.0 \quad X_4 = -1.0$$

Guessed Initial Condition

$$X_5 = -.9880 \quad X_6 = -2.16 \quad X_7 = 3.32$$

Converged Initial Condition

$$X_5 = -.49450 \quad X_6 = -1.0795 \quad X_7 = 3.3193$$

Initial Time = 0.0
Final Time = 1.0
Number of Steps = 20

Converged After:

18 Evaluations of performance index
4 Evaluations of gradient vector
10 Evaluations of matrix of second partial derivatives

Figure 20. Lewallen, Tapley, and Williams Example

several variables. In Chapter IV, it will be shown that if enough care is taken, this development can be applied directly to system design and system identification problems.

CHAPTER IV

PARAMETER IDENTIFICATION BY DIRECT METHODS

In the previous chapter, it has been demonstrated at the method of seeking principal planes can be used to solve the problem of parameter identification of dynamical systems by an indirect approach. Within this chapter, the method of seeking principal planes will be used as a direct solution approach to the problem of parameter identification of dynamical systems.

The parameter identification problem was stated in Chapter III, but it will now be presented again so that one need not page from one chapter to the other.

Statement of Parameter Identification Problem

It is assumed that a mathematical model for the system has been formulated and written in state variable notation,

$$\dot{X} = F(X, P, t). \qquad (4\text{-}1)$$

In the above equation, X is the N dimensional state variable, P is an M dimensional parameter vector, and t is the independent variable, time.

The initial condition vector is partitioned as

$$X(t_0) \begin{bmatrix} \psi \\ X_0 \end{bmatrix} \begin{matrix} \text{- unknown} \\ \text{- known.} \end{matrix} \qquad (4\text{-}2)$$

In the above equation ψ is a K dimensional vector of unknown initial conditions, and X_0 is an N-K dimensional vector of the known initial conditions.

The entire parameter vector is unknown.

$$P = \text{unknown} \qquad (4\text{-}3)$$

Values for the parameter vector, P, and the unknown initial conditions, ψ, need to be determined which will provide a local minimum for the performance index,

$$Y = \int_{t_0}^{t_f} G(X, t)\, dt. \qquad (4\text{-}4)$$

Direct Solution of Parameter Identification Problem by Method of Seeking Principal Planes

Since the performance index given by Equation (4-4) is a scalar function, the method of seeking principal planes as developed in Chapter II can be used to determine its minimum provided:

1. The performance index value can be determined when the values for the independent variables are specified,
2. The values for the gradient of the performance index can be determined when the values

for the independent variables are specified, and

3. The values for the elements of the matrix of second partial derivatives of the performance index can be determined when the independent variables are specified.

The above three steps can be performed, but the procedure needs to be explained since the performance index, Equation (4-4), is an implicit function of the independent variables, P and Ψ. This is more easily understood by studying Figure 21. As the figure indicates, the first step in the procedure for determining the value of the scalar performance index is to make a guess for the unknown initial conditions and the parameter vector. The second step of the procedure is to numerically integrate the dynamical differential equations for the initial time to the final time. Hence, the value of the state vector at each instant of time is a function of P and Ψ. Simultaneously, the integral equation for the performance index can be numerically integrated as is indicated by block three of Figure 21. The integrand of the performance index is an explicit function of the state vector, X, and time. But, since the state vector is in turn a function of the free initial conditions, Ψ, and the parameter vector, P; the scalar performance index is an implicit function of P and Ψ.

In Figure 22 the necessary equations for determining

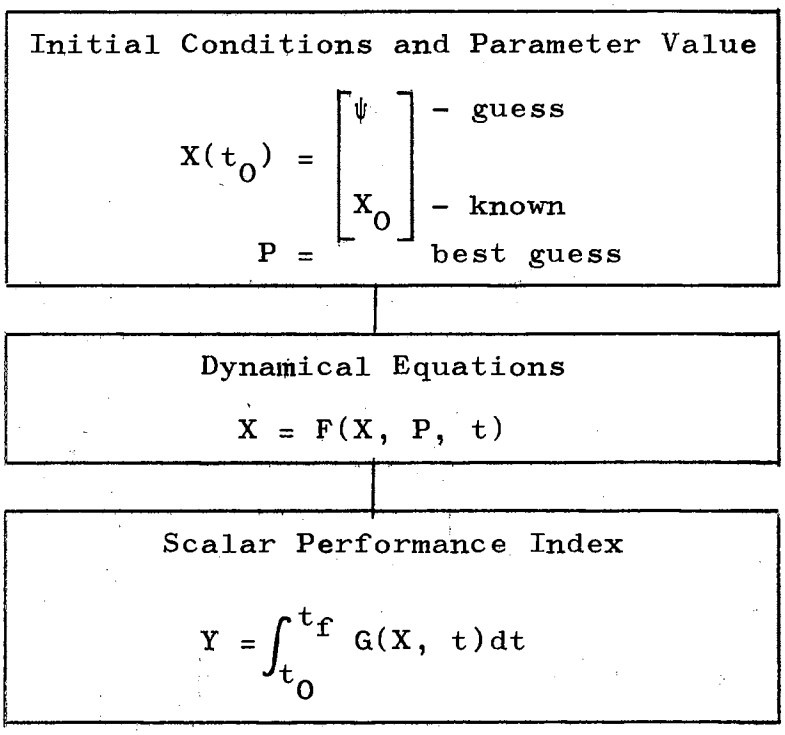


Figure 21. Equations Required to Determine the Value of the Scalar Performance Index

Initial Conditions and Parameter Value

$$X(t_0) = \begin{bmatrix} \psi & \text{guess} \\ X_0 & \text{known} \end{bmatrix}$$

$$X_{\psi_i} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{ - row } i \qquad i = 1, 2, \ldots, K$$

$$X_{P_i} = 0 \qquad i = 1, 2, \ldots, M$$

P = best guess

Dynamical Equations

$$\frac{dX}{dt} = F(X_1\ P_1\ t)$$

$$\frac{dX_{\psi_i}}{dt} = F_X\ X_{\psi_i} \qquad i = 1, 2, \ldots, K$$

$$\frac{dX_{P_i}}{dt} = F_X\ X_{P_i} + F_{P_i} \qquad i = 1, 2, \ldots, M$$

Gradient of Scalar Performance Index

$$Y_{\psi_i} = \int_{t_0}^{t_f} G_X\ X_{\psi_i}\ dt \qquad i = 1, 2, \ldots, K$$

$$Y_{P_i} = \int_{t_0}^{t_f} G_X\ X_{P_i}\ dt \qquad i = 1, 2, \ldots, M$$

Figure 22. Equations Required to Determine the Value of the Gradient Vector

the gradient of the scalar performance index are given. The only notation which is different from that used in Chapter III is

$$X_{P_i} = \frac{\partial X}{\partial P_i}, \quad Y_{P_i} = \frac{\partial Y}{\partial P_i}, \quad F_{P_i} = \frac{\partial F}{\partial P_i}$$

and

$$G_X = \left[\frac{\partial G}{\partial X_1}, \ \ldots, \ \frac{\partial G}{\partial X_N}\right].$$

The first block of Figure 22 gives the proper initial conditions for the dynamical equations, the second block contains the dynamical equations which must be numerically integrated, and the third block contains the integral equation for each of the components of the gradient vector.

The sensitivity of the state vector with respect to initial condition ψ_i is X_{ψ_i}, and the sensitivity with respect to P_i is X_{P_i}. These sensitivities are determined as a function of time by solving the set of differential equations given in the second block of Figure 22. These equations are coupled with the equations for the basic dynamical system, and their mathematical development is given in Appendix D.

In Figure 23 the necessary equations for determining the elements of the matrix of second partial derivatives of the performance index are given. The only notation which is different from that used in Chapter III is

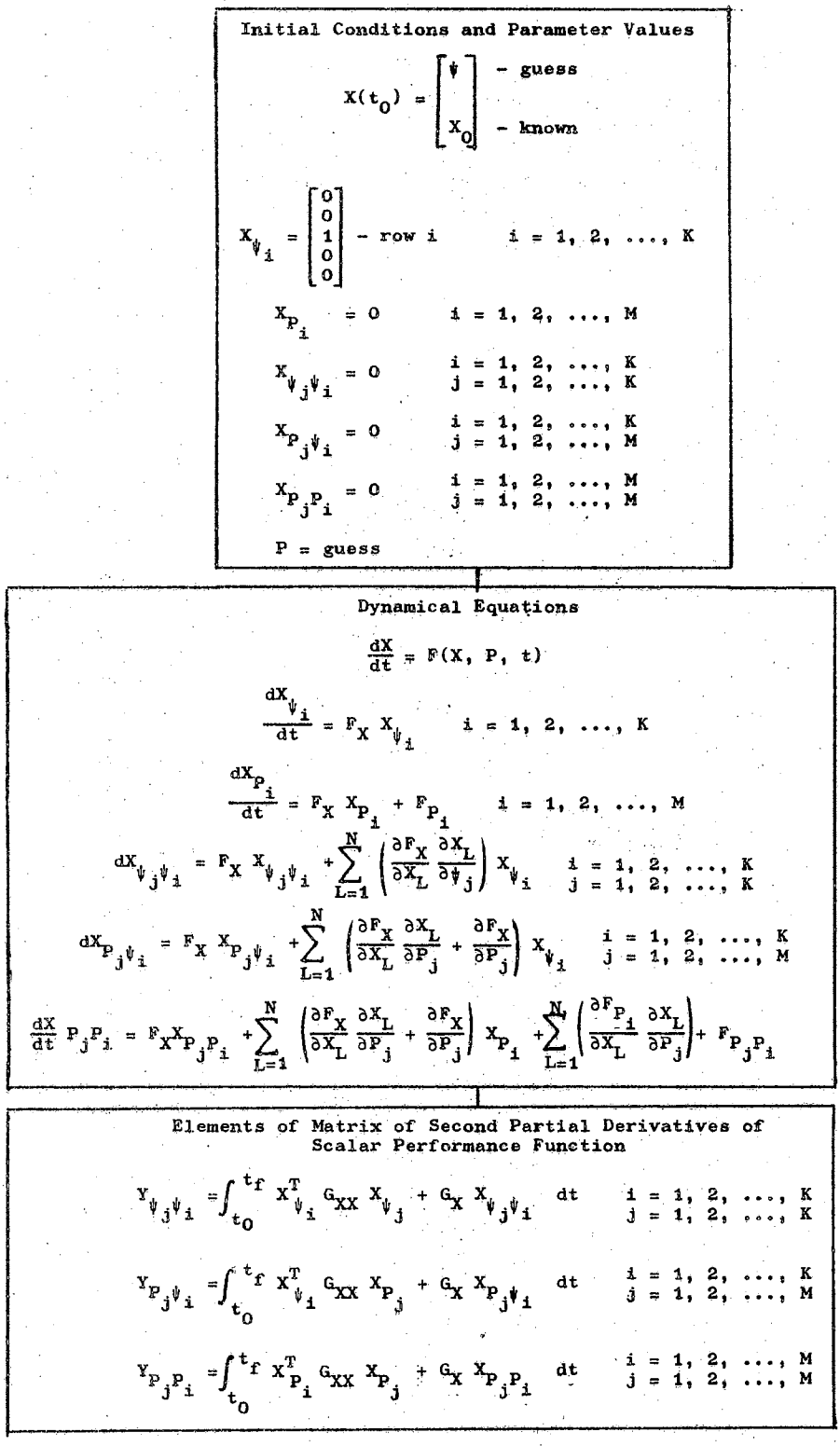


Figure 23. Equations Required to Determine the Values for the Elements of the Matrix of Second Partial Derivatives of the Performance Index

$$X_{P_j\psi_i} = \frac{\partial X_{\psi_i}}{\partial P_j}, \quad X_{P_jP_i} = \frac{\partial X_{P_i}}{\partial P_j}, \quad F_{P_jP_i} = \frac{\partial F_{P_i}}{\partial P_j}$$

and

$$G_{XX} = \begin{bmatrix} \frac{\partial^2 G}{\partial X_1 X_1} & \cdots & \frac{\partial^2 G}{\partial X_N X_1} \\ \vdots & & \vdots \\ \frac{\partial^2 G}{\partial X_1 X_N} & \cdots & \frac{\partial^2 G}{\partial X_N X_N} \end{bmatrix}.$$

The mathematical development of the equations for determining the second-order sensitivity coefficients ($X_{\psi_j\psi_i}$, $X_{P_j\psi_i}$, and $X_{P_jP_i}$) is given in Appendix D.

Figures 21, 22, and 23 contain all equations which are needed for determining (1) the value of the performance index, (2) the gradient of the performance index, and (3) the matrix of second partial derivatives of the performance index. Hence, the method of seeking principal planes can be used as a direct method for determining a minimum point for the performance index.

The differential equations which must be solved to determine the second-order sensitivity coefficient are quite involved. The second-order sensitivity coefficients would be unnecessary if the second term in the equations for determining the elements of the matrix of second partial derivatives is truncated (see Figure 23). However, example five in Chapter II demonstrates that considerable care should be exercised before making such a truncation.

The method of seeking principal planes was used to work several example problems, and the results are presented in the following section.

Examples

The computational algorithm which was used for working these example problems is presented in Appendix F. The algorithm is very versatile, and it is designed so that the user need only supply the basic information.

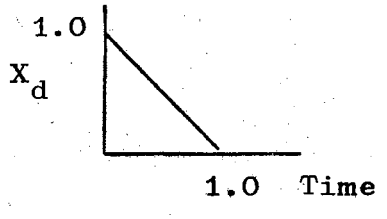
The algorithm makes extensive use of the digital computer's capability to generate partial derivatives, and this relieves the user of the responsibility for providing the equations for the partial derivatives.

Example 1

The first example worked was that of determining the proper initial condition and parameter value such that

$$\dot{X} = -PX \tag{4-5}$$

would have a dynamical response which fit the desired response, Xd,



as close as possible. The performance index used was the standard integral error squared criterion,

$$Y = \int_0^{1.0} (X - X_d)^2 \, dt. \qquad (4\text{-}6)$$

The computational effort required and other pertinent results and information for this problem are presented in Figure 24.

Example 2

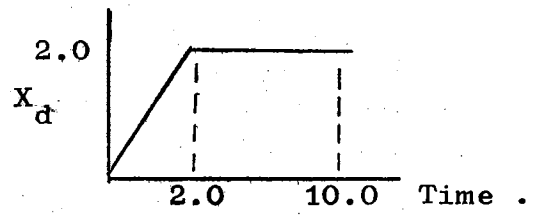
The mathematical model for this example was

$$\begin{aligned} \dot{X}_1 &= X_2 \\ \dot{X}_2 &= -P_1 X_1 - P_2 X_2 + 1.0. \end{aligned} \qquad (4\text{-}7)$$

Once again the integral error squared criterion was used

$$Y = \int_0^{10} (X - X_d)^2 \, dt. \qquad (4\text{-}8)$$

But the desired response which was used for this example was



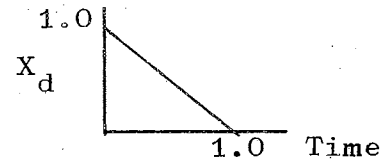
Difficulty had been encountered in previous attempts to work this parameter identification problem by using standard gradient techniques. The difficulty encountered was that of zigzagging back and forth across a valley with relatively insignificant changes in the parameters.

Dynamical Equations

$$\dot{X} = - PX$$

Performance Index

$$Y = \int_0^1 (X - X_d)^2 \, dt$$



Number of Time Steps = 5
Initial Time = 0.0
Final Time = 1.0
Number of Free Initial Conditions = 1

Initial Condition Guess

$$X_1 = 1$$

Converged Initial Condition

$$X_1 = 1.0892$$

Initial Parameter Guess

$$P = 1.0$$

Converged Parameter Value

$$P = 1.8037$$

Converged After:

10 Evaluations of performance index
3 Evaluations of gradient vector
5 Evaluations of the matrix of second partial derivatives

Figure 24. Equations and Results for Example 1

But the method of seeking principal planes was able, with no difficulties, to solve this problem. The results are given in Figure 25. In Figure 26, the response of the system for the original parameter guess and for the converged parameter vector is presented. It can be seen that the converged response is considerably improved.

Example 3

This example is the same as Example 2 except that the initial conditions for X_1 and X_2 are left free. The results are presented in Figure 27.

Since this example has more free independent parameters than Example 2, it is reasonable to expect that the converged solution will more closely fit the desired response. Figure 28 shows that the previous statement is true.

Spool Valve Example

A fundamental component of many hydraulic systems is a spool valve such as is shown in Figure 29. In Dr. Bose's (6) thesis, it is shown that under certain circumstances the following mathematical model is reasonable:

$$\dot{X}_1 = X_2$$

$$\dot{X}_2 = -.36X_2 - .24X_1 - P_1X^3_1 - P_2X_1X_2 + 1.0.$$

Proper values for P_1 and P_2 need to be determined so that the spool valve will require a minimum amount of time to open and have a steady-state opening of approximately 1.0.

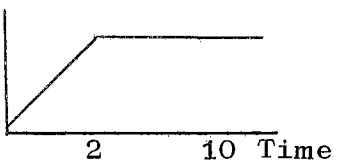
Dynamical Equations

$$X_1 = X_2$$

$$X_2 = - P_1X_1 - P_2X_2 + 1.0$$

Performance Index

$$Y = \int_0^{10} (X - X_d)^2 \, dt$$



Number of Time Steps = 20
Initial Time = 0.0
Final Time = 10.0
Number of Free Initial Conditions = None

Initial Condition Vector

$X_1 = 0.0$ $X_2 = 0.0$

Initial Parameter Guess

$P_1 = 1.0$ $P_1 = 2.0$

Converged Parameter Vector

$P_1 = .53045$ $P_2 = .67262$

Converged After:

24 Evaluations of performance index
5 Evaluations of gradient vector
8 Evaluations of matrix of second partial derivatives

Figure 25. Equations and Results for Example 2

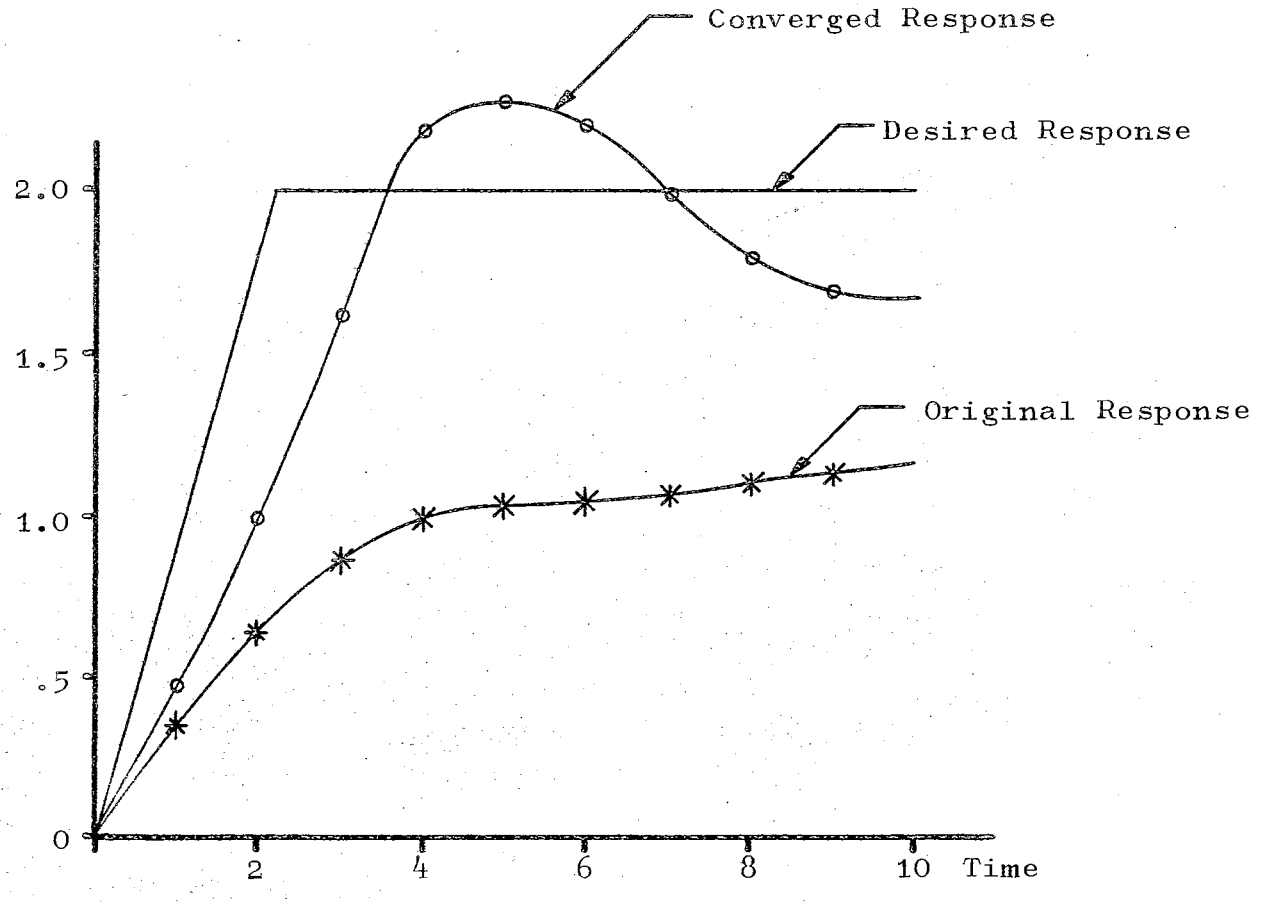


Figure 26. This Figure Shows the Response Improvement Which Was Obtained for $\ddot{Y} = -P_1X - P_2\dot{X} + 1.0$

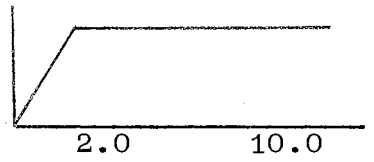
Dynamical Equations

$$\dot{X}_1 = X_2$$

$$\dot{X}_2 = -P_1X_1 - P_2X_2 + 1.0$$

Performance Index

$$Y = \int_0^{10} (X - X_d)^2 \, dt$$



Number of Time Steps = 20
Initial Time = 0.0
Final Time = 10.0
Number of Free Initial Conditions = 2

Initial Condition Guess

$X_1 = 0.0$ $X_2 = 0.0$

Converged Initial Condition

$X_1 = -.14634$ $X_2 = 1.6947$

Initial Parameter Guess

$P_1 = 1.0$ $P_2 = 2.0$

Converged Parameter Vector

$P_1 = .51136$ $P_2 = 1.1786$

Converged After:

38 Evaluations of the performance index
7 Evaluations of the gradient vector
11 Evaluations of the matrix of second partial derivatives

Figure 27. Equations and Results for Example 3

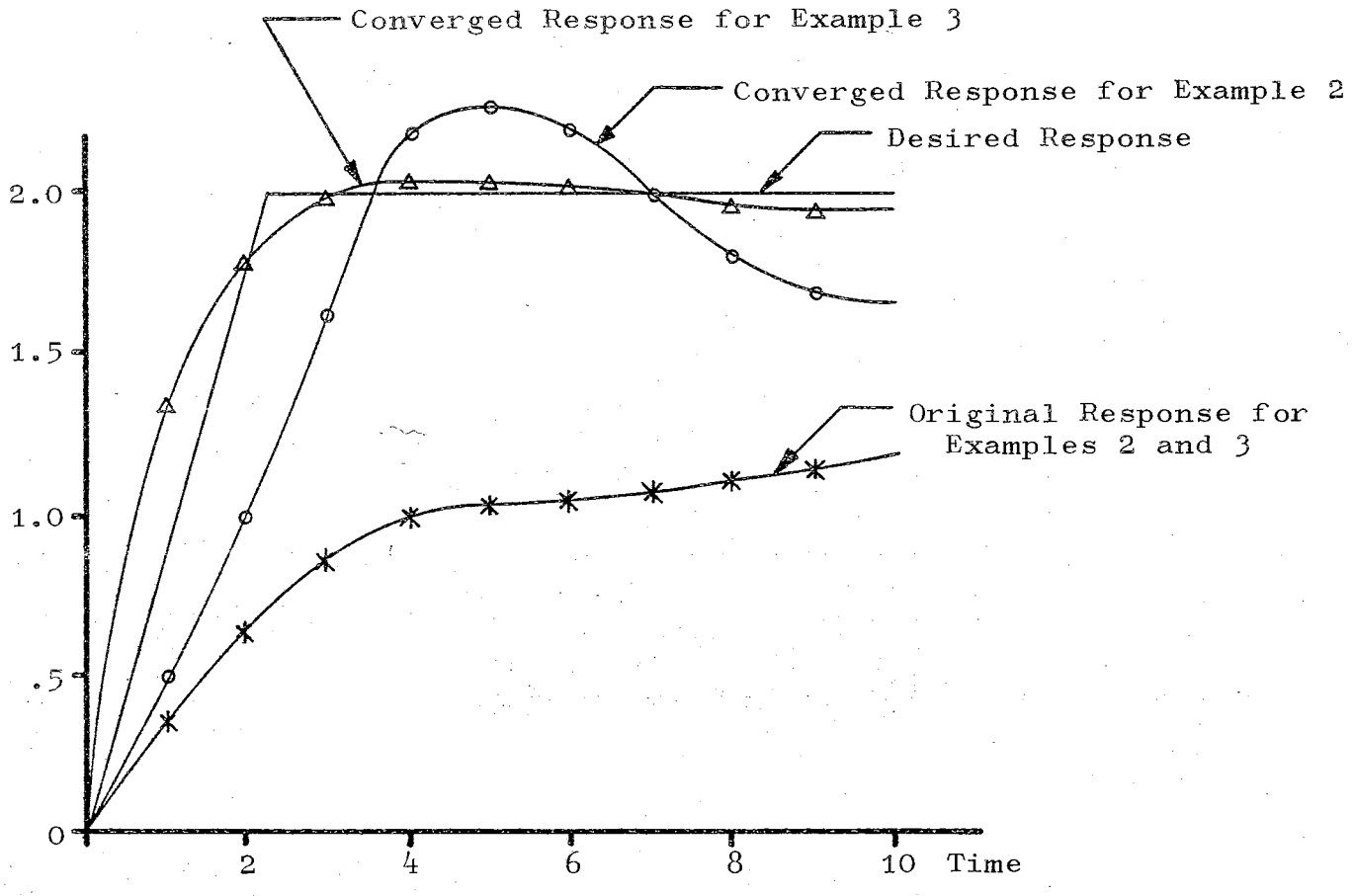


Figure 28. This Figure Shows That the Response for the Converged Solution of Example 3 is Considerably Better Than the Converged Solution Response of Example 2

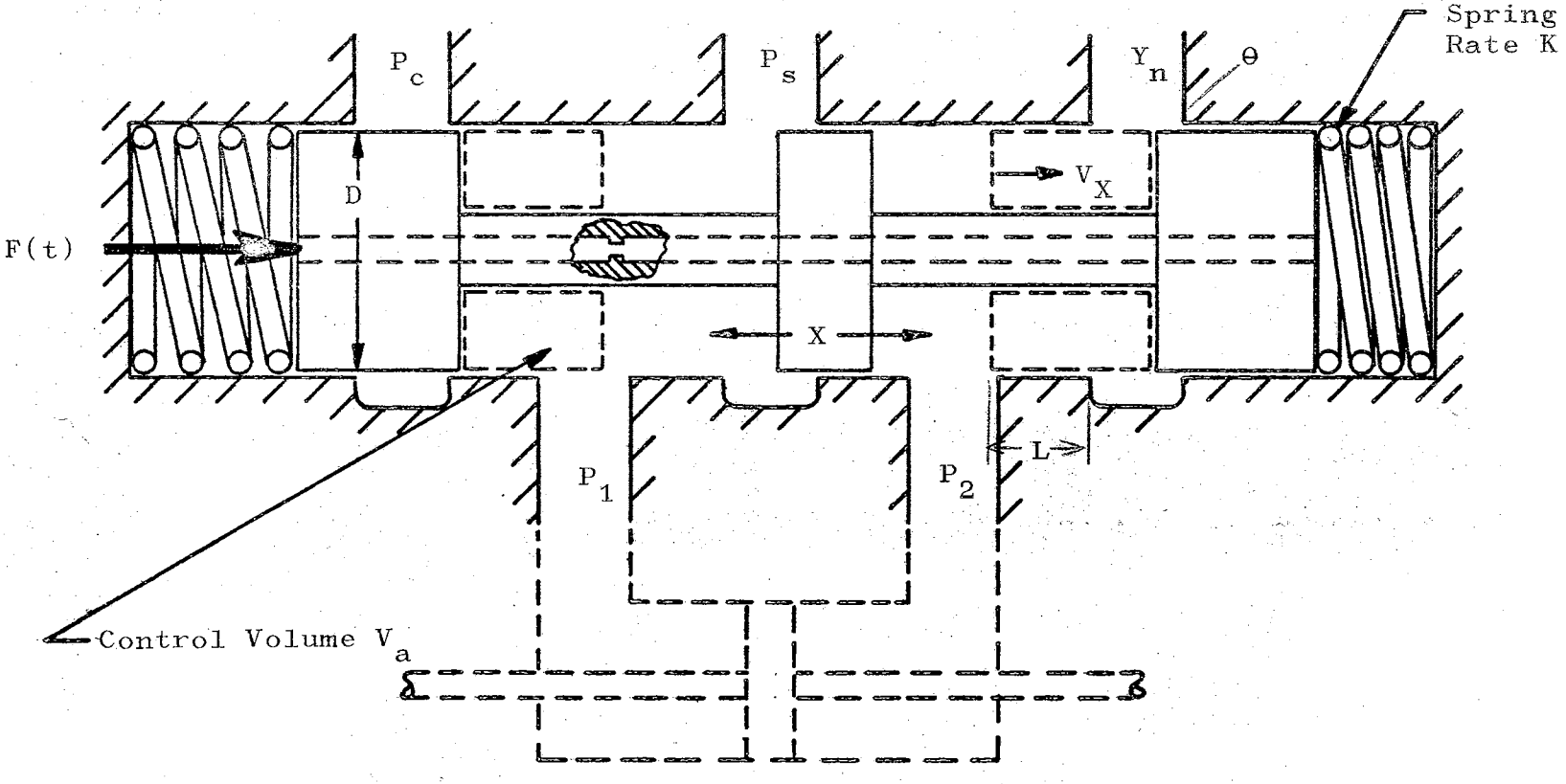
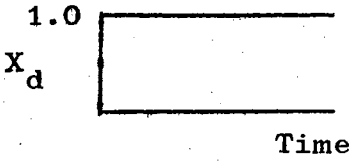


Figure 29. Hydraulic Spool Valve

The $P_1X^3_1$ term in the equation represents a "hard" spring, and the $P_2X_1X_2$ term in the equation represents the unsteady flow force.

The desired response of the valve is that it instantaneously opens, i.e.,



The values of $P_1 = 0$, $P_2 = 0$ were chosen as the starting point for determining a minimum point of

$$Y = \int_0^{5.0} (X - X_2)^2 \, dt$$

by the method of seeking principal planes. The equations used and the obtained results are presented in Figure 30. In Figure 31 the original response and the converged solution response are presented. The improvement in the response is quite dramatic.

This chapter has shown that the method of seeking principal planes can, quite effectively, be directly applied to system identification or design problems. A variety of examples were worked and all converged with no difficulties.

The method of seeking principal planes is effective for determining optimal parameters of dynamical systems. The examples worked in this chapter and Chapter III verify the preceding statement.

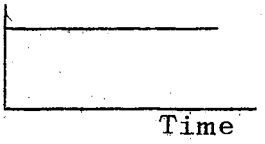
Dynamical Equations

$$\dot{X}_1 = X_2$$

$$\dot{X}_2 = -.36\ X_2 - .24\ X_1 - P_1\ X^3_1 - P_2\ X_1\ X_2 + 1.0$$

Performance Index

$$Y = \int_0^{5.0} (X - X_d)^2\ dt$$



Number of Time Steps = 20
Initial Time = 0.0
Final Time = 5.0
Number of Free Initial Conditions = None

Initial Condition Vector

$X_1 = 0.0$ $X_2 = 0.0$

Initial Parameter Guess

$P_1 = 0.0$ $P_2 = 0.0$

Converged Parameter Vector

$P_1 = .86122$ $P_2 = 1.3174$

Converged After:

21 Evaluations of the performance index
3 Evaluations of gradient vector
15 Evaluations of matrix of second partial derivatives

Figure 30. Equations and Results for the Spool Valve Example

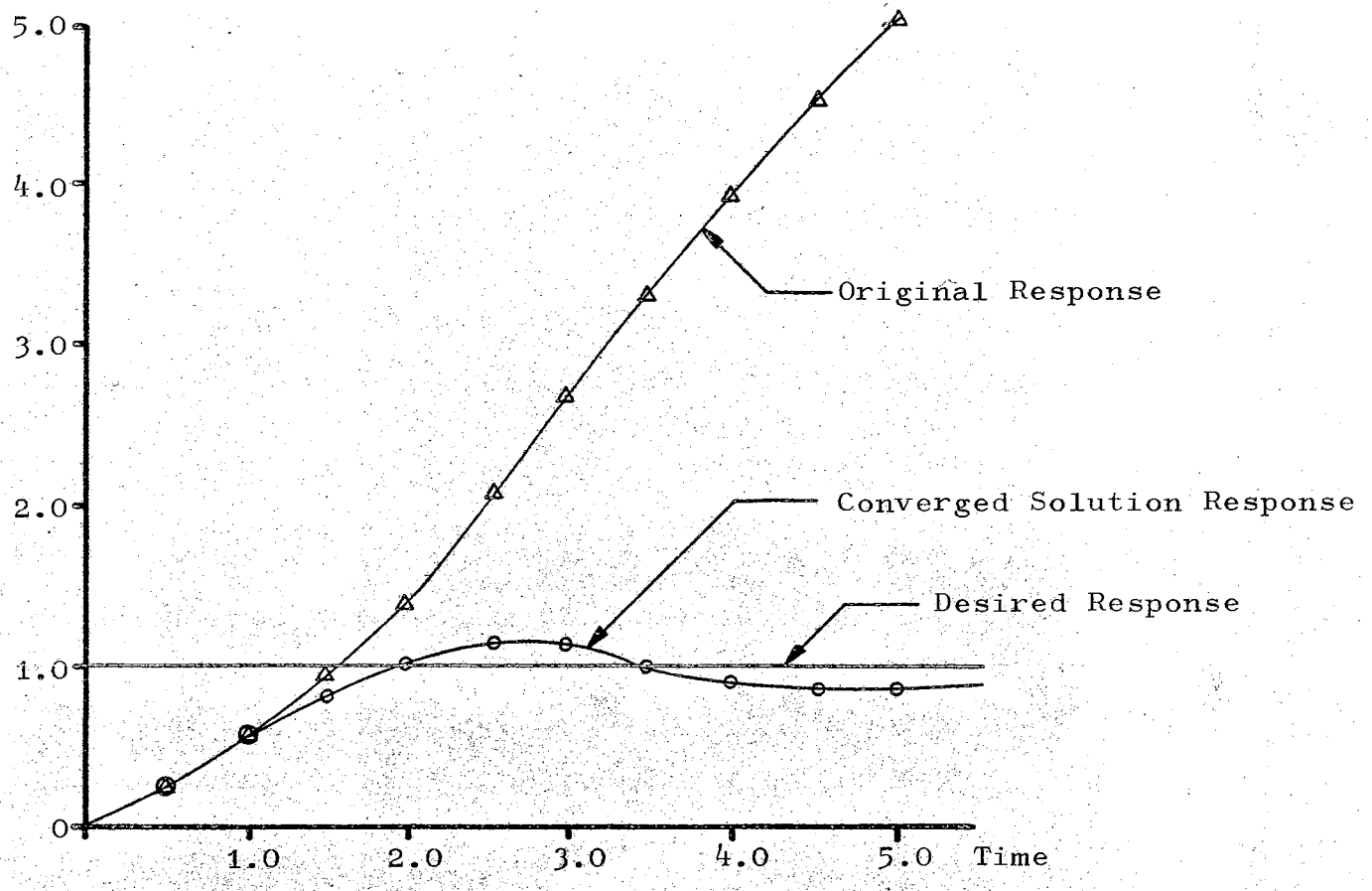


Figure 31. This Figure Shows the Improvement Which Was Obtained in the Spool Valve Response

However, all of these examples have all been continuous dynamical systems. If the dynamical system has discontinuities during the time period of interest, the problem becomes much more difficult. The following chapter deals with some of these problems and contains several proposals of methods for minimizing them.

CHAPTER V

DYNAMICAL SYSTEM DISCONTINUITIES: PROBLEMS AND PROPOSALS

Problems Caused by Dynamical System Discontinuities

In the dynamical analysis of hydraulic control systems, the functional form of the right-hand side of the state vector model will sometimes change. The time at which the functional form changes will quite often depend upon the state vector. Two examples will be presented to clarify the preceding statements.

Figure 2 shows a hydraulic system which contains a poppet type relief valve. In Figure 32 a typical response for the position and velocity of the poppet is shown. The figure depicts a situation where the relief valve is dumping oil to the reservoir during the time interval of t_0 to t_1. Since the valve is open during this time period, the functional form of the state model for the poppet will be governed by Newton's second law. At time t_2 the poppet hits the seat, and the functional form of the state model will abruptly change. This causes a discontinuity in the poppet velocity. At time t_2 the opening forces are larger

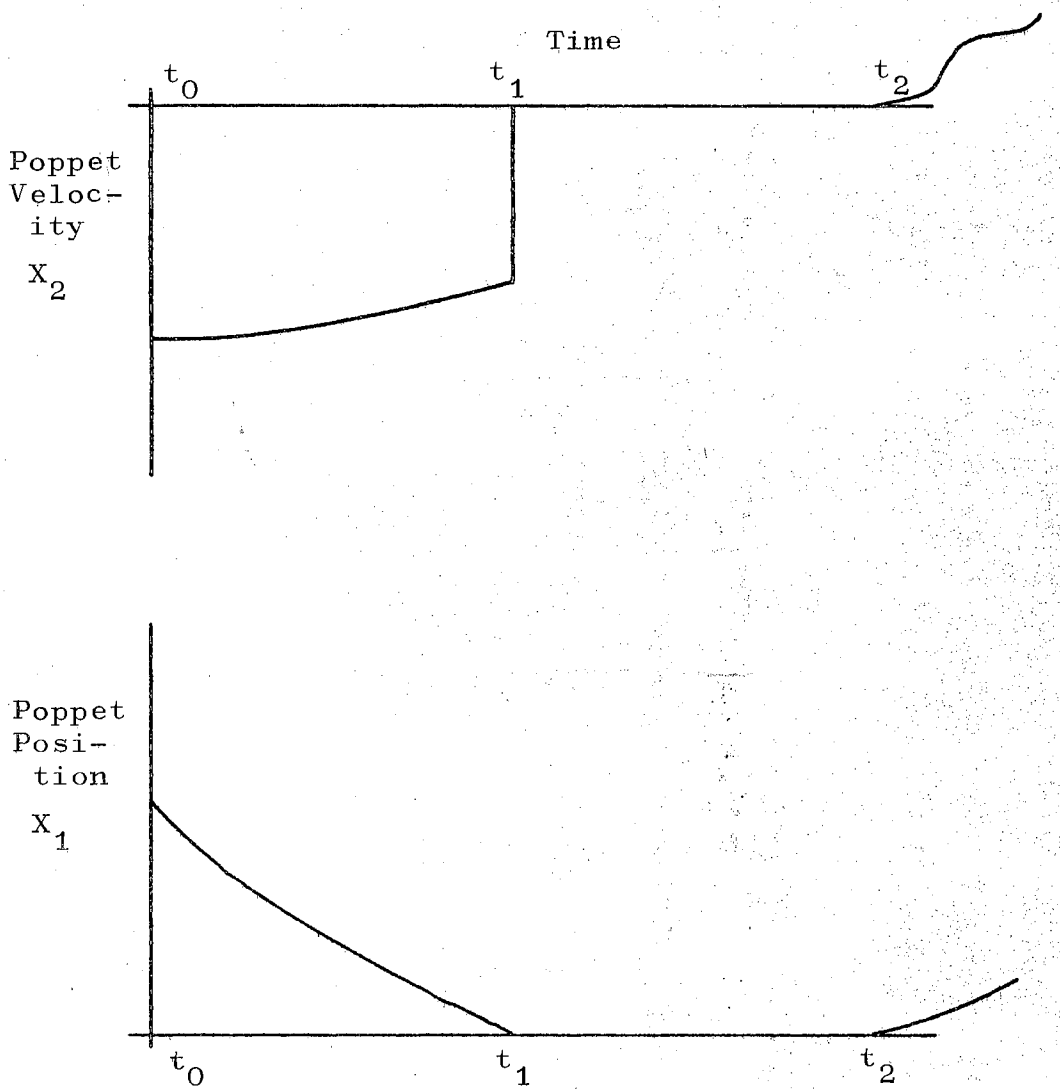


Figure 32. Response of Poppet's Velocity and Position

than the closing forces, and the functional form of the state model changes so that it is once again governed by Newton's Law.

The functional form of the state model for the spool valve shown in Figure 29 will depend upon the spool's position. If it is an overlapped valve, the mathematical model will depend upon whether the spool is open to the right, in its center position, or open to the left. And this provides another example of a dynamical system whose functional form of the right hand side of the state model depends upon the state vector.

The change in functional form of the right-hand side of the state vector model causes the following two problems to appear:

1. Difficulties are encountered in the numerical integration of the dynamical system equation. If a variable step size algorithm is employed, it will often (at points similar to that depicted at time t_1 in Figure 39) use such a small step size that very slow progress is made.
2. If the method of seeking principal planes is to be used on a parameter identification problem, it was stated in Chapter II that all of the third partial derivatives of the scalar performance function should be continuous. It is difficult if not impossible to

insure that such conditions are met at the points where the functional form of the dynamical equations change.

Proposals for Reducing the Problems Caused by Dynamical System Discontinuities

It may be possible to eliminate the second problem presented above by using techniques such as are presented in Lastman's Thesis (15). Lastman's basic approach is to expand the state vector in a special manner so that all of the state variables are continuous. This approach will not be simple, but it does provide one possible way for approaching the problem.

Zeiger (16) has recently presented a semiexhaustive search approach for solving the parameter identification problem presented in Chapter IV. The method presented by Zeiger requires only that the user be capable of numerically integrating the dynamical equations. Zeiger's approach becomes feasible for use on systems where the form of the dynamical equations change if the numerical integration can be performed without using too small a step size.

A method for avoiding the small step size problem will now be presented. A typical Runga Kutta numerical integration algorithm requires four evaluations of the right-hand side of the state model for each time step. The numerical integration of the differential equations which would provide the response shown in Figure 33 could be expected to

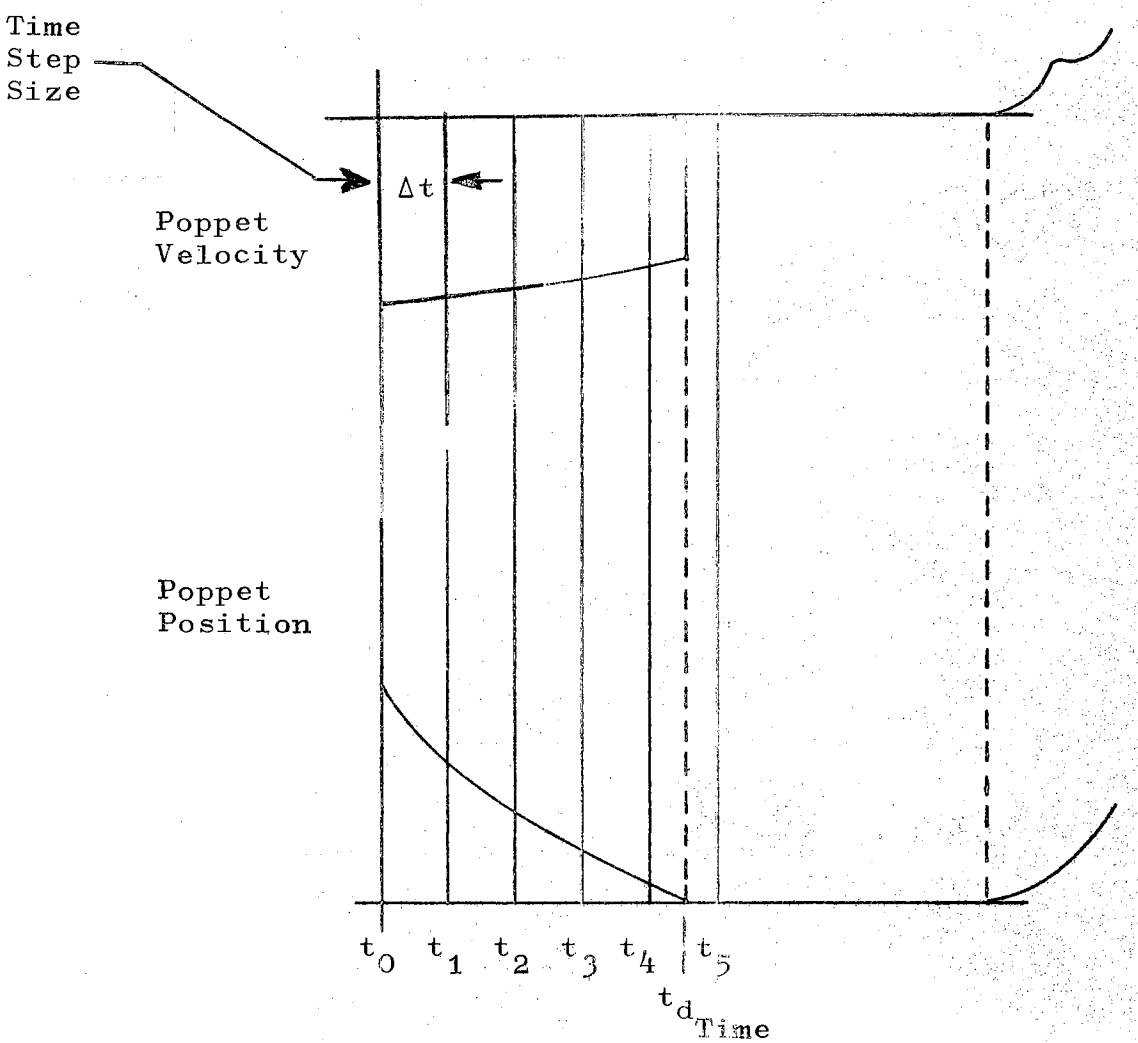


Figure 33. The Fifth Time Step Contains the Discontinuity Time, t_D

proceed smoothly during the first four time steps. But the functional form of the right-hand side of the state model will change during the fifth time step, and trouble will occur since the Runga Kutta numerical integration algorithm is based on the assumption that the functional form of the state model will not change.

If the precise time for the discontinuity, t_d, could be determined, and the value for the state vector at that time could be determined, the standard Runga Kutta Numerical integration procedure could be used to carry on from that point. One method for locating the time at which the discontinuity occurs, t_d, is to expand the state vector in a Taylor series. If the last time point at which the numerical integration was proceeding smoothly is used for the Taylor series expansion (t_4 in Figure 33), it will become

$$X(t_d) = X(t_4) + \frac{dX}{dt}\Big|_{t_4}(t_d - t_4) + \frac{1}{2}\frac{d^2X}{dt^2}\Big|_{t_4}(t_d - t_4)^2. \qquad (5\text{-}1)$$

In the hypothetical examples of the poppet valve and the spool valve which were presented earlier in this chapter, the point at which the state model changed form was governed by one component of the state vector becoming equal to a particular value. For example, the functional form of the poppet valve equations change when the poppet position becomes equal to zero ($X_1 = 0$). The discontinuity time for the poppet example can be determined by truncating Equation (5-1) after its second term. The result is

$$t_d = t_4 \frac{\frac{-dX_1}{dt} \pm \sqrt{\left(\frac{dX_1}{dt}\right)^2 - 2\left(\frac{d^2X_1}{dt^2}\right)(X_1(t_4) - X_1(t_d))}}{\frac{d^2X_1}{dt^2}}. \quad (5\text{-}2)$$

In Equation (5-2), the derivatives need to be evaluated at t_4, and the value for $X_1(t_d)$ is zero for the poppet example. The equation for the first derivative is obtained from the state model, and it is

$$\left.\frac{dX_1}{dt}\right|_{t_4} = F_1(X(t_4),\ t_4). \quad (5\text{-}3)$$

The equation for the second derivative is obtained by differentiating Equation (5-3), then the result is

$$\frac{d^2X_1}{dt^2} = \left.\frac{\partial F_1}{\partial X}\right|_{(X(t_4),t_4)} \left.\frac{dX}{dt}\right|_{(X(t_4),t_4)} + \left.\frac{\partial F_1}{\partial t}\right|_{(X(t_4),t_4)}. \quad (5\text{-}4)$$

The state model for the system is

$$\left.\frac{dX}{dt}\right|_{(X(t_4),t_4)} = F(X(t_4),t_4). \quad (5\text{-}5)$$

Equations (5-3), (5-4), and (5-5) can now be substituted into Equation (5-3) to yield

$$t_d = t_4 \frac{-F_1 \pm \sqrt{F_1^2 - 2\left(\frac{\partial F_1}{\partial X}F + \frac{\partial F_1}{\partial X}\right)\left[X_1(t_4) - X_1(t_d)\right]}}{\left(\frac{\partial F_1}{\partial X}F + \frac{\partial F_1}{\partial t}\right)}. \quad (5\text{-}6)$$

The discontinuity time can be determined from Equation (5-6). F_1 and all of the partial derivatives of F_1 in

Equation (5-6) must be evaluated at $(X(t_4), t_4)$.

Once the discontinuity time has been determined, the value for the state vector at that time can be determined by substituting Equations (5-4) and (5-5) into Equation (5-1) and truncating after two terms. The result is

$$X(t_d) = X(t_4) + F\,(t_d - t_4) + \left(\frac{\partial F}{\partial X} F + \frac{\partial F}{\partial t}\right)(t_d - t_4)^2. \tag{5-7}$$

Equations (5-6) and (5-7) appear to be rather complicated, but if the digital computer's capability for performing partial differentiation is used (see Appendix A), Equations (5-6) and (5-7) can be included in a Runga Kutta numerical integration algorithm which requires the user to provide very little additional information.

The steps which have been presented in this chapter are:

1. Use a standard Runga Kutta numerical integration algorithm until a point is reached at which the functional form of the differential equations changes.
2. Determine the precise time at which the equation functional form changes.
3. Determine the value of the state vector at the time where the equation functional form changes.
4. Return to step 1.

The steps which have been outlined in this chapter

should make the parameter identification procedure presented by Zeiger feasible for dynamical systems which have discontinuities.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

Conclusions

The basic philosophy of the method of seeking principal planes is quite flexible. This is verified by the application in this research of the basic philosophy to three types of problems (1. algebraic minimization, 2. two-point boundary value problems, and 3. parameter identification of dynamical systems) within this thesis.

The method of seeking principal planes has the following desirable characteristics when applied to the problem of parameter identification of dynamical systems:

1. The method has excellent convergence characteristics when the parameter guess is close to an optimum point. This occurs since the method is a true second-order technique.
2. The method seems to be quite stable even if the original parameter guess is not close to an optimum parameter point, and it will quite often converge.
3. The method adjusts any specified parameters in the dynamical equations instead of adjusting only the coefficients.

4. The method of seeking principal planes does not require that the parameter identification problem first be reformulated into a two-point boundary value problem.
5. The desired response and the inputs for the dynamical equations can be presented in graphical form.
6. The method does not require the knowledge of the response of the entire state vector at all instants of time.
7. The method is not restricted to linear dynamical systems. It works well on parameter identification problems whose dynamical equations are non-linear.

The method of seeking principal planes has the following undesirable characteristics:

1. As presented in this thesis, limits have not been placed on the values which the parameters can assume. Sometimes the physics of a problem dictates that certain ranges of parameter values are unrealistic. Therefore, the computational algorithm could be improved by placing constraints on the allowable parameter values.
2. The method of seeking principal planes requires considerable computational effort since the second-order influence coefficients need

to be numerically integrated over the time period of interest.

Recommendations

The following specific recommendations are areas in which further work could be performed:

1. Constraints on the allowable ranges for the parameter values could be inserted into the computational algorithm.
2. The possibility of using hybrid computer facilities with the method of seeking principal planes for parameter identification should be investigated. This appears to be a very promising area.
3. The first and second-order sensitivity coefficients (X_{ψ_i}, X_{P_i}, $X_{\psi_j\psi_i}$, $X_{\psi_j\psi_i}$, and $X_{P_jP_i}$ might need to be scaled for some particular problems. Difficulties in this area were not encountered in any of the problems worked in this thesis.
4. The determination of convergence regions for several parameter identification problems would provide useful results for comparing the method of seeking principal planes with other procedures.
5. The proposals presented in Chapter V for minimizing the difficulties encountered when dynamical system discontinuities occur should

be investigated.

6. The results presented in this thesis show that the method of seeking principal planes will work well for problems having up to five dependent variables. It would be interesting to investigate whether the procedure will work eqully well in problems which have hundreds of independent variables.

7. The method has promise of being a very powerful approach for determining the inverse of ill-conditioned matrices.

8. For the optimal control of dynamical systems, one normally tries to find a continuous control which will cause the response of the dynamic system to minimize some specified performance index. Figure 34 illustrates that a continuous optimal control trajectory can be approximated by a series of constant control levels. Hence, the continuous optimal control of dynamical systems can be approximated as a parameter identification problem with the free parameters P_1 through P_7 as indicated in Figure 34.

9. Kokotovic (17) has shown that the gradient vector for the parameter identification problem can be determined through the use of the adjoint differential equations. The advantage

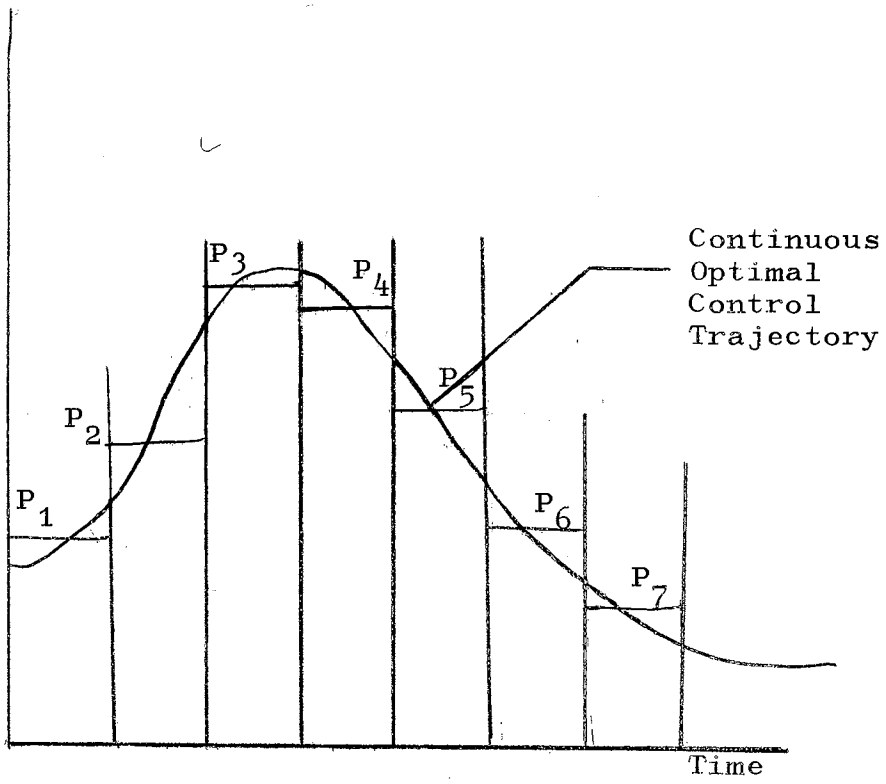


Figure 34. This Figure Illustrates That a Continuous Optimal Control Trajectory Can be Approximated by a Series of Constant Control Levels

of this approach is that the required computational effort remains feasible even when the number of parameters becomes large. If this approach can also be used for determining the matrix of second partial derivatives for the parameter identification problem, it may become quite feasible to use the method of seeking principal planes for the parameter identification problem even when a large number of parameters are involved.

SELECTED BIBLIOGRAPHY

(11) Bellman, R. E., and R. E. Kalaba. Quasilinearization and Nonlinear Boundary Value Problems. New York: American Elsevier Publishing Company, 1965.

(12) Lewallen, T. M., B. D. Tapley, and S. D. Williams. "Iteration Procedures for Indirect Trajectory Optimization Methods." Journal of Spacecraft, Vol. 5, No. 3 (March, 1968), pp. 321-327.

(13) Birta, L. G., and P. T. Trushel. "Conjugate Gradient Search in the Initial Costate Space." IEEE Transactions on Automatic Control, Vol. AC-12, No. 6 (December, 1967), pp. 776-777.

(14) Schlictling, Herman. Boundary Layer Theory. New York: McGraw-Hill, 1960.

(15) Lastman, G. J. "Optimization of Nonlinear Systems With Inequality Constraints." (Ph.D. Thesis, University of Texas, December, 1966.)

(16) Zeiger, Giora. "An Algorithm for Parameter Optimization in Dynamic Systems Design." Basic Fluid Power Research Program, School of Mechanical and Aerospace Engineering, Oklahoma State University (July, 1969).

(17) Kokotovic, P., and J. Heller. "Direct and Adjoint Sensitivity Equations for Parameter Optimization." IEEE Transactions on Automatic Control, Vol. AC-12, No. 5 (October, 1967), pp. 609-610.

(18) Sammet, J. E., and E. R. Bond. "Introduction to FORMAC." IEEE Transactions on Electronic Computers, Vol. 13, No. 4 (August, 1964), pp. 386-394.

(19) Bekey, G. A., and R. B. McGhee. "Gradient Methods for the Optimization of Dynamical System Parameters by Hybrid Computation." Computing Methods in Optimization Problems, ed., New York: Academic Press, pp. 305-326.

APPENDIX A

ANALYTIC PARTIAL DERIVATIVES OF ALGEBRAIC FUNCTIONS BY THE DIGITAL COMPUTER

In engineering and mathematical analysis, the need often arises to perform total or partial differentiation of an algebraic function. The basic rules which the mathematicians have established for differentiating an algebraic function are very systematic. In practice, the application of these rules to an arbitrary function often becomes very tedious and time consuming.

IBM has a compiler which is called FORMAC that will perform formal mathematical manipulations of mathematical functions. An integral part of the FORMAC compiler is the ability to differentiate most mathematical expressions. However, this program as described by Sammet and Bond (18) is available for only IBM computers operating under the PL/1 monitor.

Since the author did not have access to a FORMAC compiler, the author developed the algorithm given in this appendix. It will analytically differentiate most mathematical expressions which can be written in the Fortran IV language, and a listing is given at the end of this appendix. It is written in the Fortran IV language and can

be used on any computer which can process Fortran IV.

An algebraic function can contain two types of variables: dependent and independent variables. The value for a dependent variable cannot be determined unless the values of the particular independent variables upon which it depends are given first.

For a function which contains one independent variable and no dependent variables, the total derivative and partial derivative are identical.

If a function contains one independent variable and one or more dependent variables, the total derivative is determined by use of the chain rule. To utilize the chain rule, one must be able to calculate the partial derivative of the analytic function with respect to each of the dependent variables and with respect to the independent variable. Hence, if the partial derivatives of an algebraic function can be calculated, the total derivative can be determined by use of the chain rule. It is for this reason that the scope of this appendix is limited to the determination of analytical partial derivatives of algebraic functions.

Since the basic rules for calculating partial derivatives of algebraic functions are very systematic, the digital computer can be programmed to execute these rules. The following sections of this appendix will demonstrate one particular method for performing partial differentiation on the digital computer.

Operators and Operands Involved in Basic Algebraic Functions

The operators which this computational program will handle and their corresponding Fortran IV symbols are given in Table II.

It is convenient in the following pages to refer to any algebraic function which contains a single type operator as a basic algebraic function. It will be demonstrated in the following section that any analytic function can be decomposed into a series of basic algebraic functions.

Since each basic algebraic function has a single type of operator, it is reasonable to ask what constitutes an allowable operand and how many operands are required for each operator. There are three types of allowable operands for a basic algebraic function, and they are (1) a single constant, (2) a variable, or (3) an algebraic function. Categories one and two are actually special case of the third category. The first two categories are very important since they are the simplest algebraic functions known.

The addition and multiplication operators are variary operators since the number of operands depends upon the particular function being considered. The addition operator has one or more operands, and the multiplication operator has two or more operands. The power raising operator has two operands. The division and square root operations are treated within the computational program as a combination of multiplication and power raising. The rest of the operators

TABLE II

OPERATORS AND FORTRAN IV SYMBOLS WHICH THE COMPUTATION PROGRAM WILL ACCEPT

Operator	Fortran IV Symbol
Addition	+
Multiplication	*
Division	/
Power Raising	**
Subtraction	-
Exponentation	EXP
Natural Logrithm	ALOG
Base 10 Logrithm	ALOG10
Sine	SIN
Cosine	COS
Tangent	TAN
Inverse Cosine	ARCOS
Inverse Sine	ARSIN
Inverse Tangent	ATAN
Hyperbolic Sine	SINH
Hyperbolic Cosine	COSH
Hyperbolic Tangent	TANH
Square Root	SQRT
Absolute Value	ABS
Cotangent	COTAN

in Table II are unary operators since they have only one operand.

Decomposing an Algebraic Function

In order to decompose an algebraic function which has been written in Fortran IV notation into a series of basic algebraic functions, it is necessary to utilize the standard Fortran IV order of computation. This hierarchy of operations is given in Table III. In the instances where the operations are of equivalent level, the operations are performed as they are encountered by proceeding from the left to the right through the function.

The decomposition of an algebraic function into a series of basic algebraic functions is best illustrated through the use of an example. It will be noted that the procedure uses the hierarchy of operations table extensively. The algebraic function whose equation is

$$(X(1)*X(2)+X(1))*SIN(X(2)) \qquad (A-1)$$

will now be composed.

The hierarchy of operations table indicates that the first operation which must be performed is to remove the parenthesis which surround the subscripts of the variables. The algebraic function being considered can then be rewritten as

$$(X_1*X_2+X_1)*SIN(X_2). \qquad (A-2)$$

TABLE III

HIERARCHY OF OPERATIONS

	Hierarchy
Remove parenthesis delimiting subscript	1st
Evaluate subfunction delimited by innermost parenthesis set by utilizing hierarchy steps three through six	2nd
Evaluation of Functions	3rd
Power Raising (**)	4th
Multiplication and Division (* and /)	5th
Addition and Subtraction (+ and -)	6th

The next step indicated by the hierarchy table is to evaluate the subfunction which is delimited by the innermost parenthesis set. The first innermost parenthesis set which is encountered in proceeding from the left to the right is the subfunction

$$X_1*X_2+X_1. \tag{A-3}$$

Hierarchy steps three through six are now performed on this subfunction. There are no function operators or power raising operators involved in the subfunction, and hierarchy steps three and four yield no results. The multiplication operator is involved; and this yields the first basic analytic function, F_1. The equation for this basic analytic function is

$$F_1 = X_1*X_2.$$

The first basic algebraic function is now substituted into Equation (A-3), and the subfunction becomes

$$F_1+X_1. \tag{A-4}$$

The resulting subfunction contains only one operator and is defined as the second basic algebraic function, F_2, where

$$F_2 = F_1+X_1. \tag{A-5}$$

Equation (A-5) is the final form for the subfunction, and it is now substituted into the algebraic function being considered, Equation (A-2). The algebraic function becomes

$$F_2*SIN(X_2). \tag{A-6}$$

The hierarchy of operations table now indicates that the subfunction X_2 should now be evaluated. This subfunction is really a basic algebraic function with the implied operation of addition. Hence, the third basic algebraic function is

$$F_3 = + X_2. \tag{A-7}$$

The substitution of Equation (A-7) into Equation (A-6) yields

$$F_2*SIN\ F_3. \tag{A-8}$$

Hierarchy step number three yields the next basic algebraic function which is

$$F_4 = SIN\ F_3. \tag{A-9}$$

The substitution of Equation (A-9) into Equation (A-8) completes the reduction of the given algebraic function, Equation (A-1), to the basic algebraic function which is

$$F_2*F_4. \tag{A-10}$$

The results of decomposing the example function given in this section are summarized in Table IV, and it is noted that the algebraic function can be rewritten as a series of five basic algebraic functions.

TABLE IV

EXAMPLE OF DECOMPOSING AN ALGEBRAIC FUNCTION
TO A SERIES OF BASIC ALGEBRAIC FUNCTIONS

Algebraic Function Being Considered

(X(1)*X(2)+X(1)) * SIN(X(2))

Description as a Series of Basic Algebraic Functions

$$F_1 = X_1 * X_2$$

$$F_2 = F_1 + X_1$$

$$F_3 = + X_2$$

$$F_4 = SIN\ F_3$$

Algebraic Function = $F_2 * F_4$

Partial Derivatives of Basic Algebraic Functions

The partial derivatives for the basic algebraic functions are given in Table V. In this table the symbols U and V stand for algebraic functions; X_i stands for the variable X_i; and the symbols U_{X_i} and V_{X_i} stand for the partial derivatives of U and V with respect to X_i.

The bases for the ability to perform partial differentiation on the digital computer of an algebraic function have now been formulated, and they are:

1. An algebraic function can be decomposed into a series of basic algebraic functions.
2. The partial derivatives for the basic algebraic functions are given in Table V.
3. The partial derivatives for an algebraic function can be obtained by the proper combination of the partial derivatives of the basic algebraic functions.

Representing an Algebraic Function by a String of Integers

It is convenient in the computational program to represent the algebraic function as a string of integers. This can easily be done since there are only nineteen allowable operators and only three types of operands for the algebraic functions under consideration.

In Table VI a number code has been set up for each of

TABLE V

PARTIAL DERIVATIVES OF BASIC ALGEBRAIC FUNCTIONS

$$\frac{\partial(U + V)}{\partial X_i} = U_{X_i} + V_{X_i}$$

$$\frac{\partial(U * V)}{\partial X_i} = U_{X_i} * V + U * V_{X_i}$$

$$\frac{\partial(U ** V)}{\partial X_i} = V * U^{(V-1)} * U_{X_i} + U^V * (LOG_e\ U) * V_{X_i}$$

$$\frac{\partial(-U)}{\partial X_i} = - U_{X_i}$$

$$\frac{\partial\ EXP(U)}{\partial X_i} = EXP(U) U_{X_i}$$

$$\frac{\partial\ ALOG(U)}{\partial X_i} = U_{X_i} * U^{-1}$$

$$\frac{\partial\ ALOG10(U)}{\partial X_i} = (LOG_{10} e) U_{X_i} * U^{-1}$$

$$\frac{\partial\ SIN(U)}{\partial X_i} = COS(U)\ U_{X_i}$$

$$\frac{\partial\ COS(U)}{\partial X_i} = - SIN(U)\ U_{X_i}$$

$$\frac{\partial\ TAN(U)}{\partial X_i} = U_{X_i} * COS^{-2}(U)$$

$$\frac{\partial\ COTAN(U)}{\partial X_i} = - U_{X_i} * SIN^{-2}(U)$$

$$\frac{\partial\ ARSIN\ (U)}{\partial X_i} = U_{X_i} * (1 - U^2)^{-\frac{1}{2}}$$

$$\frac{\partial\ ARCOS\ (U)}{\partial X_i} = - U_{X_i} * (1 - U^2)^{-\frac{1}{2}}$$

TABLE V (Continued)

$\frac{\partial \text{ ATAN (U)}}{\partial X_i} = U_{X_i} * (1 + U^2)^{-1}$
$\frac{\partial \text{ SINH (U)}}{\partial X_i} = U_{X_i} * \text{COSH (U)}$
$\frac{\partial \text{ COSH (U)}}{\partial X_i} = U_{X_i} * \text{SINH (U)}$
$\frac{\partial \text{ TANH (U)}}{\partial X_i} = U_{X_i} * \text{COSH}^{-2} \text{ (U)}$
$\frac{\partial \text{ ABS (U)}}{\partial X_i} = U_{X_i} * U * \text{ABS}^{-1} \text{ (U)}$

TABLE VI[1]

NUMBER CODE USED TO REPRESENT
ALGEBRAIC FUNCTION

Operator	Code Number	Operator	Code Number
+	1	TAN	10
*	2	ATAN	11
**	3	COTAN	12
-	4	ALOG	13
		ALOG10	14
EXP	5	SINH	15
SIN	6	COSH	16
ARSIN	7	TANH	17
COS	8	SQRT	18
ARCOS	9	ABS	19

Operand	Code Number
Constant	51
Variable	52
Function	53

[1]The idea of using a number code to represent an algebraic function was first presented to the author by Stanley Wendt, a fellow graduate student.

the operators and operands.

It is possible to go directly from the description of the function as a series of basic algebraic functions to an integer string representation. This can be done in three simple steps:

1. The basic algebraic function representation is written as a row rather than as a column. The example given in Table IV then becomes

$$\overbrace{X_1 * X_2}^{F_1} \quad \overbrace{F_1 + X_1}^{F_2} \quad \overbrace{+X_2}^{F_3} \quad \overbrace{\text{SIN } F_3}^{F_4} \quad \overbrace{F_2 * F_4}^{\text{Algebraic Function}} \quad . \qquad \text{(A-11)}$$

2. As an aid for distinguishing where each basic function begins, the operator is written first. Hence, Equation (A-11) is rewritten

$$\overbrace{*X_1 X_2}^{F_1} \quad \overbrace{+F_1 X_1}^{F_2} \quad \overbrace{+X_2}^{F_3} \quad \overbrace{\text{SIN } F_3}^{F_4} \quad \overbrace{*F_2 F_4}^{\text{Algebraic Function}} \quad . \qquad \text{(A-12)}$$

3. The number code which is given in Table VI is now substituted into Equation (A-12). This results in an integer string representation for the algebraic function. The auxiliary integer string gives the subscripts for the operands.

	F_1		F_2		F_3	F_4	Algebraic Function	
Integer String -	2 52	52	1 53	52	1 52	6 53	2 53	53
Subscripts -	1	2	1	1	2	3	2	4

(A-13)

Partial Derivatives of Algebraic Functions

Represented as an Integer String

In Table VII the partial derivatives for the basic algebraic functions are given. If either U or V in the table are independent of the variable X_i, a considerable amount of simplification can be performed since many of the terms in the resulting derivative are multiplied by zero.

The performance of this simplification can be done as the derivatives of the algebraic functions are being formed if one adopts a dependency notation. If a dot is placed above an operand, it is dependent upon X_i. If there is no dot above the operand, it is independent of X_i.

In Table VII the partial derivatives for the basic functions are written in the notation equivalent to that used in Equation (A-12). In this table U, V, W, and Z represent the operands. A dot placed above them indicates that they are dependent. The subscript X_i represents the partial derivative with respect to X_i.

The column labeled "operands involved" will be discussed in detail later.

The procedure for determining if an operand is

TABLE VII

PARTIAL DERIVATIVES WHICH UTILIZE DEPENDENCY INFORMATION

	Operands Involved
$\frac{\partial\ (+\dot{U}V\dot{W}\dot{Z})}{\partial X_i} = + U_{X_i}\ Z_{X_i}$	None
$\frac{\partial(*\ \dot{U}V\dot{W}\dot{Z})}{\partial X_i} = \overbrace{*\ U_{X_i}\ VWZ}^{F_j}\ \overbrace{*\ UVWZ_{X_i}}^{F_{j+1}} + F_j\ F_{j+1}$	All if the number of dependent operands is greater than one. All except the dependent operand if there is only one dependent operand.
$\frac{\partial(**\dot{U}\dot{V})}{\partial X_i} = \overbrace{-1}^{F_j}\ \overbrace{+\ VF_j}^{F_{j+1}}\ \overbrace{**\ UF_{j+1}}^{F_{j+2}}$ etc.	
$* VF_{j+2} U_{X_i}\ **\ UV\ LOG_e\ U$	
$* F_{j+4}\ F_{j+5}\ V_{X_i} + F_{j+3} F_{j+6}$	U, V
$\frac{\partial(**\dot{U}V)}{\partial X_i} = -1 + V\ F_j\ **\ U\ F_{j+1}$	
$* V\ F_{j+2}\ U_{X_i}$	U, V
$\frac{\partial(**U\dot{V})}{\partial X_i} = **\ UV\ LOG_e U * F_j\ F_{j+1}\ V_{X_i}$	U, V
$\frac{\partial(-\dot{U})}{\partial X_i} = -U_{X_i}$	None
$\frac{\partial\ EXP\ \dot{U}}{\partial X_i} = EXP\ U * F_j\ U_{X_i}$	U
$\frac{\partial\ ALOG\ \dot{U}}{\partial X_i} = -1\ **\ U\ F_j * F_{j+1}\ U_{X_i}$	U

TABLE VII (Continued)

	Operands Involved
$\frac{\partial \text{ ALOG10 } \dot{U}}{\partial X_i} = -1 \text{ ** } U \; F_j + \text{LOG}_{10} e$	
$\text{* } F_{j+1} \; F_{j+2} \; U_{X_i}$	U
$\frac{\partial \text{ SIN } \dot{U}}{\partial X_i} = \text{COS } U \text{ * } F_j \; U_{X_i}$	U
$\frac{\partial \text{ COS } \dot{U}}{\partial X_i} = -1 \text{ SIN } U \text{ * } F_j \; F_{j+1} \; U_{X_i}$	U
$\frac{\partial \text{ TAN } \dot{U}}{\partial X_i} = -2 \text{ COS } U \text{ ** } F_{j+1} \; F_j$	
$\text{* } F_{j+2} \; U_{X_i}$	U
$\frac{\partial \text{ COTAN } \dot{U}}{\partial X_i} = -1 \; -2 \text{ SIN } U \text{ ** } F_{j+2} \; F_{j+1}$	
$\text{* } F_j F_{j+3} \; U_{X_i}$	U
$\frac{\partial \text{ ARSIN } \dot{U}}{\partial X_i} = -.50 + 2 \text{ ** } U \; F_{j+1}$	
$- F_{j+2} + 1 \; F_{j+3}$	
$\text{** } F_{j+4} \; F_j \text{ * } F_{j+5} \; U_{X_i}$	U
$\frac{\partial \text{ ARCOS } \dot{U}}{\partial X_i} = -.50 + 2 \text{ ** } U \; F_{j+1}$	
$- F_{j+2} + 1 \; F_{j+3}$	
$\text{** } F_{j+4} \; F_j \text{ * } F_{j+5} \; U_{X_i}$	
$- F_{j+6}$	U
$\frac{\partial \text{ ATAN } \dot{U}}{\partial X_i} = -1 + 2 \text{ ** } U \; F_{j+1} + 1 \; F_{j+2}$	
$\text{** } F_{j+3} \; F_j \text{ * } F_{j+4} \; U_{X_i}$	U

TABLE VII (Continued)

	Operands Involved
$\frac{\partial \text{ SINH } \dot{U}}{\partial X_i} = \text{COSH U} * F_j \, U_{X_i}$	U
$\frac{\partial \text{ COSH } \dot{U}}{\partial X_i} = \text{SINH U} * F_j \, U_{X_i}$	U
$\frac{\partial \text{ TANH } \dot{U}}{\partial X_i} = -2 \text{ COSHU} ** F_{j+1} \, F_j * F_{j+2} \, U_{X_i}$	U
$\frac{\partial \text{ ABS } \dot{U}}{\partial X_i} = -1 \text{ ABSU} ** F_{j+1} \, F_j * \text{U } F_{j+2} \, U_{X_i}$	U

dependent after the equation has been set up in a form analogous to Equation (A-12) involves three steps:

1. Steps two and three are performed on one basic function at a time starting from the left.
2. The dependency notation is set for the operand if
    a. it is equal to X_i or
    b. the operator for the basic function to which it refers is dependent.
3. The operator is defined to be dependent if any of its operands are dependent.

In Table VII there is a column which is labeled operands involved. Information in this column is used to determine which if any of the basic algebraic functions should be included in the equation for the partial derivative. This can be determined by a five-step process:

1. Steps two, three, and four are performed on one basic function at a time starting from the right side of the equation.
2. If the involvement notation for the operator has already been set, all of its operands are involved.
3. Even though the operator is not involved, the operands may still be involved as is indicated by the "operands involved" column of Table VII.
4. The operator of a basic function is defined

as being involved if an operand which refers to it is involved.

5. Only the basic functions whose operator has been designated as involved are included in the equation for the partial derivative.

The dependency and involvement information is used to keep the expression for the derivative of the function relatively free of unnecessary terms. The formal rules for setting up the equation for the derivative of a function will now be stated, and then an example problem will be worked.

In the rules which follow the subscript I refers to the number for the next basic function to be inserted into the derivative equation chain. N stands for the number of basic functions which are required for writing the derivative of the basic function being operated upon:

1. The equation is set up in a form analogous to Equation (A-12).
2. The operands which are dependent and the basic functions which are involved are determined by using the rules outlined previously.
3. An auxiliary chain is set up which contains the partial derivative of each of the operands.
4. The function which is being differentiated is operated upon one basic function at a time. The basic function at the extreme left of the

equation is considered first, and then one proceeds toward the right.

5. Is the basic function involved?

   a. Yes. It is inserted into the derivative chain; I is set equal to I + 1; and one proceeds to step 6.

   b. No. The following changes are performed on the portion of the equation chain and the auxiliary chain which lies to the right of the basic function which is being operated upon. F_j is changed to F_{j-1}, and $\frac{\partial F_j}{\partial X_i}$ is changed to $\frac{F_{j-1}}{X_i}$ if j is greater than I. One then proceeds to step 6.

6. Are any of the operands dependent?

   a. Yes. The derivative for the basic function is inserted into the derivative chain.

   The following changes are performed on the portion of the equation chain and the auxiliary chain which lie to the right of the basic function which is being operated upon. F_j is changed to F_{j+N} and $\frac{\partial F_j}{\partial X_i}$ is changed to $\frac{\partial F_{j+N}}{\partial X_i}$ if $j \geq I$. F_{I-1} is changed to F_{I-1+N}; $\frac{\partial F_{I-1}}{\partial X_i}$ is changed to F_{I+N-1}; I is set equal to I + N; and one returns to

step 4.

b. No. One returns to step 4.

The example function whose derivative with respect to X_2 will be formed using the above rules is given by Equation (A-12). If a dot is placed above each operator and operand which is dependent, the function will appear as follows:

$$\dot{*}\ X_1\ \dot{X}_2\ \dot{+}\ \dot{F}_1\ X_1\ \dot{+}\ \dot{X}_2\ \dot{SIN}\ \dot{F}_3\ \dot{*}\ \dot{F}_2\ \dot{F}_4. \qquad \text{(A-14)}$$

If a $\dot{\wedge}$ is used to indicate involvement, the equation becomes

$$\dot{\wedge}\ \dot{*}\ X_1\ \dot{X}_2\ \dot{+}\ \dot{F}_1\ X_1\ \dot{+}\ \dot{X}_2\ \dot{SIN}\ \dot{F}_3\ \dot{*}\ \dot{F}_2\ \dot{F}_4. \qquad \text{(A-15)}$$

Hence, it is noted that all, except the final one, of the basic functions will appear in the equation for the derivative.

Step three is now performed and yields

$$\dot{\wedge}\ \dot{*}\ X_1\ \dot{X}_2\ \dot{+}\ \dot{F}_1\ X_1\ \dot{+}\ \dot{X}_2\ \dot{SIN}\ \dot{F}_3\ \dot{*}\ \dot{F}_2\ \dot{F}_4 \qquad \text{(A-16)}$$

$$0\quad 1\quad \frac{\partial F_1}{\partial X_2}\quad 0\qquad 1\qquad \frac{\partial F_3}{\partial X_2}\quad \frac{\partial F_2}{\partial X_2}\quad \frac{\partial F_4}{\partial X_2}\ .$$

If steps five and six are performed on the first basic function of Equation (A-16), the equation which results for the first portion of the derivative is

$$\overbrace{*\ X_1\ X_2}^{F_1}\quad \overbrace{*\ X_1\ 1}^{F_2} \qquad \text{(A-17)}$$

After steps five and six have been completed, Equation (A-16) becomes

$$\overset{\wedge}{*}\ \overset{\wedge}{X_1}\ \overset{\wedge}{X_2}\ \overset{\wedge}{+}\ \overset{\wedge}{F_1}\ \overset{\wedge}{X_1}\ \overset{\wedge}{+}\ \overset{\wedge}{X_2}\ \overset{\wedge}{\text{SIN}}\ \overset{\wedge}{F_4}\ \overset{\wedge}{*}\ \overset{\wedge}{F_3}\ \overset{\wedge}{F_5} \qquad \text{(A-18)}$$

$$0\quad 1\quad \mathbf{F_2}\quad 0\quad 1\quad \frac{\partial F_4}{\partial X_2}\ \frac{\partial F_3}{\partial X_2}\ \frac{\partial F_5}{\partial X_2}\ .$$

The items which changed in Equation (A-18) are indicated in bold face type.

If steps five and six are now performed on the second basic function of Equation (A-18), the resulting equation for the derivative of the first two basic functions is

$$\overbrace{*\ X_1\ X_2}^{F_1}\quad \overbrace{*\ X_1\ 1}^{F_2}\quad \overbrace{+\ F_1\ X_1}^{F_3}\quad \overbrace{+\ F_2}^{F_4}. \qquad \text{(A-19)}$$

Equation (A-18) becomes

$$*\ X_1\ X_2\ +\ F_1\ X_1\ +\ X_2\ \text{SIN}\ F_5\ *\ F_4\ F_6 \qquad \text{(A-20)}$$

$$0\quad 1\quad F_2\quad 0\quad 1\quad \frac{\partial F_5}{\partial X_2}\quad F_4\ \frac{\partial F_6}{\partial X_2}\ .$$

In a similar manner rules five and six are applied to basic functions three, four, and five to obtain the final equation for the derivative.

Users Guide for Partial Differentiation Algorithm

The names of the subroutines which are needed for the partial differentiation algorithm are FORTIN, PARDIF, ISTORE, VAL, and DVAL.

The command which the user must supply within his Fortran IV program in order to read the equation which is given on the data card is

CALL FORTIN(NAME).

In the previous statement, NAME is any subscripted or simple integer variable. The computation algorithm will assign an integer constant to this integer variable which is symbolic information for where the equation is stored.

Subroutine FORTIN changes the equation which the user supplies on a data card to an integer string similar to that given by Equation (A-13). The equation which is supplied on the data card must be contained in columns 7 through 72. The independent variables must be either X or Y, and they must be subscripted variables. Any proper Fortran IV equation which can be written by using the operators given in Table II, floating point constants, E type constants, integer constants, and the subscripted independent variables X or Y is allowable. If the equation is continued on more than one data card, a C must be placed in column 6 of every data card except the last one.

Subroutine PARDIF will determine the partial derivative for an equation, and the necessary command is

CALL PARDIF (NAME1, NAME2,N).

In the previous command NAME1 is the integer name for the equation whose partial derivative with respect to X(N) is

to be taken. The resulting equation is assigned the integer name NAME2.

Subroutine ISTORE stores the integer strings for every equation which is read in from data cards or generated within a program.

A function subprogram with the title

VAL (NAME, X)

has been supplied. Upon return from this function subprogram VAL will contain the value for the equation whose symbolic name is NAME when it is evaluated for independent variable values which have been supplied through X. The dimension statement DIMENSION X(50) must be provided in all programs which use function subprogram VAL.

A function subprogram with the title

DVAL (NAME, X)

has been supplied. Upon return from this function subprogram DVAL will contain the double precision value for the equation whose symbolic name is NAME when it is evaluated for the double precision independent variable values which have been supplied through X. The following statements must be provided in all programs which use function subprogram DVAL.

```
DIMENSION X(50)
DOUBLE PRECISION DVAL, X.
```

In Figure 35 an example program is supplied which determines the value for the equation

$$3.2 * X(1) ** X(2) - 4.2E02 * X(2) \qquad \text{(A-21)}$$

when evaluated at

$$X(1) = 1.5$$
$$X(2) = -.72.$$

The partial derivative of the equation with respect to X(2) is then determined and its value is evaluated at the point

$$X(1) = -23.0$$
$$X(2) = 7.2.$$

In Figure 36 an example program has been supplied for determining, evaluating, and writing all of the first and second partial derivatives for Equation (A-21).

```
0000000001111111111222222222233333333334444444444555555555566666666677
12345678901234567890123456789012345678901234567890123456789012345678901
      IMPLICIT INTEGER*2 (I-N)
      DIMENSION X(50)
  100 FORMAT (E15.7)
C
C     READ THE EQUATION WHICH IS ON THE DATA CARD
C
      CALL FORTIN(IF)
      X(1)=1.5
      X(2)=-.72
C
C     DETERMINE AND WRITE THE VALUE FOR THE EQUATION EVALUATED AT
C      THE POINT X(1)=1.5    X(2)=-.72
C
      FVAL=VAL(IF,X)
      WRITE (6,100) FVAL
C
C     DETERMINE THE PARTIAL DERIVITIVE FOR THE EQUATION WITH RESPECT
C     TO X(2)
C
      J=2
      CALL PARDIF (IF,IPARF,J)
      X(1)=-23.0
      X(2)=-.72
C
C     DETERMINE AND WRITE THE PARTIAL DERIVITIVE FOR THE EQUATION
C     EVALUATED AT X(1)=-23.0    X(2)=-.72
C
      FPAR=VAL(IPARF,X)
      WRITE (6,100) FPAR
      CALL EXIT
      END
```

```
3.2*X(1)**X(2)-4.2E02*X(2)
```

Figure 35. Example of Calculating a Partial Derivative

```
00000000011111111112222222222333333333344444444445555555556
12345678901234567890123456789012345678901234567890123456789
      IMPLICIT INTEGER*2 (I-N)
      DIMENSION X(50)
      DIMENSION IF1(2),IF2(2,2)
  100 FORMAT (4E15.7)
      X(1)=1.0
      X(2)=2.0
      CALL FORTIN (IF)
      VALIF=VAL(IF,X)
      WRITE (6,100) VALIF
      DO 1 I=1,2
      CALL PARDIF (IF,IF1(I),I)
      VALIFI=VAL(IF1(I),X)
      WRITE (6,100) VALIFI
      DO 1 J=1,2
      CALL PARDIF (IF1(I),IF2(I,J),J)
      VALIF2=VAL(IF2(I,J),X)
      WRITE (6,100) VALIF2
    1 CONTINUE
      END




      3.2*X(1)**X(2)-4.2E02*X(2)
```

Figure 36. Example of Calculating First and Second Partial Derivatives

80/90 LISI

A-24

```
0000000001111111111222222222233333333334444444444555555555566666666677777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      SUBROUTINE FORTIN (NAME)                                                  001  006
C                                                                               002
C       I  -  POSITION OF FIN BEING CHECKED                                     003
C       J  -  POSITION OF NFIN BEING SET.                                       004
C       L  -  POSITION OF NFIN WITHIN THE AUXILARY FUNCTION BEING               005
C             CHECKED.                                                          006
C      ALPH  -  VARIABLE USED TO DECODE THE ALPHAMERIC FORTRAN 4 INPUT          007
C       FUNCTION.                                                               008
C      NFTOT  -  TOTAL LENGTH OF NFIN(1,J)                                      009
C      NFIN(I,J)                                                                010
C        NFIN(1,J)  -  INTEGER STRING FOR THE INPUT FUNCTION.  J INDI-          011
C                      CATES THE COLUMN BEING REFERED TO IN THE STRING.         012
C        NFIN(2,J)  -  THE SUBSCRIPT WHICH BELONGS TO OPERAND NFIN(1,J).        013
C                      IF NFIN(1,J) IS NOT AN OPERAND, THAN NFIN(2,J)=0.        014
C      NFUN(I,J)                                                                015
C        NFUN(1,J)  -  THE FUNCTION IN ITS FINAL DECODED FORM.                  016
C        NFUN(2,J)  -  SUBSCRIPT WHICH GOES ALONG WITH NFUN(1,J).               017
C      NFUNTO  -  TOTAL LENGTH OF NFUN(1,J)                                     018
C      C(I)  -  CONSTANT NUMBER I.                                              019
C      NPARR  -  COLUMN OF RIGHT HAND PARENTHESIS.                              020
C      NPARL  -  COLUMN OF LEFT HAND PARENTHESIS.                               021
C      NAME  -  FUNCTION NAME.                                                  022
C      FIN  -  ALPHAMERIC FORTRAN 4 INPUT OF THE FUNCTION.                      023
C      NC  -  CONSTANT NUMBER.                                                  024
C      NLEFT  -  NUMBER OF PLACES THE DECIMAL POINT IS TO THE LEFT              025
C        OF NUMERAL BEING CONSIDERED.                                           026
C                                                                               027
C     CODE FOR ALPH                                                             028
C        I  1      21  A      35  O                                            029  009
C        2  2      22  B      36  P                                            030  010
C        3  3      23  C      37  Q                                            031  011
C        4  4      24  D      38  R                                            032  012
C        5  5      25  E      39  S                                            033  013
C        6  6      26  F      40  T                                            034  014
C        7  7      27  G      41  U                                            035  015
C        8  8      28  H      42  V                                            036  016
C        9  9      29  I      43  W                                            037  017
C        10  0     30  J      44  X                                            038  018
C        11  .      31  K     45  Y                                            039  019
C        12  +     32  L      46  Z                                            040  020
C        13  -     33  M      47  (                                            041  021
C        14  *     34  N      48  )                                            042  022
C        15  /                                                                  043  023
C        16 THROUGH 20  CORRESPOND TO BLANK                                     044  024
C                                                                               045
C     CODE FOR NFIN                                                             046
C        1  +            11  TAN        47  (                                  047  027
C        2  *            12  ATAN       48  )                                  048
C        3  /            13  ABS         50   INTEGER CONSTANT                 049
C        4  **           14  ALOG       51  FLOATING POINT CONSTANT           050
C        5  -            15  ALOG10      52  VARIABLE                          051
C        6  EXP          16  SING       53  AUX. FUNCTION                     052  032
C        7  SIN          17  COSH                                             053  033
C        8  ARSIN        18  TANH                                             054
C        9  COS          19  COTAN                                            055
```

```
00000000011111111112222222222333333333344444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

C         10  ARCOS       20  SQRT                                          056
C      THE FINAL CODE FOR NFUN IS THE SAME AS THE ABOVE EXCEPT THAT          057
C          3  (FLOATING POINT VALUE)**(FLOATING POINT VALUE)                 058
C          4  (FLOATING POINT VALUE)**(INTEGER VALUE)                       059
C      REMARKS                                                              060
C       THE FORTRAN 4 FUNCTION THAT IS TO BE READ IN MUST BE WRITTEN        061
C       IN COLUMNS 7 THROUGH 72 OF THE DATA CARDS.  COLUMN 6 MUST BE        062
C       BLANK IF THE FUNCTION IS TATALY CONTAINED ON ONE CARD.  IF THE      063
C       FUNCTION IS CONTINUED ON MORE THAN ONE CARD, THAN THE LAST CARD     064
C       MUST HAVE A BLANK IN COLUMN 6 AND EACH OF THE PREVIOUS CARDS        065
C       OF THE CONTINUED FUNCTION MUST HAVE A 'C (FOR CONTINUATION) IN      066
C       COLUMN 6.                                                           067
C      MAXL  - MAXIMUM LENGTH OF INPUT FUNCTION (66 PER CARD).              068
       IMPLICIT INTEGER*2 (I-N)                                            069
       INTEGER*4 IC                                                        070
C      DIMENSION ALPH(48),CC(8),NFIN(2,MAXL),FIN(MAXL),NPARL(10)           071
       DIMENSION ALPH(48),NFIN(2,200),FIN(200),NPARL(10),CC(8),IC(2)       072
       DATA CC(1),CC(2),CC(3),CC(4),CC(5),CC(6),CC(7),CC(8),NC1,NC2       /073
      10.0,.50,-.50,1.0,-1.0,2.0,-2.0,.43429     ,9,0/                     074
       DATA ALPH/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,1H0,1H.,1H+,1H-,      075  039
      11H*,1H/,1H ,1H ,1H ,1H ,1H ,1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,        076  040
      21HI,1HJ,1HK,1HL,1HM,1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,            077  041
      31HW,1HX,1HY,1HZ,1H(,1H)/                                            078  042
C      MAXFUN  -  MAXIMUM LENGTH OF ANY FUNCTION TO BE GENERATED.          079
C      COMMON /BLK1/ NFUN(2,MAXFUN),NFUNTO                                 080
       COMMON /BLK1/  NFUN(2,300),NFUNTO                                   081
C      MAXC  - MAXIMUM NUMBER OF CONSTANTS ALLOWED.                        082
C      COMMON /BLK2/ C(MAXC),NC                                            083
       COMMON /BLK2/ C(200),NC                                             084
  100 FORMAT (5X,A1,66A1,8X)                                               085
  101 FORMAT (40I3)                                                        086  046
  102 FORMAT (' AN IMPROPER FORTRAN 4 FUNCTION HAS BEEN SUPPLIED ON        087  047
     1THE DATA CARD')                                                      088  048
  103 FORMAT (' NFIN(1,J)=')                                               089  049
  104 FORMAT (' NFIN(2,J)=')                                               090  050
  105 FORMAT (' NAUXF(1,J)=')                                              091  051
  106 FORMAT (' NAUXF(2,J)=')                                              092  052
  107 FORMAT (' NFUN(1,J)=')                                               093  053
  108 FORMAT (' NFUN(2,J)=')                                               094  054
  109 FORMAT (' C(',I5,')=',E15.7)                                         095  055
C      READ THE INPUT FUNCTION                                             096  056
       II=1                                                                097
       NFTOT=66                                                            098
   90 READ (5,100) CONT,(FIN(I),I=II,NFTOT)                                099
       WRITE (6,100) (FIN(I),I=II,NFTOT)                                   100
       IF (CONT.EQ.ALPH(20)) GO TO 91                                      101
       II=NFTOT+1                                                          102
       NFTOT=NFTOT+66                                                      103
       GO TO 90                                                            104
   91 CONTINUE                                                             105
C      SET CONSTANTS C(1) THROUGH C(8)                                     106
       NC2=NC2+1                                                           107
       IF (NC2.GT.1) GO TO 8                                               108
       NC=9                                                                109
       DO 6 I=1,8                                                          110
```

```
00000000011111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

    6 C(I)=CC(I)                                                          111
    8 CONTINUE                                                            112
C     REMOVE THE BLANKS                                                   113
      J=0                                                                 114
      DO 1 I=1,NFTOT                                                      115
      IF (FIN(I).EQ.ALPH(16)) GO TO 1                                     116
      J=J+1                                                               117
      FIN(J)=FIN(I)                                                       118
    1 CONTINUE                                                            119
      NFTOT=J                                                             120
C                                                                         121
C                                                                         122
C     CODE NFIN(1,J) AND NFIN(2,J)                                        123
C                                                                         124
C                                                                         125
      J=0                                                                 126
      IBASE =0                                                            127
      I=1                                                                 128
      NLC=1                                                               129
      NPARL(1)=0                                                          130
      NFUNTO=0                                                            131
    2 CONTINUE                                                            132
      J=J+1                                                               133
      IF (I.GT.NFTOT) GO TO 50                                            134
      NFIN(2,J)=0                                                         135
C     CHECK IF FIN(I) IS A VARIABLE                                       136
      IF (FIN(I).NE.ALPH(44).AND.FIN(I).NE.ALPH(45)) GO TO 4              137
      NFIN(1,J)=52                                                        138
C     DETERMINE THE VALUE FOR THE SUBSCRIPT                               139
      KK=I+2                                                              140
      J=J+1                                                               141
      IE=1                                                                142
      NIC=1                                                               143
      GO TO 41                                                            144
C     CODE (                                                              145
    4 IF (FIN(I).NE.ALPH(47)) GO TO 5                                     146
      NFIN(1,J)=47                                                        147
      NLC=NLC+1                                                           148
      NPARL(NLC)=J                                                        149
      I=I+1                                                               150
      GO TO 2                                                             151
C     CODE )                                                              152
    5 IF (FIN(I).NE.ALPH(48)) GO TO 10                                    153
      I=I+1                                                               154
      NFIN(1,J)=48                                                        155
      GO TO 50                                                            156
C     CODE +                                                              157
   10 IF (FIN(I).NE.ALPH(12)) GO TO 11                                    158
      NFIN(1,J)=1                                                         159
      I=I+1                                                               160
      GO TO 2                                                             161
C     CODE -                                                              162
   11 IF (FIN(I).NE.ALPH(13)) GO TO 12                                    163
      NFIN(1,J)=5                                                         164
      I=I+1                                                               165
```

```
0000000001111111111222222222233333333334444444444555555555566666666677777777788
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      GO TO 2                                                                  166
C     CODE *                                                                   167
   12 IF (FIN(I).NE.ALPH(14)) GO TO 14                                         168
      NFIN(1,J)=2                                                              169
C     CODE **                                                                  170
   13 I=I+1                                                                    171
      IF (FIN(I).NE.ALPH(14)) GO TO 2                                          172
      NFIN(1,J)=4                                                              173
      I=I+1                                                                    174
      GO TO 2                                                                  175
C     CODE /                                                                   176
   14 IF (FIN(I).NE.ALPH(15)) GO TO 15                                         177
      NFIN(1,J)=3                                                              178
      I=I+1                                                                    179
      GO TO 2                                                                  180
C     CODE EXP                                                                 181
   15 IF (FIN(I).NE.ALPH(25)) GO TO 21                                         182
      NFIN(1,J)=6                                                              183
      I=I+3                                                                    184
      GO TO 2                                                                  185
   21 IF (FIN(I).NE.ALPH(39)) GO TO 24                                         186
      IF (FIN(I+1).NE.ALPH(37)) GO TO 22                                       187
C     CODE SQRT                                                                188
      NFIN(1,J)=20                                                             189
      I=I+4                                                                    190
      GO TO 2                                                                  191
   22 IF (FIN(I+3).NE.ALPH(28)) GO TO 23                                       192
C     CODE SINH                                                                193
      NFIN(1,J)=16                                                             194
      I=I+4                                                                    195
      GO TO 2                                                                  196
C     CODE SIN                                                                 197
   23 NFIN(1,J)=7                                                              198
      I=I+3                                                                    199
      GO TO 2                                                                  200
   24 IF (FIN(I).NE.ALPH(21)) GO TO 35                                         201
      IF (FIN(I+1).NE.ALPH(40)) GO TO 25                                       202
C     CODE ATAN                                                                203
      NFIN(1,J)=12                                                             204
      I=I+4                                                                    205
      GO TO 2                                                                  206
   25 IF (FIN(I+1) .NE.ALPH(22)) GO TO 26                                      207
C     CODE ABS                                                                 208
      NFIN(1,J)=13                                                             209
      I=I+3                                                                    210
      GO TO 2                                                                  211
   26 IF (FIN(I+1).NE.ALPH(38)) GO TO 28                                       212
      IF (FIN(I+2).NE.ALPH(39)) GO TO 27                                       213
C     CODE ARSIN                                                               214
      NFIN(1,J)=8                                                              215
      I=I+5                                                                    216
      GO TO 2                                                                  217
C     CODE ARCOS                                                               218
   27 NFIN(1,J)=10                                                             219
      I=I+5                                                                    220
```

```
00000000011111111112222222222333333333344444444445555555556666666666777777777788
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      GO TO 2                                                                  221
   28 IF (FIN(I+4). NE.ALPH(1)) GO TO 29                                       222
C     CODE ALOG10                                                              223
      NFIN(1,J)=15                                                             224
      I=I+6                                                                    225
      GO TO 2                                                                  226
C     CODE ALOG                                                                227
   29 NFIN(1,J)=14                                                             228
      I=I+4                                                                    229
      GO TO 2                                                                  230
   35 IF (FIN(I).NE.ALPH(23)) GO TO 38                                         231
      IF (FIN(I+2).NE.ALPH(40)) GO TO 36                                       232
C     CODE COTAN                                                               233
      NFIN(1,J)=19                                                             234
      I=I+5                                                                    235
      GO TO 2                                                                  236
   36 IF (FIN(I+3).NE.ALPH(28)) GO TO 37                                       237
C     CODE COSH                                                                238
      NFIN(1,J)=17                                                             239
      I=I+4                                                                    240
      GO TO 2                                                                  241
C     CODE COS                                                                 242
   37 NFIN(1,J)=9                                                              243
      I=I+3                                                                    244
      GO TO 2                                                                  245
   38 IF (FIN(I).NE.ALPH(40)) GO TO 40                                         246
      IF (FIN(I+3).NE.ALPH(28)) GO TO 39                                       247
C     CODE TANH                                                                248
      NFIN(1,J)=18                                                             249
      I=I+4                                                                    250
      GO TO 2                                                                  251
C     CODE TAN                                                                 252
   39 NFIN(1,J)=11                                                             253
      I=I+3                                                                    254
      GO TO 2                                                                  255
C     IE=1 INTEGER.  IE=2 FLOATING POINT.  IE=3 E TYPE RAISED TO A             256
C     POSITIVE POWER.  IE=4  E TYPE RAISED TO NEGATIVE POWER.                  257
   40 CONTINUE                                                                 258
      KK=I                                                                     259
      IE=1                                                                     260
      NIC=1                                                                    261
   41 NLEFT=0                                                                  262
      IC(NIC)=0                                                                263
      DO 42 K=KK,NETOT                                                         264
      I=K                                                                      265
      IF (FIN(K).EQ.ALPH(11)) GO TO 43                                         266
      IF (FIN(K).EQ.ALPH(25)) GO TO 44                                         267
      IFIN=-1                                                                  268
      DO 93 M=1,9                                                              269
   93 IF (FIN(K).EQ.ALPH(M)) IFIN=M                                            270
      IF (FIN(K).EQ.ALPH(10)) IFIN=0                                           271
      IF (IFIN.EQ.-1) GO TO 46                                                 272
      NLEFT=NLEFT+1                                                            273
      IC(NIC)=IC(NIC)*10+IFIN                                                  274
      GO TO 42                                                                 275
```

```
00000000011111111112222222222333333333344444444445555555555666666666677777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

   43 IE=2                                                                  276
      NLEFT=0                                                               277
      NIC=NIC+1                                                             278
      IC(NIC)=0                                                             279
   42 CONTINUE                                                              280
      I=NETOT+1                                                             281
   46 GO TO (48,47,49,49),IE                                                282
   44 IE=3                                                                  283
      KK=I+2                                                                284
      IF (FIN(I+1).EQ.ALPH(13)) GO TO 45                                    285
      IF (FIN(I+1).NE.ALPH(12)) KK=I+1                                      286
      GO TO 47                                                              287
   45 IE=4                                                                  288
   47 RIC=IC(NIC)                                                           289
      RICM=IC(NIC-1)                                                        290
      C(NC)=RICM+RIC*10.0**(-NLEFT)                                         291
      WRITE (6,109) NC,C(NC)                                                292
      NFIN(1,J)=51                                                          293
      NFIN(2,J)=NC                                                          294
      NC=NC+1                                                               295
      NIC=NIC-1                                                             296
      IF (IE.EQ.2) GO TO 2                                                  297
      GO TO 41                                                              298
   48 NFIN(1,J)=50                                                          299
      NFIN(2,J)=IC(NIC)                                                     300
      IF (NFIN(1,J-1).NE.52) GO TO 2                                        301
      J=J-1                                                                 302
      NFIN(2,J)=IC(NIC)                                                     303
      I=I+1                                                                 304
      GO TO 2                                                               305
   49 ICCC=IC(NIC)                                                          306
      IF (IE.EQ.4)  ICCC=-ICCC                                              307
      C(NC-1)=C(NC-1)*10.0**ICCC                                            308
      NCM=NC-1                                                              309
      WRITE (6,109) NCM,C(NCM)                                              310
      GO TO 2                                                               311
   50 NPARR=J                                                               312
      J=NPARL(NLC)+1                                                        313
      L=NPARL(NLC)                                                          314
C                                                                           315
C     DEFINE FORTRAN 4 FUNCTIONS AS BASIC FUNCTIONS                         316
C                                                                           317
C                                                                           318
   58 L=L+1                                                                 319
      IF (L.EQ.NPARR) GO TO 61                                              320
      IF (NFIN(1,L).GT.20.OR.NFIN(1,L).LT.6) GO TO 60                       321
      NFUNTO=NFUNTO+1                                                       322
      NFUN(1,NFUNTO)=NFIN(1,L)                                              323
      NFUN(2,NFUNTO)=0                                                      324
      NFUNTO=NFUNTO+1                                                       325
      L=L+1                                                                 326
      NFUN(1,NFUNTO)=NFIN(1,L)                                              327
      NFUN(2,NFUNTO)=NFIN(2,L)                                              328
      IF (NFIN(1,L-1).NE.20) GO TO 62                                       329
      NFUN(1,NFUNTO-1)=3                                                    330
```

```
00000000011111111112222222222333333333344444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      NFUNTO=NFUNTO+1                                                    331
      NFUN(1,NFUNTO)=51                                                  332
      NFUN(2,NFUNTO)=2                                                   333
      GO TO 59                                                           334
   62 IF (NFIN(1,L-1).NE.19) GO TO 59                                    335
C     CHANGE COTAN TO 1.0/TAN                                            336
      NFUN(1,NFUNTO-1)=11                                                337
      NFUNTO=NFUNTO+1                                                    338
      IBASE=IBASE+1                                                      339
      NFUN(1,NFUNTO)=4                                                   340
      NFUN(2,NFUNTO)=0                                                   341
      NFUNTO=NFUNTO+1                                                    342
      NFUN(1,NFUNTO)=53                                                  343
      NFUN(2,NFUNTO)=IBASE                                               344
      NFUNTO=NFUNTO+1                                                    345
      NFUN(1,NFUNTO)=50                                                  346
      NFUN(2,NFUNTO)=-1                                                  347
   59 IBASE=IBASE+1                                                      348
      NFIN(1,J)=53                                                       349
      NFIN(2,J)=IBASE                                                    350
      J=J+1                                                              351
      GO TO 58                                                           352
   60 NFIN(1,J)=NFIN(1,L)                                                353
      NFIN(2,J)=NFIN(2,L)                                                354
      J=J+1                                                              355
      GO TO 58                                                           356
   61 NPARR=J                                                            357
C                                                                        358
C     REMOVE ** OPERATOR AND DEFINE IT AS A BASIC FUNCTION               359
      L=NPARL(NLC)                                                       360
      J=NPARL(NLC)+1                                                     361
   65 L=L+1                                                              362
      IF (L.EQ.NPARR) GO TO 67                                           363
      IF (NFIN(1,L).NE.4) GO TO 66                                       364
      NFUNTO=NFUNTO+1                                                    365
      NFUN(1,NFUNTO)=NFIN(1,L)                                           366
      NFUN(2,NFUNTO)=0                                                   367
      NFUNTO=NFUNTO+1                                                    368
      NFUN(1,NFUNTO)=NFIN(1,J-1)                                         369
      NFUN(2,NFUNTO)=NFIN(2,J-1)                                         370
      NFUNTO=NFUNTO+1                                                    371
      L=L+1                                                              372
      NFUN(1,NFUNTO)=NFIN(1,L)                                           373
      NFUN(2,NFUNTO)=NFIN(2,L)                                           374
      IF (NFIN(1,L).NE.50) NFUN(1,NFUNTO-2)=3                            375
      IBASE=IBASE+1                                                      376
      NFIN(1,J-1)=53                                                     377
      NFIN(2,J-1)=IBASE                                                  378
      GO TO 65                                                           379
   66 NFIN(1,J)=NFIN(1,L)                                                380
      NFIN(2,J)=NFIN(2,L)                                                381
      J=J+1                                                              382
      GO TO 65                                                           383
   67 NPARR=J                                                            384
C                                                                        385
```

```
00000000011111111112222222222333333333344444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

C       REMOVE THE / AND - OPERATIONS.                                          386
        L=NPARL(NLC)                                                            387
   76   L=L+1                                                                   388
        IF (L.EQ.NPARR) GO TO 69                                                389
        IF (NFIN(1,L).NE.5) GO TO 77                                            390
C       REMOVE THE - OPERATIONS AND DEFINE AS BASIC FUNCTIONS.                  391
        NFUNTO=NFUNTO+1                                                         392
        NFUN(1,NFUNTO)=NFIN(1,L)                                                393
        NFUN(2,NFUNTO)=0                                                        394
        NFUNTO=NFUNTO+1                                                         395
        L=L+1                                                                   396
        NFUN(1,NFUNTO)=NFIN(1,L)                                                397
        NFUN(2,NFUNTO)=NFIN(2,L)                                                398
        IBASE=IBASE+1                                                           399
        NFIN(1,L-1)=1                                                           400
        NFIN(1,L)=53                                                            401
        NFIN(2,L)=IBASE                                                         402
        GO TO 76                                                                403
C       REMOVE / OPERATORS AND CHANGE TO *                                      404
   77   IF (NFIN(1,L).NE.3) GO TO 76                                            405
        NFUNTO=NFUNTO+1                                                         406
        NFUN(1,NFUNTO)=4                                                        407
        NFUN(2,NFUNTO)=0                                                        408
        NFUNTO=NFUNTO+1                                                         409
        L=L+1                                                                   410
        NFUN(1,NFUNTO)=NFIN(1,L)                                                411
        NFUN(2,NFUNTO)=NFIN(2,L)                                                412
        NFUNTO=NFUNTO+1                                                         413
        NFUN(1,NFUNTO)=50                                                       414
        NFUN(2,NFUNTO)=-1                                                       415
        IBASE=IBASE+1                                                           416
        NFIN(1,L-1)=2                                                           417
        NFIN(1,L)=53                                                            418
        NFIN(2,L)=IBASE                                                         419
        GO TO 76                                                                420
C                                                                               421
C       REMOVE * OPERATORS AND DEFINE AS BASIC FUNCTIONS.                       422
   69   L=NPARL(NLC)                                                            423
        J=NPARL(NLC)+1                                                          424
   70   L=L+1                                                                   425
        IF (L.GE.NPARR) GO TO 75                                                426
        IF (NFIN(1,L).NE.2) GO TO 74                                            427
        NFUNTO=NFUNTO+1                                                         428
        NFUN(1,NFUNTO)=NFIN(1,L)                                                429
        NFUN(2,NFUNTO)=0                                                        430
        NFUNTO=NFUNTO+1                                                         431
        NFUN(1,NFUNTO)=NFIN(1,L-1)                                              432
        NFUN(2,NFUNTO)=NFIN(2,L-1)                                              433
   71   L=L+1                                                                   434
        NFUNTO=NFUNTO+1                                                         435
        NFUN(1,NFUNTO)=NFIN(1,L)                                                436
        NFUN(2,NFUNTO)=NFIN(2,L)                                                437
        L=L+1                                                                   438
        IF (L.EQ.NPARR) GO TO 72                                                439
        IF (NFIN(1,L).EQ.2) GO TO 71                                            440
```

```
00000000011111111112222222222333333333344444444455555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

   72 CONTINUE                                                          441
      IBASE=IBASE+1                                                     442
      NFIN(1,J-1)=53                                                    443
      NFIN(2,J-1)=IBASE                                                 444
      L=L-1                                                             445
      GO TO 70                                                          446
   74 NFIN(1,J)=NFIN(1,L)                                               447
      NFIN(2,J)=NFIN(2,L)                                               448
      J=J+1                                                             449
      GO TO 70                                                          450
   75 NPARR=J                                                           451
C                                                                       452
C     REMOVE THE + OPERATION AND DEFINE IT AS A BASIC FUNCTION.         453
   79 L=NPARL(NLC)+1                                                    454
      J=NPARL(NLC)                                                      455
      IF (NFIN(1,L).EQ.1) L=L+1                                         456
      IF (NPARR-L.EQ.1.AND.J.NE.0) GO TO 81                             457
      NFUNTO=NFUNTO+1                                                   458
      NFUN(1,NFUNTO)=1                                                  459
      NFUN(2,NFUNTO)=0                                                  460
   80 NFUNTO=NFUNTO+1                                                   461
      NFUN(1,NFUNTO)=NFIN(1,L)                                          462
      NFUN(2,NFUNTO)=NFIN(2,L)                                          463
      L=L+2                                                             464
      IF (L.LT.NPARR) GO TO 80                                          465
      IF (J.EQ.0) GO TO 95                                              466
      IBASE=IBASE+1                                                     467
      NLC=NLC-1                                                         468
      NFIN(1,J)=53                                                      469
      NFIN(2,J)=IBASE                                                   470
      GO TO 2                                                           471
   81 NLC=NLC-1                                                         472
      NFIN(1,J)=NFIN(1,L)                                               473
      NFIN(2,J)=NFIN(2,L)                                               474
      GO TO 2                                                           475
   95 CONTINUE                                                          476
      IBF=0                                                             477
      III=1                                                             478
  196 II=III                                                            479
      I2=NFUNTO                                                         480
      DO 198 I=II,I2                                                    481
      IF (NFUN(1,I).LE.20) IBF=IBF+1                                    482
      IF (NFUN(1,I).NE.1) GO TO 198                                     483
      IF (I+2.GT.NFUNTO) GO TO 199                                      484
      IF (NFUN(1,I+2).GT.20) GO TO 198                                  485
      NFS1=NFUN(1,I+1)                                                  486
      NFS2=NFUN(2,I+1)                                                  487
      NFUNTO=NFUNTO-2                                                   488
      III=I                                                             489
      DO 197 J=III,NFUNTO                                               490
      NFUN(1,J)=NFUN(1,J+2)                                             491
      NFUN(2,J)=NFUN(2,J+2)                                             492
      IF (NFUN(1,J).NE.53.OR.NFUN(2,J).LE.IBF) GO TO 201                493
      NFUN(2,J)=NFUN(2,J)-1                                             494
      GO TO 197                                                         495
```

```
000000000111111111122222222223333333333444444444455555555556666666667777777778
123456789012345678901234567890123456789012345678901234567890123456789012345678
  201 IF (NFUN(1,J).NE.53.OR.NFUN(2,J).NE.IBF) GO TO 197                    496
      NFUN(1,J)=NFS1                                                        497
      NFUN(2,J)=NFS2                                                        498
  197 CONTINUE                                                              499
      IBF=IBF-1                                                             500
      GO TO 196                                                             501
  198 CONTINUE                                                              502
      GO TO 200                                                             503
  199 IF (NFUNTO.EQ.2.OR.NFUN(1,NFUNTO).NE.53) GO TO 200                    504
      NFUNTO=NFUNTO-2                                                       505
  200 CONTINUE                                                              506
      WRITE (6,107)                                                         507
      WRITE (6,101) (NFUN(1,K),K=1,NFUNTO)                                  508
      WRITE (6,101) (NFUN(2,K),K=1,NFUNTO)                                  509
      I=1                                                                   510
      CALL ISTORE(NAME,I)                                                   511
      RETURN                                                                512
      END                                                                   513
```

```
00000000011111111112222222222333333333344444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      SUBROUTINE PARDIF (NAME1,NAME2,N)                                  001  001
C      NFUN(I,J)  -  FUNCTION TO BE DIFFERENTIATED.                      002  002
C        NFUN(1,J)  -  OPERATOR, OPERAND STRING.                         003  003
C        NFUN(2,4)  -  SUBSCRIPTS FOR THE OPERANDS.                      004  004
C      NFUNTO  -  TOTAL LENGTH OF NFUN.                                  005  005
C      NDERF(I,J)  -  DIFFERENTIAL OF THE FUNCTION                       006  006
C        NDERF(1,J)  -  OPERATOR, OPERAND STRING FOR DERIVITIVE.         007  007
C      NDERTO  -  TOTAL LENGTH OF NDERF                                  008  008
C     NDEROP(1,J)  -  CONTAINS DEPENDENCY INFORMATION IF NFUN(1,J) IS    009
C       AN OPERATOR.  CONTAINS A 0 IF THE OPERAND NFUN(1,J) IS NOT       010
C       DEPENDENT.  CONTAINS THE DERIVITIVE OF THE OPERAND IF OPERAND    011
C       IS DEPENDENT.                                                    012
C     NDEROP(2,J)  -  CONTAINS INVOLVEMENT INFORMATION IF NFUN(1,J) IS   013
C       AN OPERATOR.  OTHERWISE IT CONTAINS THE SUBSCRIPT FOR            014
C       NDEROP(1,J).                                                     015
C     IBASE(I)  -  CONTAINS THE COLUMN IN WHICH THE OPERATOR FOR BASIC   016
C       FUNCTION I IS LOCATED.                                           017
      IMPLICIT INTEGER*2 (I-N)                                           018
C     DIMENSION NDERF(2,MAXFUN),NDEROP(2,MAXFUN)                         019
      DIMENSION NDERF(2,600),NDEROP(2,600),IBASE(400)                    020
C     COMMON /BLK1/ NFUN(2,MAXFUN),NFUNTO                                021
      COMMON /BLK1/  NFUN(2,300),NFUNTO                                  022
      COMMON /BLK2/ C(200),NC                                            023
  100 FORMAT (' NDERF(1,J)=' )                                           024  020
  101 FORMAT (40I3)                                                      025  021
  102 FORMAT (' NDERF(2,J)=')                                            026  022
  103 FORMAT (' NFUN(1,J)=')                                             027  023
  104 FORMAT (' NFUN(2,J)=')                                             028  024
  105 FORMAT (' NDEROP(1,J)=')                                           029  025
  106 FORMAT (' NDEROP(2,J)=')                                           030  026
  109 FORMAT(4H NC=,I10,7H C(NC)=,E15.7)                                 031
      I=2                                                                032
      CALL ISTORE (NAME1,I)                                              033
      NFUN(1,NFUNTO+1)=0                                                 034
      NFUN(2,NFUNTO+1)=0                                                 035
      NDEROP(1,NFUNTO+1)=0                                               036
      NDEROP(2,NFUNTO+1)=0                                               037
C     SET UP NDEROP WHICH CONTAINS THE PARTIAL DERIVITIVES OF EACH OF    038  032
C       THE OPERANDES IN THE NFUN CHAIN.                                 039  033
C     ALSO SET DEPENDENCY INFORMATION.                                   040
      I=0                                                                041
      DO 4 J=1,NFUNTO                                                    042
      IF (NFUN(1,J).LE.20) GO TO 2                                       043
      IF (NFUN(1,J).NE.53) GO TO 8                                       044
      IF (NDEROP(1,IBASE(NFUN(2,J))).NE.0) GO TO 1                       045
    8 IF (NFUN(2,J).EQ.0) GO TO 3                                        046
      IF (NFUN(1,J).NE.52) GO TO 3                                       047
      IF (NFUN(2,J).NE.N) GO TO 3                                        048
      NDEROP(1,J)=51                                                     049
      NDEROP(2,J)=4                                                      050
      NDEROP(1,IBASE(I))=1                                               051
      GO TO 4                                                            052
    1 NDEROP(1,J)=54                                                     053
      NDEROP(2,J)=NFUN(2,J)                                              054  042
      NDEROP(1,IBASE(I))=1                                               055
```

```
00000000011111111112222222222333333333344444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      GO TO 4                                                          056
    2 I=I+1                                                            057
      IBASE(I)=J                                                       058
    3 NDEROP(1,J)=0                                                    059
      NDEROP(2,J)=0                                                    060
    4 CONTINUE                                                         061
C     SET INVOLVEMENT INFORMATION                                      062
      I=I+1                                                            063
      IBASE(I)=NFUNTO+1                                                064
   40 I=I-1                                                            065
      IF (I.EQ.0) GO TO 44                                             066
      J=IBASE(I)                                                       067
      IF (NDEROP(1,J).EQ.0.AND.NDEROP(2,J).EQ.0) GO TO 40              068
      NF=NFUN(1,J)                                                     069
      IF (NF.EQ.1.OR.NF.EQ.2.OR.NF.EQ.5) GO TO 42                      070
      J=J+1                                                            071
      IF (NFUN(1,J).NE.53) GO TO 41                                    072
      NDEROP(2,IBASE(NFUN(2,J)))=1                                     073
   41 IF (NF.NE.3) GO TO 40                                            074
      J=J+1                                                            075
      IF (NFUN(1,J).NE.53) GO TO 40                                    076
      NDEROP(2,IBASE(NFUN(2,J)))=1                                     077
      GO TO 40                                                         078
   42 IF (NDEROP(2,J).EQ.0.AND.(NF.EQ.1.OR.NF.EQ.5)) GO TO 40          079
      NTERMS=0                                                         080
      KK=IBASE(I)+1                                                    081
      KKK=IBASE(I+1)-1                                                 082
      DO 43 K=KK,KKK                                                   083
      IF (NDEROP(1,K).EQ.0) GO TO 48                                   084
      NTERMS=NTERMS+1                                                  085
      L=K                                                              086
   48 IF (NFUN(1,K).NE.53) GO TO 43                                    087
      NDEROP(2,IBASE(NFUN(2,K)))=1                                     088
   43 CONTINUE                                                         089
      IF (NDEROP(2,IBASE(I)).EQ.1) GO TO 40                            090
      IF (NF.NE.2.OR.NTERMS.GT.1) GO TO 40                             091
      IF (NFUN(1,L).NE.53) GO TO 40                                    092
      NDEROP(2,IBASE(NFUN(2,L)))=0                                     093
      GO TO 40                                                         094
   44 CONTINUE                                                         095
C                                                                      096
C                                                                      097
C                                                                      098
      NDERTO=0                                                         099  051
      IAUXDF=0                                                         100  053
      I=0                                                              101
   45 I=I+1                                                            102
      J=IBASE(I)                                                       103
      IF (J.GT.NFUNTO) GO TO 95                                        104
C     RESET THE SUBSCRIPTS IF THE BASIC FUNCTION IS NOT INVOLVED.      105
      IF (NDEROP(2,J).EQ.1) GO TO 46                                   106
      KK=IBASE(I+1)                                                    107
      DO 9 K=KK,NFUNTO                                                 108
      IF (NFUN(1,K).NE.53) GO TO 9                                     109
      IF (NFUN(2,K).LE.IAUXDF) GO TO 9                                 110
```

```
0000000001111111111222222222233333333334444444445555555556666666667777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456790

      NFUN(2,K)=NFUN(2,K)-1                                              111
      IF (NDEROP(1,K).EQ.0) GO TO 9                                      112
      NDEROP(2,K)=NDEROP(2,K)-1                                          113
    9 CONTINUE                                                           114
      GO TO 47                                                           115
C     SET THE BASIC FUNCTION CHAIN INTO THE DERIVITIVE CHAIN.            116  058
   46 NDERTO=NDERTO+1                                                    117
      NDERF(1,NDERTO)=NFUN(1,J)                                          118
      NDERF(2,NDERTO)=NFUN(2,J)                                          119
      IAUXDF=IAUXDF+1                                                    120
      JJ=J                                                               121
    7 JJ=JJ+1                                                            122
      IF (NFUN(1,JJ).LE.20) GO TO 6                                      123
      NDERTO=NDERTO+1                                                    124
      NDERF(1,NDERTO)=NFUN(1,JJ)                                         125
      NDERF(2,NDERTO)=NFUN(2,JJ)                                         126
      GO TO 7                                                            127
    6 CONTINUE                                                           128
C     EXPAND THE DERIVITIVE CHAIN SO THAT IT INCLUDES THE PARTIAL        129  074
C     DERIVITIVE OF THE AUXILARY FUNCTION PRESENTLY BEING CONSIDERED.    130  075
   47 IF (NDEROP(1,J).EQ.0) GO TO 45                                     131
      NF=NFUN(1,J)                                                       132  076
    5 GO TO (10,19,30,31,10,60,61,82,64,82,70,82,72,80,80,62,63,71),NF   133
C       +   AND   -                                                      134
   10 NDERTO=NDERTO+1                                                    135
      NDERF(1,NDERTO)=NF                                                 136
      NDERF(2,NDERTO)=0                                                  137
      IDIFF=1                                                            138
      JJ=J                                                               139
   11 JJ=JJ+1                                                           140
      IF (NFUN(1,JJ).LE.20) GO TO 1000                                   141
      IF (NDEROP(1,JJ).EQ.0) GO TO 11                                    142
      NDERTO=NDERTO+1                                                    143
      NDERF(1,NDERTO)=NDEROP(1,JJ)                                       144
      NDERF(2,NDERTO)=NDEROP(2,JJ)                                       145
      GO TO 11                                                           146
C       *                                                                147
   19 NTERMS=0                                                           148
   20 JJ=J                                                               149
   26 JJ=JJ+1                                                           150
      IF (NDEROP(1,JJ).GT.1) GO TO 21                                    151
      IF (NFUN(1,JJ).LE.20) GO TO 24                                     152
      GO TO 26                                                           153
   21 JJ=JJ-1                                                           154
      DO 22 K=J,JJ                                                       155
      NDERTO=NDERTO+1                                                    156
      NDERF(1,NDERTO)=NFUN(1,K)                                          157
   22 NDERF(2,NDERTO)=NFUN(2,K)                                          158
      NDERTO=NDERTO+1                                                    159
      JJ=JJ+1                                                            160
      NDERF(1,NDERTO)=NDEROP(1,JJ)                                       161
      NDERF(2,NDERTO)=NDEROP(2,JJ)                                       162
      NDEROP(1,JJ)=0                                                     163
      NTERMS=NTERMS+1                                                    164
   23 JJ=JJ+1                                                           165
```

```
0000000001111111111222222222233333333334444444445555555556666666667777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      IF (NFUN(1,JJ).LE.20) GO TO 20                                    166
      NDERTO=NDERTO+1                                                   167
      NDERF(1,NDERTO)=NFUN(1,JJ)                                        168
      NDERF(2,NDERTO)=NFUN(2,JJ)                                        169
      GO TO 23                                                          170
   24 NDERTO=NDERTO+1                                                   171
      NDERF(1,NDERTO)=1                                                 172
      NDERF(2,NDERTO)=0                                                 173
      IF (NTERMS.EQ.0) GO TO 26                                         174
      DO 25 K=1,NTERMS                                                  175
      NDERTO=NDERTO+1                                                   176
      NDERF(1,NDERTO)=53                                                177
   25 NDERF(2,NDERTO)=IAUXDF+K                                          178
      IDIFF=NTERMS+1                                                    179
      GO TO 1000                                                        180
C     **                                                                181
   30 IDIFF=0                                                           182
      IF (NDEROP(1,J+1).EQ.0) GO TO 35                                  183
      NDERTO=NDERTO+1                                                   184
      NDERF(1,NDERTO)=1                                                 185
      NDERF(2,NDERTO)=0                                                 186
      NDERF(1,NDERTO+1)=NFUN(1,J+2)                                     187
      NDERF(2,NDERTO+1)=NFUN(2,J+2)                                     188
      NDERTO=NDERTO+2                                                   189
      NDERF(1,NDERTO)=51                                                190
      NDERF(2,NDERTO)=5                                                 191
      NDERF(1,NDERTO+1)=3                                               192
      NDERF(1,NDEKTO+3)=53                                              193
      NDERF(2,NDERTO+3)=IAUXDF+1                                        194
      NDERF(2,NDERTO+5)=IAUXDF+2                                        195
      NDERF(1,NDERTO+6)=NFUN(1,J+2)                                     196
      NDERF(2,NDERTO+6)=NFUN(2,J+2)                                     197
      IDIFF=3                                                           198
      GO TO 32                                                          199
   31 IDIFF=0                                                           200
      IF (NDEROP(1,J+1).EQ.0) GO TO 35                                  201
      NDERF(1,NDERTO+1)=4                                               202
      NDERF(1,NDERTO+3)=50                                              203
      NDERF(2,NDERTO+3)=NFUN(2,J+2)-1                                   204
      IF (NDERF(2,NDERTO+3).EQ.0) GO TO 37                              205
      NDERF(2,NDERTO+5)=IAUXDF+1                                        206
      NDERF(1,NDERTO+6)=51                                              207
      C(NC)=NFUN(2,J+2)                                                 208
      WRITE (6,109) NC,C(NC)                                            209
      NC=NC+1                                                           210
      NDERF(2,NDERTO+6)=NC-1                                            211
      IDIFF=2                                                           212
   32 NDERF(2,NDERTO+1)=0                                               213
      NDERF(1,NDERTO+2)=NFUN(1,J+1)                                     214
      NDERF(2,NDERTO+2)=NFUN(2,J+1)                                     215
      NDERF(1,NDERTO+4)=2                                               216
      NDERF(2,NDERTO+4)=0                                               217
      NDERF(1,NDERTO+5)=53                                              218
      NDERF(1,NDERTO+7)=NDEROP(1,J+1)                                   219
      NDERF(2,NDERTO+7)=NDEROP(2,J+1)                                   220
```

```
00000000011111111112222222222333333333344444444445555555555666666666777777777788
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      NDERTO=NDERTO+7                                                          221
   35 IF (NDEROP(1,J+2).EQ.0) GO TO 36                                         222
      NDERF(1,NDERTO+1)=3                                                      223
      NDERF(2,NDERTO+1)=0                                                      224
      NDERF(1,NDERTO+2)=NFUN(1,J+1)                                            225
      NDERF(2,NDERTO+2)=NFUN(2,J+1)                                            226
      NDERF(1,NDERTO+2)=NFUN(1,J+2)                                            227
      NDERF(2,NDERTO+2)=NFUN(2,J+2)                                            228
      NDERF(1,NDERTO+3)=2                                                      229
      NDERF(2,NDERTO+3)=0                                                      230
      NDERF(1,NDERTO+4)=53                                                     231
      NDERF(2,NDERTO+4)=IAUXDF+IDIFF+1                                         232
      NDERF(1,NDERTO+5)=51                                                     233
      NDERF(2,NDERTO+5)=9                                                      234
      NDERF(1,NDERTO+6)=NDEROP(1,J+2)                                          235
      NDERF(2,NDERTO+6)=NDEROP(2,J+2)                                          236
      NDERTO=NDERTO+6                                                          237
      IDIFF=IDIFF+2                                                            238
   36 IF (NDEROP(1,J+1).EQ.0.OR.NDEROP(1,J+2).EQ.0) GO TO 1000                 239
      NDERF(1,NDERTO+1)=1                                                      240
      NDERF(2,NDERTO+1)=0                                                      241
      NDERF(1,NDERTO+2)=53                                                     242
      NDERF(2,NDERTO+2)=IAUXDF+IDIFF                                           243
      NDERF(1,NDERTO+3)=53                                                     244
      NDERF(2,NDERTO+3)=IAUXDF+IDIFF-2                                         245
      NDERTO=NDERTO+3                                                          246
      IDIFF=IDIFF+1                                                            247
      GO TO 1000                                                               248
   37 IDIFF=1                                                                  249
      NDERF(1,NDERTO+1)=1                                                      250
      NDERF(2,NDERTO+1)=0                                                      251
      NDERTO=NDERTO+2                                                          252
      NDERF(1,NDERTO)=NDEROP(1,J+1)                                            253
      NDERF(2,NDERTO)=NDEROP(2,J+1)                                            254
      GO TO 1000                                                               255
C     EXP                                                                      256
   60 NDERF(1,NDERTO+1 )=6                                                     257
      GO TO 65                                                                 258
C     SIN                                                                      259
   61 NDERF(1,NDERTO+1)=9                                                      260
      GO TO 65                                                                 261
C     SINH                                                                     262
   62 NDERF(1,NDERTO+1 )=17                                                    263
      GO TO 65                                                                 264
C     COSH                                                                     265
   63 NDERF(1,NDERTO+1)=16                                                     266
      GO TO 65                                                                 267
C     COS                                                                      268
   64 NDERF(1,NDERTO+1)=7                                                      269
   65 NDERF(2,NDERTO+1)=0                                                      270
      NDERTO=NDERTO+2                                                          271
      NDERF(1,NDERTO)=NFUN(1,J+1)                                              272
      NDERF(2,NDERTO)=NFUN(2,J+1)                                              273
      IDIFF=1                                                                  274
      IF (NF.EQ.9) GO TO 66                                                    275
```

```
00000000011111111112222222223333333334444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      IF (NDEROP(1,J+1).EQ.51) GO TO 1000                                  276
   66 NDERF(1,NDERTO+1)=2                                                  277
      NDERF(2,NDERTO+1)=0                                                  278
      NDERF(1,NDERTO+2)=53                                                 279
      NDERF(2,NDERTO+2)=IAUXDF+1                                           280
      NDERTO=NDERTO+3                                                      281
      NDERF(1,NDERTO)=NDEROP(1,J+1)                                        282
      NDERF(2,NDERTO)=NDEROP(2,J+1)                                        283
      IDIFF=2                                                              284
      IF (NF.NE.9) GO TO 1000                                              285
      IF (NDEROP(1,J+1).EQ.51) GO TO 67                                    286
      NDERTO=NDERTO+1                                                      287
   67 NDERF(1,NDERTO)=51                                                   288
      NDERF(2,NDERTO)=5                                                    289
      GO TO 1000                                                           290
C     TAN                                                                  291
   70 NDERF(1,NDERTO+1)=9                                                  292
      NDERF(2,NDERTO+5)=-2                                                 293
      IDIFF=3                                                              294
      GO TO 73                                                             295
C     TANH                                                                 296
   71 NDERF(1,NDERTO+1)=17                                                 297
      NDERF(2,NDERTO+5)=-2                                                 298
      IDIFF=3                                                              299
      GO TO 73                                                             300
C     ABS                                                                  301
   72 NDERF(1,NDERTO+1)=13                                                 302
      NDERF(2,NDERTO+5)=-1                                                 303
      NDERF(1,NDERTO+9)=NFUN(1,J+1)                                        304
      NDERF(2,NDERTO+9)=NFUN(2,J+1)                                        305
      IDIFF=3                                                              306
   73 NDERF(2,NDERTO+1)=0                                                  307
      NDERF(1,NDERTO+2)=NFUN(1,J+1)                                        308
      NDERF(2,NDERTO+2)=NFUN(2,J+1)                                        309
      NDERF(1,NDERTO+3)=4                                                  310
      NDERF(2,NDERTO+3)=0                                                  311
      NDERF(1,NDERTO+4)=53                                                 312
      NDERF(2,NDERTO+4)=IAUXDF+1                                           313
      NDERF(1,NDERTO+5)=50                                                 314
      NDERF(1,NDERTO+6)=2                                                  315
      NDERF(2,NDERTO+6)=0                                                  316
      NDERF(1,NDERTO+7)=NDEROP(1,J+1)                                      317
      NDERF(2,NDERTO+7)=NDEROP(2,J+1)                                      318
      NDERTO=NDERTO+8                                                      319
      NDERF(1,NDERTO)=53                                                   320
      NDERF(2,NDERTO)=IAUXDF+2                                             321
      IF (NF.EQ.13) NDERTO=NDERTO+1                                        322
      GO TO 1000                                                           323
C     ALOG                                                                 324
   80 NDERF(2,NDERTO+3)=-1                                                 325
      NDERF(1,NDERTO+6)=NDEROP(1,J+1)                                      326
      NDERF(2,NDERTO+6)=NDEROP(2,J+1)                                      327
      IDIFF=2                                                              328
      IF (NF.EQ.15) GO TO 81                                               329
      GO TO 85                                                             330
```

80/80 LIST A-40

```
00000000011111111112222222222333333333344444444455555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

C      ALOG10                                                                  331
   81  NDERF(1,NDERTO+7)=51                                                    332
       NDERF(2,NDERTO+7)=8                                                     333
       GO TO 85                                                                334
C      ARSIN                                                                   335
   82  NDERF(2,NDERTO+3)=2                                                     336
       NDERF(1,NDERTO+6)=51                                                    337
       NDERF(2,NDERTO+6)=5                                                     338
       NDERF(1,NDERTO+10)=3                                                    339
       NDERF(1,NDERTO+12)=51                                                   340
       NDERF(2,NDERTO+12)=3                                                    341
       IDIFF=5                                                                 342
       IF (NF.EQ.10) GO TO 83                                                  343
       IF (NF.EQ.12) GO TO 84                                                  344
       GO TO 85                                                                345
C      ARCOS                                                                   346
   83  NDERF(1,NDERTO+16)=51                                                   347
       NDERF(2,NDERTO+16)=5                                                    348
       GO TO 85                                                                349
C      ATAN                                                                    350
   84  NDERF(2,NDERTO+6)=4                                                     351
       NDERF(1,NDERTO+10)=4                                                    352
       NDERF(1,NDERTO+12)=50                                                   353
       NDERF(2,NDERTO+12)=-1                                                   354
   85  NDERF(1,NDERTO+1)=4                                                     355
       NDERF(2,NDERTO+1)=0                                                     356
       NDERF(1,NDERTO+2)=NFUN(1,J+1)                                           357
       NDERF(2,NDERTO+2)=NFUN(2,J+1)                                           358
       NDERF(1,NDERTO+3)=50                                                    359
       NDERF(1,NDERTO+4)=2                                                     360
       NDERF(2,NDERTO+4)=0                                                     361
       NDERF(1,NDERTO+5)=53                                                    362
       NDERF(2,NDERTO+5)=IAUXDF+1                                              363
       NDERTO=NDERTO+6                                                         364
       IF (NF.EQ.14) GO TO 1000                                                365
       IF (NF.NE.15) GO TO 86                                                  366
       NDERTO=NDERTO+1                                                         367
       GO TO 1000                                                              368
   86  NDERF(1,NDERTO+1)=1                                                     369
       NDERF(2,NDERTO+1)=0                                                     370
       NDERF(1,NDERTO+2)=53                                                    371
       NDERF(2,NDERTO+2)=IAUXDF+2                                              372
       NDERF(1,NDERTO+3)=51                                                    373
       NDERF(2,NDERTO+3)=4                                                     374
       NDERF(2,NDERTO+4)=0                                                     375
       NDERF(1,NDERTO+5)=53                                                    376
       NDERF(2,NDERTO+5)=IAUXDF+3                                              377
       NDERF(1,NDERTO+7)=2                                                     378
       NDERF(2,NDERTO+7)=0                                                     379
       NDERF(1,NDERTO+8)=53                                                    380
       NDERF(2,NDERTO+8)=IAUXDF+4                                              381
       NDERF(1,NDERTO+9)=NDEROP(1,J+1)                                         382
       NDERF(2,NDERTO+9)=NDEROP(2,J+1)                                         383
       NDERTO=NDERTO+9                                                         384
       IF (NF.NE.10) GO TO 1000                                                385
```

```
0000000001111111111222222222233333333334444444445555555556666666667777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456890
      NDERTO=NDERTO+1                                                    386
      GO TO 1000                                                         387
C     CHANGE THE PARTIAL DERIFITIVE TO ITS PROPER AUXILARY FUNCTION      388
C     NUMBER.  UPDATE THE AUXILARY FUNCTION NUMBERS AND THE NDEROPT(2,   389
C     JJ) NUMBERS.                                                       390
 1000 CONTINUE                                                           391
      KMAX=0                                                             392
      L=I                                                                393
   87 L=L+1                                                              394
      K=IBASE(L)                                                         395
      IF (K.EQ.NFUNTO+1) GO TO 90                                        396
      KMAX=IBASE(L+1)                                                    397
   88 K=K+1                                                              398
      IF (K.EQ.KMAX) GO TO 87                                            399
      IF (NFUN(1,K).NE.53) GO TO 88                                      400
      IF (NFUN(2,K).NE.IAUXDF) GO TO 91                                  401
      IF (NDEROP(1,K).NE.54) GO TO 88                                    402
      NDEROP(1,K)=53                                                     403
      NDEROP(2,K)=NDEROP(2,K)+IDIFF                                      404
      GO TO 88                                                           405
   91 IF (NFUN(2,K).LE.IAUXDF) GO TO 88                                  406
      NFUN(2,K)=NFUN(2,K)+IDIFF                                          407
      IF (NDEROP(1,K).NE.54) GO TO 88                                    408
      NDEROP(2,K)=NDEROP(2,K)+IDIFF                                      409
      GO TO 88                                                           410
   90 IAUXDF=IAUXDF+IDIFF                                                411
      GO TO 45                                                           412
   95 IF (NDERTO.EQ.0) GO TO 97                                          413
      DO 96 I=1,NDERTO                                                   414
      NFUN(1,I)=NDERF(1,I)                                               415
   96 NFUN(2,I)=NDERF(2,I)                                               416
      NFUNTO=NDERTO                                                      417
      GO TO 98                                                           418
   97 NFUN(1,1)=1                                                        419
      NFUN(2,1)=0                                                        420
      NFUNTO=2                                                           421
      NFUN(1,NFUNTO)=51                                                  422
      NFUN(2,NFUNTO)=1                                                   423
   98 CONTINUE                                                           424
      IBF=0                                                              425
      III=1                                                              426
  196 I1=III                                                             427
      I2=NFUNTO                                                          428
      DO 198 I=I1,I2                                                     429
      IF (NFUN(1,I).LE.20) IBF=IBF+1                                     430
      IF (NFUN(1,I).NE.1) GO TO 198                                      431
      IF (I+2.GT.NFUNTO) GO TO 199                                       432
      IF (NFUN(1,I+2).GT.20) GO TO 198                                   433
      NFS1=NFUN(1,I+1)                                                   434
      NFS2=NFUN(2,I+1)                                                   435
      NFUNTO=NFUNTO-2                                                    436
      III=I                                                              437
      DO 197 J=III,NFUNTO                                                438
      NFUN(1,J)=NFUN(1,J+2)                                              439
      NFUN(2,J)=NFUN(2,J+2)                                              440
```

```
00000000011111111112222222222333333333344444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      IF (NFUN(1,J).NE.53.OR.NFUN(2,J).LE.IBF) GO TO 201                    441
      NFUN(2,J)=NFUN(2,J)-1                                                 442
      GO TO 197                                                             443
  201 IF (NFUN(1,J).NE.53.OR.NFUN(2,J).NE.IBF) GO TO 197                    444
      NFUN(1,J)=NFS1                                                        445
      NFUN(2,J)=NFS2                                                        446
  197 CONTINUE                                                              447
      IBF=IBF-1                                                             448
      GO TO 196                                                             449
  198 CONTINUE                                                              450
      GO TO 200                                                             451
  199 IF (NFUNTO.EQ.2.OR.NFUN(1,NFUNTO).NE.53) GO TO 200                    452
      NFUNTO=NFUNTO-2                                                       453
  200 CONTINUE                                                              454
      WRITE (6,105)                                                         455
      WRITE (6,101) (NFUN(1,JJJ),JJJ=1,NFUNTO)                              456
      WRITE (6,101) (NFUN(2,JJJ),JJJ=1,NFUNTO)                              457
      I=1                                                                   458
      CALL ISTORE (NAME2,I)                                                 459
      RETURN                                                                460  488
      END                                                                   461  489
```

```
00000000011111111112222222222333333333344444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      FUNCTION DVAL(NAME,X)                                                    001
C      NFUN(I,J) -  FUNCTION TO BE EVALUATED.                                  002  002
C      NFUNTO  -  TOTAL LENGTH OF NFUN                                         003  003
C      VALUE(I)  -  NUMERICAL VALUE OF AUXILARY FUNCTION I.                    004  004
      IMPLICIT INTEGER*2(I-N),REAL*8(A-H,O-Z)                                  005
      REAL*4 C                                                                 006
C     DIMENSION VALUE (MAXFUN/2),X(50)                                         007
      DIMENSION VALUE(100),X(10)                                               008
C     COMMON /BLK1/ NFUN(2,MAXFUN),NFUNTO                                      009
      COMMON /BLK1/  NFUN(2,300),NFUNTO                                        010
C     COMMON /BLK2/ C(MAXC),NC                                                 011
      COMMON /BLK2/ C(200),NC                                                  012
      I=2                                                                      013
      CALL ISTORE (NAME,I)                                                     014
      NFUN(1,NFUNTO+1)=20                                                      015
      J=0                                                                      016
      IAUXF=0                                                                  017
   50 J=J+1                                                                    018
   51 NF=NFUN(1,J)                                                             019
      IF (NFUN(1,J).EQ.20) GO TO 80                                            020
      IAUXF=IAUXF+1                                                            021
      J=J+1                                                                    022
      NOP=NFUN(1,J)-50                                                         023
      GO TO (52,53,54),NOP                                                     024
   52 VAL1=C(NFUN(2,J))                                                        025
      GO TO 55                                                                 026
   53 VAL1=X(NFUN(2,J))                                                        027
      GO TO 55                                                                 028
   54 VAL1=VALUE(NFUN(2,J) )                                                   029
   55 GO TO (56,56,57,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18),NF               030
   56 VALUE(IAUXF)=VAL1                                                        031
   57 J=J+1                                                                    032
      NOP=NFUN(1,J)-50                                                         033
      IF (NOP.LT.1) GO TO 51                                                   034
      GO TO (58,59,60),NOP                                                     035
   58 VAL2=C(NFUN(2,J))                                                        036
      GO TO (1,2,3),NF                                                         037
   59 VAL2=X(NFUN(2,J))                                                        038
      GO TO (1,2,3),NF                                                         039
   60 VAL2=VALUE(NFUN(2,J))                                                    040
      GO TO (1,2,3),NF                                                         041
    1 VALUE(IAUXF)=VALUE(IAUXF)+VAL2                                           042
      GO TO 57                                                                 043
    2 VALUE(IAUXF)=VALUE(IAUXF)*VAL2                                           044
      GO TO 57                                                                 045
    3 VALUE(IAUXF)=VAL1**VAL2                                                  046
      GO TO 50                                                                 047
    4 J=J+1                                                                    048
      VALUE(IAUXF)=VAL1**NFUN(2,J)                                             049
      GO TO 50                                                                 050
    5 VALUE(IAUXF)=-VAL1                                                       051
      GO TO 50                                                                 052
    6 VALUE(IAUXF)=DEXP  (VAL1)                                                053
      GO TO 50                                                                 054  040
    7 VALUE(IAUXF)=DSIN  (VAL1)                                                055
```

```
00000000011111111112222222222333333333444444444455555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      GO TO 50                                                          056  042
    8 VALUE(IAUXF)=DARSIN(VAL1)                                         057
      GO TO 50                                                          058  044
    9 VALUE(IAUXF)=DCOS  (VAL1)                                         059
      GO TO 50                                                          060  046
   10 VALUE(IAUXF)=DARCOS(VAL1)                                         061
      GO TO 50                                                          062  048
   11 VALUE(IAUXF)=DTAN  (VAL1)                                         063
      GO TO 50                                                          064  050
   12 VALUE(IAUXF)=DATAN (VAL1)                                         065
      GO TO 50                                                          066  052
   13 VALUE(IAUXF)=DABS(VAL1)                                           067
      GO TO 50                                                          068  054
   14 VALUE(IAUXF)=DLOG  (VAL1)                                         069
      GO TO 50                                                          070  056
   15 VALUE(IAUXF)=DLOG10(VAL1)                                         071
      GO TO 50                                                          072  058
   16 VALUE(IAUXF)=DSINH (VAL1)                                         073
      GO TO 50                                                          074  060
   17 VALUE(IAUXF)=DCOSH (VAL1)                                         075
      GO TO 50                                                          076  062
   18 VALUE(IAUXF)=DTANH (VAL1)                                         077
      GO TO 50                                                          078  064
   80 DVAL=VALUE(IAUXF)                                                 079
      RETURN                                                            080
      END                                                               081
```

```
00000000011111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      SUBROUTINE ISTORE (NAME,ILOGIC)                                    001
C      NFUN  -  FUNCTION WHICH IS BEING STORED OR RETRIEVED.             002  002
C      NFUNTO  - TOTAL LENGTH OF NFUN.                                   003  003
C      NAME  -  NAME OF FUNCTION BEING STORED OR RETRIEVED.              004  004
C      ILOGIC  -  1  IF FUNCTION IS TO BE STORED.  2  IF FUNCTION IS TO  005  005
C       RETRIEVED.  3 NSTOR IS BEING PUNCHED.  4 NSTOR IS BEING READ     006
C       FROM DATA CARDS.                                                 007
C      LOCAT (I,J)  -                                                    008  007
C          I =  NAME OF FUNCTION BEING STORED OR RETRIVED.               009  008
C          LOCAT (I,1) =  COLUMN IN WHICH FUNCTION BEGINS.               010  009
C          LOCAT(I,2)  =  NFUNTO.                                        011  010
C      NSTOR  -  MATRIX WHICH HAS ALL FUNCTIONS STORED IN IT.            012  011
C      NSTORT  -  TOTAL LENGTH OF NSTOR                                  013  012
C     THIS SUBROUTINE IS SET UP TO STORE UP TO 100 FUNCTIONS WHOSE       014
C     AVERAGE LENGTH IS ONE FIFTH OF THE MAXIMUM ALLOWED LENGTH.         015
      IMPLICIT INTEGER*2 (I-N)                                           016
C     DIMENSION NSTOR (2,20*MAXFUN),LOCAT(100,2)                         017
      DIMENSION NSTOR(2,5000),LOCAT(726,2)                               018
C     COMMON /BLK1/ NFUN(2,MAXFUN),NFUNTO                                019
      COMMON /BLK1/  NFUN(2,300),NFUNTO                                  020
      DATA NAMEI,LOCAT(1,1),LOCAT(1,2),NSTORT/1,1,1,1/                   021  014
      DATA NSTOR(1,1),NSTOR(2,1)/0,0/                                    022
  100 FORMAT (16I5)                                                      023
      GO TO (1,2,30,40), ILOGIC                                          024
C     INSERT FUNCTION INTO STORAGE                                       025  016
    1 NAMEI=NAMEI+1                                                      026
      NAME=NAMEI                                                         027  018
      LOCAT(NAME,1)=LOCAT(NAME-1,1)+LOCAT(NAME-1,2)                      028  019
      LOCAT(NAME,2)=NFUNTO                                               029  020
      DO 4 J=1,NFUNTO                                                    030  021
      JJ=NSTORT+J                                                        031  022
      NSTOR(1,JJ)=NFUN(1,J)                                              032  023
    4 NSTOR(2,JJ)=NFUN(2,J)                                              033  024
      NSTORT=JJ                                                          034  025
      RETURN                                                             035  026
C     RETRIEVE FUNCTION FROM STORAGE.                                    036  027
    2 NFUNTO=LOCAT(NAME,2)                                               037  028
      DO 3 J=1,NFUNTO                                                    038  029
      JJ=LOCAT(NAME,1)+J-1                                               039  030
      NFUN(1,J)=NSTOR(1,JJ)                                              040  031
    3 NFUN(2,J)=NSTOR(2,JJ)                                              041  032
      RETURN                                                             042  033
   30 LENGTH=LOCAT(NAMEI,1) +LOCAT(NAMEI,2)-1                            043
      WRITE (7,100) NAMEI,LENGTH                                         044
      WRITE (7,100)((LOCAT(I,J),J=1,2),I=1,NAMEI)                        045
      WRITE (7,100) ((NSTOR(I,J),J=1,LENGTH),I=1,2)                      046
      RETURN                                                             047
   40 READ(5,100) NAMEI,LENGTH                                           048
      READ(5,100)  ((LOCAT(I,J),J=1,2),I=1,NAMEI)                        049
      READ(5,100)   ((NSTOR(I,J),J=1,LENGTH),I=1,2)                      050
      RETURN                                                             051
      END                                                                052  034
```

```
00000000011111111112222222222333333333344444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      FUNCTION VAL(NAME,X)                                                    001  001
C      NFUN(I,J) -  FUNCTION TO BE EVALUATED.                                 002  002
C      NFUNTO  -  TOTAL LENGTH OF NFUN                                        003  003
C      VALUE(I)  -  NUMERICAL VALUE OF AUXILARY FUNCTION I.                   004  004
      IMPLICIT INTEGER*2 (I-N)                                                005
C     DIMENSION VALUE (MAXFUN/2),X(50)                                        006
      DIMENSION VALUE(100) ,X(50)                                             007
C     COMMON /BLK1/ NFUN(2,MAXFUN),NFUNTO                                     008
      COMMON /BLK1/  NFUN(2,300),NFUNTO                                       009
C     COMMON /BLK2/ C(MAXC),NC                                                010
      COMMON /BLK2/ C(200),NC                                                 011
      I=?                                                                     012
      CALL ISTORE (NAME,I)                                                    013
      NFUN(1,NFUNTO+1)=20                                                     014
      J=0                                                                     015
      IAUXF=0                                                                 016
   50 J=J+1                                                                   017
   51 NF=NFUN(1,J)                                                            018
      IF (NFUN(1,J).EQ.20) GO TO 80                                           019
      IAUXF=IAUXF+1                                                           020
      J=J+1                                                                   021
      NOP=NFUN(1,J)-50                                                        022
      GO TO (52,53,54),NOP                                                    023
   52 VAL1=C(NFUN(2,J))                                                       024
      GO TO 55                                                                025
   53 VAL1=X(NFUN(2,J))                                                       026
      GO TO 55                                                                027
   54 VAL1=VALUE(NFUN(2,J) )                                                  028
   55 GO TO (56,56,57,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18),NF              029
   56 VALUE(IAUXF)=VAL1                                                       030
   57 J=J+1                                                                   031
      NOP=NFUN(1,J)-50                                                        032
      IF (NOP.LT.1) GO TO 51                                                  033
      GO TO (58,59,60),NOP                                                    034
   58 VAL2=C(NFUN(2,J))                                                       035
      GO TO (1,2,3),NF                                                        036
   59 VAL2=X(NFUN(2,J))                                                       037
      GO TO (1,2,3),NF                                                        038
   60 VAL2=VALUE(NFUN(2,J))                                                   039
      GO TO (1,2,3),NF                                                        040
    1 VALUE(IAUXF)=VALUE(IAUXF)+VAL2                                          041
      GO TO 57                                                                042
    2 VALUE(IAUXF)=VALUE(IAUXF)*VAL2                                          043
      GO TO 57                                                                044
    3 VALUE(IAUXF)=VAL1**VAL2                                                 045
      GO TO 50                                                                046
    4 J=J+1                                                                   047
      VALUE(IAUXF)=VAL1**NFUN(2,J)                                            048
      GO TO 50                                                                049
    5 VALUE(IAUXF)=-VAL1                                                      050
      GO TO 50                                                                051
    6 VALUE(IAUXF)=EXP    (VAL1)                                              052  039
      GO TO 50                                                                053  040
    7 VALUE(IAUXF)=SIN    (VAL1)                                              054  041
      GO TO 50                                                                055  042
```

```
0000000001111111111222222222233333333334444444445555555555666666666677777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456790
```

```
    8 VALUE(IAUXF)=ARSIN (VAL1)                                    056  043
      GO TO 50                                                     057  044
    9 VALUE(IAUXF)=COS   (VAL1)                                    058  045
      GO TO 50                                                     059  046
   10 VALUE(IAUXF)=ARCOS (VAL1)                                    060  047
      GO TO 50                                                     061  048
   11 VALUE(IAUXF)=TAN   (VAL1)                                    062  049
      GO TO 50                                                     063  050
   12 VALUE(IAUXF)=ATAN  (VAL1)                                    064  051
      GO TO 50                                                     065  052
   13 VALUE(IAUXF)= ABS(VAL1)                                      066
      GO TO 50                                                     067  054
   14 VALUE(IAUXF)=ALOG  (VAL1)                                    068  055
      GO TO 50                                                     069  056
   15 VALUE(IAUXF)=ALOG10(VAL1)                                    070  057
      GO TO 50                                                     071  058
   16 VALUE(IAUXF)=SINH  (VAL1)                                    072  059
      GO TO 50                                                     073  060
   17 VALUE(IAUXF)=COSH  (VAL1)                                    074  061
      GO TO 50                                                     075  062
   18 VALUE(IAUXF)=TANH  (VAL1)                                    076  063
      GO TO 50                                                     077  064
   80 VAL=VALUE(IAUXF)                                             078
      RETURN                                                       079
      END                                                          080
```

APPENDIX B

COMPUTATIONAL ALGORITHM FOR METHOD OF OF SEEKING PRINCIPAL PLANES

In this section a detailed explanation of a computational algorithm which utilizes the method of seeking principal planes is presented. A listing of this program is given at the end of this appendix. It should be pointed out that although this computational program at first hand appears to be rather complicated, the user need only supply the equation for the function being minimized in Fortran IV on a data card.

A flow chart for the method of seeking principal planes has been presented in Figure 37, and a detailed explanation of this flow chart will now be given.

The first two blocks of the flow chart presented in Figure 37 are self-explanatory. In the third block any available computational algorithm for determining eigenvalues of the symmetric matrix of second partial derivatives can be utilized. The number of positive eigenvalues will be denoted by P. Due to roundoff errors in the digial computer, if the matrix of second partial derivatives has a true eigenvalue equal to zero, the computational algorithm will probably not return an exact value of zero. Instead, the

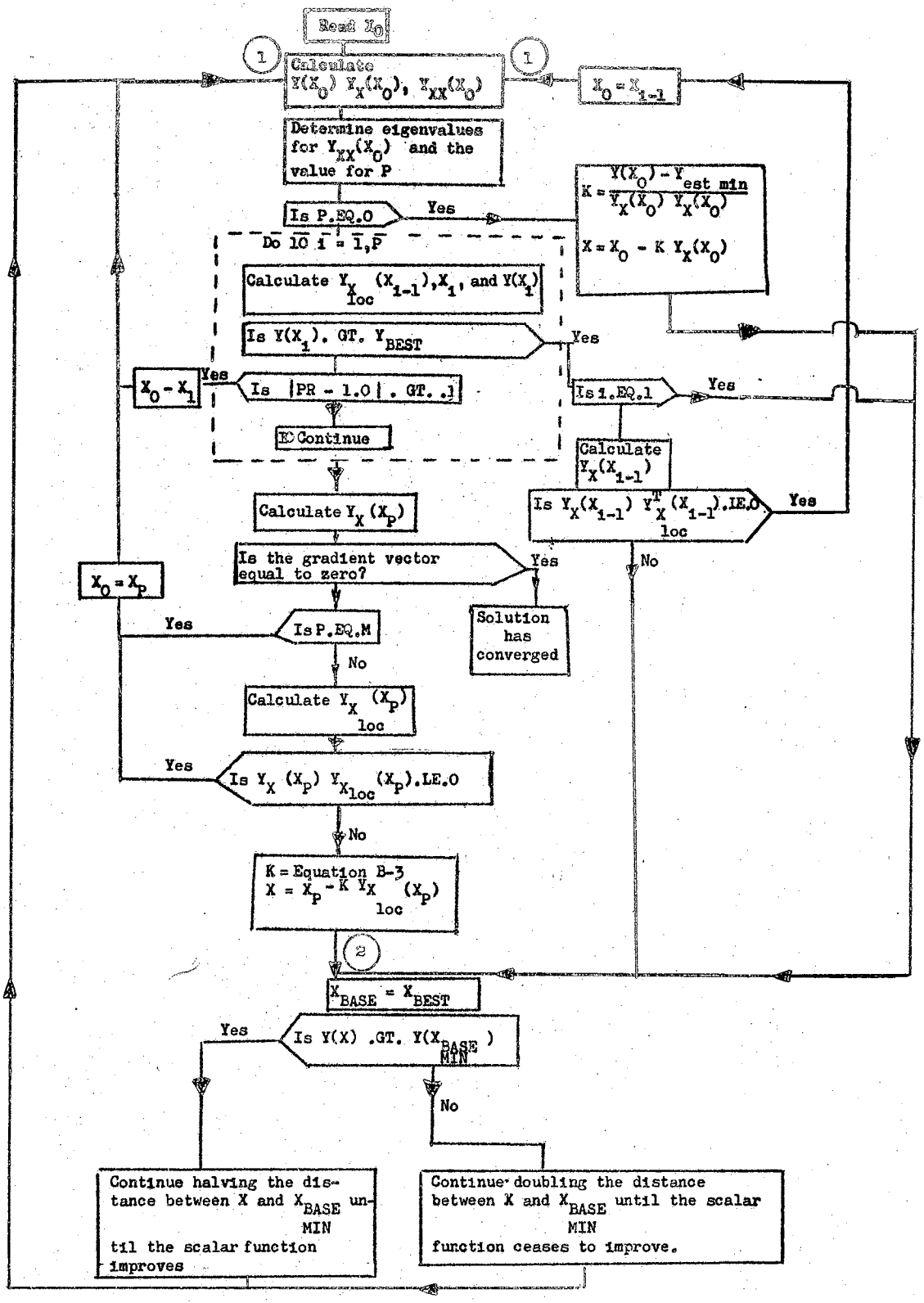


Figure 37. Flow Chart for Method of Seeking Principal Planes

value for the eigenvalue returned by the computational program might very likely be a small positive number. This difficulty is avoided by labeling an eigenvalue as zero if (1) it is positive, and (2) the value of the eigenvalue when multiplied by 10^6 is not larger than the largest eigenvalue.

The next block of the flow chart is a logic check to see if the number of positive eigenvalues, P, is equal to zero.

When the number of positive eigenvalues is zero, a linear search in the negative gradient direction for a minimum is performed. The location of this minimum need not be exact. The first problem which must be resolved in searching for the general area of this minimum is the magnitude of the initial step in the negative gradient direction. In the input data the user will provide an estimated minimum for the scalar function. The initial step used is such that if the gradient remained constant, then the scalar function would be reduced from its present value to the estimated minimum in one step. The resulting equations are:

$$K = \frac{Y(X_0) - Y_{\text{est min}}}{Y_X(X_0)\ Y_X(X_0)^T} \qquad \text{(B-1)}$$

and

$$X = X_0 - K\ Y_X(X_0). \qquad \text{(B-2)}$$

If $Y(X_0)$ is already smaller than the estimated minimum, the algorithm is terminated, and a message which describes

why the algorithm was terminated is written. Otherwise, the algorithm continues to block 2. This is a block for determining the approximate minimum point for the scalar function along the predicted negative gradient direction, and it will now be described in detail.

The base point from which a minimum along the predicted negative gradient direction is being sought is called X_{BASE}. X_{BEST} is the value for the independent variable at the point where the smallest value for Y(X) has thus far been obtained.

If the value for Y(X) is greater than $Y(X_{BEST})$, the distance between X and X_{BASE} is continually halved until a value for X is obtained such that Y(X) is less than $Y(X_{BASE})$. X_0 is set equal to X and the algorithm returns to block 1.

If the value for Y(X) is less than $Y(X_{BASE})$, the distance between X and X_{BASE} is continually doubled until a point X is reached at which the scalar function ceases to improve. X_0 is set equal to the value of X at which Y(X) was smallest, and the algorithm returns to block 1.

When the matrix of second partial derivatives has some positive eigenvalues, the algorithm proceeds into the iterative portion of the program which has been surrounded by a dotted line in Figure 37. This is the main part of the program, and it is within this area that the algorithm proceeds sequentially to the principal planes.

The subscript i is the iteration number and i - 1 refers

to the iteration just previous to the i^{th} iteration. The first block contained within the dotted lines indicates the calculation of (1) the gradient direction predicted by the truncated Taylor series at the point X_{i-1}, (2) the point, X_i, at which the predicted negative gradient intersects the desired principal plane, and (3) the magnitude of the scalar function evaluated at X_i.

When $Y(X_i)$ is greater than Y_{BEST}, it is desired to determine a new point such that the value of the scalar function is less than Y_{BEST}. Hence, the algorithm proceeds directly to block 2 if i is equal to 1.

If i is greater than 1 the direction being used in proceeding from point X_{i-1} to X_i is the gradient direction predicted by the truncated Taylor series. The predicted gradient direction will in general not be collinear with the true gradient direction. The value of the directional derivative along the predicted negative gradient direction will be negative (positive) if the dot product of the two gradient directions is positive (negative).

For the case of a positive directional derivative, the base point, X_0, is set equal to X_{i-1}, and the algorithm returns to block 1 of the flow chart.

For the case of a negative directional derivative a minimum is determined along predicted negative gradient direction by going to block 2.

The next step within the iterative loop indicates the calculation of a performance ratio, PR. This performance

ratio is defined as

$$PR = \frac{Y(X_i) - Y(X_0)}{Y_{loc}(X_i) - Y(X_0)}.$$

The reason for calculating the performance ratio is to determine if the point X_i is sufficiently close to the point X_0 so that the Taylor series approximation for the scalar function is accurate. If the truncated Taylor series described the function exactly at the point X_i, performance ratio would equal one.

When the absolute value of (PR-1) is greater than .1, the Taylor series approximation is no longer very accurate. X_0 is then set equal to X_i and the algorithm returns to block 1 of the flow chart.

After P successful steps to the principle planes, a check is performed to determine if the scalar function has converged to a minimum. Hence, the next step after having completed the iterative loop is to calculate the true value of the gradient at X_p. If the absolute value of each component of the gradient is less than 10^{-8}, it is assumed that the true gradient is equal to zero; and the solution has converged.

The solution can converge even though the matrix of second partial derivatives has some zero eigenvalues. If this situation occurs, it may indicate that the minimum which has been located is not unique and that there are an infinite number of points located in the region of X_p that

have the same minimum value for the scalar function. An example of this type is presented in Chapter II. It is also possible for the scalar function to have a unique minimum at a point where the matrix of second partial derivatives has some zero eigenvalues. An example problem with this characteristic has also been worked in Chapter II.

The number of positive eigenvalues, P, is either equal to or less than the number of independent variables, M.

If P is equal to M, the base point, X_0, is set equal to X_P; and the algorithm returns to block 1 of the flow diagram.

When P is less than M, it is necessary to determine the sign of the directional derivative in the predicted negative gradient direction.

A positive directional derivative indicates that a better value for the scalar function cannot be obtained by going in the predicted negative gradient direction. The base point, X_0, is set equal to X_P, and the algorithm returns to point 1 of the flow diagram.

A negative directional derivative indicates that a smaller value for the scalar function can be obtained by traveling in the predicted negative gradient direction.

The initial step size used is such that its distance is the same as the distance between the points X_0 and X_P. The magnitude for K and the equation for X then become

$$K^2 = \frac{[X_P - X_0]^T [X_P - X_0]}{Y_{X_{loc}}(X_P)\ Y_{X_{loc}}(X_P)^T} \qquad (B-3)$$

and

$$X = X_P - K Y_{X_{loc}}(X_P). \qquad (B-4)$$

The algorithm then proceeds to determine the approximate location of the minimum along this direction by going to block 2.

The computational algorithm for the method of seeking principal planes is composed of seven subroutines which are named SEEK, FCT, EIGEN, FORTIN, PARDIF, ISTORE, and DVAL.

The listings for subroutines SEEK, FCT, and EIGEN are given at the end of this appendix. The listings and descriptions of subroutines FORTIN, PARDIF, ISTORE, and DVAL are given in Appendix A.

A general flow chart for the subroutine SEEK is given in Figure 37.

The subroutine FCT is called by SEEK, and its purpose is to supply the values for the function, gradient, and matrix of second partial derivatives.

EIGEN is a subroutine which IBM provides through their Scientific Subroutine Package. This subroutine is used to determine the eigenvalues of the matrix of second partial derivatives.

Users Guide for the SEEK Algorithm

In order to use the SEEK algorithm, the user must supply the simple three card calling program which is given in Part A of Figure 38.

```
                    CALL    SEEK
                    CALL    EXIT
                    END
```

A. Example of the calling program which is required for the SEEK algorithm.

```
                1       1       2
Column -    5   0       5       0
First Data  2                 0.000
Card

                1       1       2
Column      5   0       5       0
Second          100.0 * (X(2) - X(1) **2) **2 + (1.0 - X(1))**2
Data Card
```

B. Data cards required to work the Rosenbrock curved valley problem using the SEEK algorithm.

Figure 38. Example Program for Determining the Minimum of an Algebraic Function

The SEEK algorithm requires two data cards.

The first data card has FORMAT (I5, D15.7) and contains the number of independent variables in column 5 and an estimate for the minimum of the function between columns 6 and 20.

The second data card must contain the right-hand side of the function which is to be minimized. It should be written in columns 7 through 72. If it is necessary to continue the equation onto more data cards, a C should be placed in column 6 of every one of the equation data cards except the last one.

The equation for Rosenbrock's curved valley is

```
Y = 100.0 * (X(2) - X(1) **2) **2 + (1.0 - X(1)) **2.
```

The data cards which are required for determining the minimum of this function are given in Part B of Figure 38.

```
00000000011111111112222222222333333333344444444445555555556666666666777777777
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      CALL SEEK                                                          001
      CALL EXIT                                                          002
      END                                                                003
      SUBROUTINE SEEK                                                    004
C                                                                        005
C     THIS PROGRAM WAS WRITTEN AND PROGRAMED BY DENNIS UNRUH AT THE      006
C     DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING, OKLAHOMA STATE 007
C     UNIVERSITY, STILLWATER, OKLAHOMA.  JULY 1969                       008
C                                                                        009
C       DELEPR -  THE PREDICTED DIFFERENCE BETWEEN THE VALUE OF F AT     010 0023
C        THE AXIS PLAIN AND F AT THE BASE POINT.                         011 0024
C       DEN -  SEE NUM/DEN.                                              012 0043
C       DIRECT -  IF THIS NUMBER IS POSITIVE, THAN THE PREDICTED         013 0038
C        GRADIENT DIRECTION AND ACTUAL GRADIENT DIRECTION ARE WITHIN     014 0039
C        90 DEGREES OF EACH OTHER.                                       015 0040
C       EIGVAL -  EIGENVALUE VECTOR OF GX.                               016 0037
C       F -  FUNCTION BEING MINIMIZED.  F=F(X)                           017 0015
C       FBASE -  VALUE OF F AT BASE POINT                                018 0027
C       FBEST -  VALUE OF F AT BEST POINT.                               019 0028
C     FESMIN -  ESTIMATED MINIMUM FOR THE SCALAR FUNCTION.               020
C       G -  GRADIENT VECTOR OF FUNCTION F.  FOR EXAMPLE G(2)=PARTIAL F/ 021 0016
C        PARTIAL X(2).                                                   022 0017
C     GMAG  -  MAGNITUDE OF GRADIENT AFTER NPOS STEPS TO AXIS PLANES.    023
C       GPRED -  PREDICTED VALUE OF GRADIENT AT AXIS PLAIN.              024 0022
C       GX -  MATRIX OF SECOND PARTIALS OF F.  I.E. GX(2,3)=PARTIAL G(2) 025 0018
C        /PARTIAL X(3).                                                  026 0019
C       GXS -  VECTOR WHICH CONTAINS GX IN SCIENTIFIC SUBROUTINE PACKAGE 027 0035
C        SYMMETRIC STORAGE MODE.                                         028 0036
C      N  -  THE NUMBER OF INDEPENDENT VARIABLES.                        029
C      NF  -  NUMBER OF TIMES F HAS BEEN CALCULATED.                     030
C      NG  -  NUMBER OF TIMES THE GRADIENT VECTOR HAS BEEN CALCULATED.   031
C      NGX  -  NUMBER OF TIMES MATRIX GX HAS BEEN CALCULATED.            032
C       NPOS -  NUMBER OF POSITIVE EIGENVALUES.                          033 0029
C       NUM/DEN -  THIS IS THE MAGNITUDE USED ON STEP NPOS+1 PROVIDED    034 0032
C        NPOS SUCCESSFUL STEPS TO THE AXIS PLANES HAVE BEEN MADE.  THIS  035 0033
C        IS USED ONLY IF NNEG IS NOT ZERO.                               036 0034
C       NXMAX -  MAXIMUM NUMBER OF INDEPENDENT VARIABLES.                037 0044
C       PERRAT -  RATIO BETWEEN THE PREDICTED FUNCTION CHANGE AND THE    038 0025
C        ACTUAL FUNCTION CHANGE.                                         039 0026
C       R -  DUMMY VARIABLE NEEDED FOR CALL STATEMENT OF EIGEN.          040 0045
C       WORK -  WORKING VARIABLE.                                        041 0046
C       X -  INDEPENDENT VARIABLE VECTOR.                                042 0012
C     XBAMIN -  BASE POINT FROM WHICH A MINIMUM IS BEING SOUGHT BY A     043
C       LINEAR SEARCH IN THE PREDICTED NEGATIVE GRADIENT DIRECTION.      044
C       XBASE -  THE VALUE OF THE INDEPENDENT VARIABLE VECTOR AT THE     045 0013
C        BASE POINT.                                                     046 0014
C       XBEST -  THE VALUE OF THE INDEPENDENT VARIABLE AT THE POINT      047 0020
C        WHERE THE CALCULATED VALUE OF F WAS SMALLEST.                   048 0021
C       XNPOS -  THE VALUE OF THE INDEPENDENT VARIABLE AFTER NPOS        049 0041
C        SUCCESSFUL STEPS TO THE AXIS PLANES.                            050 0042
      IMPLICIT INTEGER*2 (I-N),REAL*8 (A-H,O-Z)                          051
      INTEGER*4 MMM                                                      052
      REAL*8 NUM,DEN,MAG                                                 053
      NXMAX=10                                                           054 0047
C     DIMENSION X(NXMAX),G(NXMAX),XBEST(NXMAX),                          055
```

```
00000000011111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

C      XBASE(NXMAX),GXS((NXMAX*(NXMAX+1))/2),R(1),EIGVAL(NXMAX),                  056
C      GPRED(NXMAX),XBAMIN(NXMAX)                                                057
C      ,GXSS(NXMAX*(NXMAX+1))/2)                                                 058
       DIMENSION X(50),G(10),XBEST(10),XBASE(10),GXS(55),                        059
      1R(1),EIGVAL(10),GPRED(10),XBAMIN(50)                                      060
      2,GXSS(55)                                                                 061
       COMMON /BLK3/ IF,IG(10),IGXS(55),N                                        062
  100 FORMAT (8D15.7)                                                            063 0052
  101 FORMAT (' THE EIGENVALUE VECTOR IS')                                       064 0053
  102 FORMAT (6H NPOS=,I5)                                                       065
  103 FORMAT (4HONF=,I5,4H NG=,I5,5H NGX=,I5)                                    066 0055
  104 FORMAT (' INDEPENDENT VECTOR X IS')                                        067
  105 FORMAT (3H F=,D15.7)                                                       068 0057
  106 FORMAT (' THE GRADIENT VECTOR IS')                                         069 0058
  107 FORMAT (' GXS(I),I=1,IJMAX')                                               070
  108 FORMAT (8H PERRAT=,D15.7)                                                  071 0060
  109 FORMAT (' THE SOLUTION HAS CONVERGED')                                     072 0061
  110 FORMAT (8HODIRECT=,D15.7)                                                  073 0062
  111 FORMAT (7H FBEST=,D15.7)                                                   074 0063
  112 FORMAT (' THE VALUE OF X TO AXIS PLANE',I5,'   IS')                        075 0064
  113 FORMAT (' THE VALUE OF X AFTER HALVING IS')                                076 0065
  114 FORMAT (' THE VALUE OF X AFTER DOUBLING IS')                               077 0066
  115 FORMAT (I5,D15.7)                                                          078
  116 FORMAT (' THE FUNCTION VALUE IS ALREADY LESS THAN YOUR PREDICTED           079
     1MINIMUM')                                                                  080
  117 FORMAT (3H N=,I5,8H FESMIN=,D15.7)                                         081
  118 FORMAT (' GRADIENT EVALUATED AT XBAMIN')                                   082
       READ(5,115) N,FESMIN                                                      083
       READ(5,100) (X(I),I=1,N)                                                  084
       WRITE (6,117) N,FESMIN                                                    085
       IJMAX=(N*(N+1))/2                                                         086
       CALL FORTIN (IF)                                                          087
       DO 35 I=1,N                                                               088
       CALL PARDIF(IF,IG(I),I)                                                   089
       DO 35 J=I,N                                                               090
       IJ=((J-1)*J)/2+I                                                          091
   35  CALL PARDIF(IG(I),IGXS(IJ),J)                                             092
       NF=0                                                                      093 0067
       NG=0                                                                      094 0068
       NGX=0                                                                     095 0069
       I=1                                                                       096
       CALL FCT(X,F,G,GXS,NXMAX,I,NF,NG,NGX)                                     097
       WRITE (6,103) NF,NG,NGX                                                   098 0071
       WRITE (6,104)                                                             099 0072
       WRITE (6,100) (X(I),I=1,N)                                                100 0073
       WRITE (6,105) F                                                           101 0074
       DO 1 I=1,N                                                                102 0075
     1 XBEST(I)=X(I)                                                             103 0076
       FBEST=F                                                                   104 0077
     2 I=2                                                                       105
       CALL FCT(X,F,G,GXS,NXMAX,I,NF,NG,NGX)                                     106
       WRITE (6,103) NF,NG,NGX                                                   107 0079
       WRITE (6,106)                                                             108 0082
       WRITE (6,100) (G(I),I=1,N)                                                109 0083
     3 I=3                                                                       110
```

```
0000000001111111111222222222233333333344444444445555555555666666666677777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456789

      CALL FCT(X,F,G,GXS,NXMAX,I,NF,NG,NGX)                                  111
      WRITE (6,103) NF,NG,NGX                                                112 0085
      WRITE (6,107)                                                          113 0088
      WRITE (6,100) (GXS(I),I=1,IJMAX)                                       114
      DO 5 I=1,N                                                             115
    5 XBASE(I)=X(I)                                                          116
      FBASE=F                                                                117 0092
C                                                                            118 0093
C     DETERMINE EIGENVALUES OF GX AND THE NUMBER NPOS.                       119
C                                                                            120 0095
      DO 7 I=1,IJMAX                                                         121
    7 GXSS(I)=GXS(I)                                                         122
      MMM=N                                                                  123
      CALL EIGEN(GXSS,R,MMM,1)                                               124
      DO 6 I=1,N                                                             125 0101
      II=(I*(I+1))/2                                                         126 0102
    6 EIGVAL(I)=GXSS(II)                                                     127
      NPOS=N                                                                 128 0105
      DO 8 I=1,N                                                             129 0106
      IF (1.0D05*EIGVAL(I).GT.EIGVAL(1)) GO TO 8                             130
      NPOS=NPOS-1                                                            131 0108
    8 CONTINUE                                                               132 0115
      IF (EIGVAL(1).LE.0.0D0) NPOS=0                                         133
      WRITE (6,101)                                                          134 0117
      WRITE (6,100) (EIGVAL(I),I=1,N)                                        135 0118
      WRITE (6,102) NPOS                                                     136
C                                                                            137
C     IF NPOS IS EQUAL TO ZERO , A LINEAR SEARCH IS PERFORMED FOR A          138
C     MINIMUM IN THE NEGATIVE GRADIENT DIRECTION.                            139
C                                                                            140
      IF (NPOS.NE.0) GO TO 52                                                141
      DO 53 I=1,N                                                            142
   53 XBAMIN(I)=X(I)                                                         143
      NUM=FBEST-FESMIN                                                       144
      IF (NUM.LT.0.0D0) GO TO 30                                             145
      DEN=0.0D0                                                              146
      DO 50 I=1,N                                                            147
   50 DEN=DEN+G(I)*G(I)                                                      148
      DO 51 I=1,N                                                            149
   51 X(I)=X(I)-(NUM/DEN)*G(I)                                               150
      GO TO 23                                                               151
C                                                                            152 0120
C     DETERMINE VALUES FOR X TO THE SUCCESSIVE AXIS PLAINS.                  153 0121
C                                                                            154 0122
   52 DO 14 K=1,NPOS                                                         155
      DO 9 I=1,N                                                             156 0125
      GPRED(I)=G(I)                                                          157 0126
      DO 9 J=1,N                                                             158 0127
      IJ=((J-1)*J)/2+I                                                       159
      IF (J.LT.I) IJ=((I-1)*I)/2+J                                           160
    9 GPRED(I)=GPRED(I)+GXS(IJ)*(X(J)-XBASE(J))                              161
      DO 10 I=1,N                                                            162 0129
   10 X(I)=X(I)-GPRED(I)/EIGVAL(K)                                           163 0130
      I=1                                                                    164
      CALL FCT(X,F,G,GXS,NXMAX,I,NF,NG,NGX)                                  165
```

```
00000000011111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      WRITE (6,103) NF,NG,NGX                                          166 0132
      WRITE (6,112) K                                                  167 0133
      WRITE (6,100) (X(I),I=1,N)                                       168 0134
      WRITE (6,105) F                                                  169 0135
      IF (F.GT.FBEST.AND.K.EQ.1) GO TO 41                              170
      IF (F.GT.FBEST) GO TO 60                                         171
C                                                                      172 0140
C     CALCULATE PERRAT                                                 173 0141
C                                                                      174 0142
      DELFPR=0.0                                                       175 0143
      DO 13 I=1,N                                                      176 0144
      WORK=0.0                                                         177 0145
      DO 12 J=1,N                                                      178 0146
      IJ=((J-1)*J)/2+I                                                 179
      IF (J.LT.I) IJ=((I-1)*I)/2+J                                     180
   12 WORK=WORK+GXS(IJ)*(X(J)-XBASE(J))                                181
   13 DELFPR=DELFPR+(G(I)+.50*WORK)*(X(I)-XBASE(I))                    182 0148
      PERRAT =(F-FBASE)/DELFPR                                         183 0149
      WRITE (6,108) PERRAT                                             184 0150
      IF (DABS(PERRAT-1.0D0).GT..1D0) GO TO 2                          185
      DO 11 I=1,N                                                      186 0137
   11 XBEST(I)=X(I)                                                    187 0138
      FBEST=F                                                          188 0139
   14 CONTINUE                                                         189 0152
      DO 15 I=1,N                                                      190 0154
      GPRED(I)=G(I)                                                    191
      DO 15 J=1,N                                                      192 0156
      IJ=((J-1)*J)/2+I                                                 193
      IF (J.LT.I) IJ=((I-1)*I)/2+J                                     194
   15 GPRED(I)=GPRED(I)+GXS(IJ)*(X(J)-XBASE(J))                        195
      I=2                                                              196
      CALL FCT(X,F,G,GXS,NXMAX,I,NF,NG,NGX)                            197
      WRITE (6,103) NF,NG,NGX                                          198 0159
      WRITE (6,106)                                                    199 0162
      WRITE (6,100) (G(I),I=1,N)                                       200 0163
C                                                                      201
C     CHECK FOR CONVERGENCE.                                           202
C                                                                      203
      DO 45 I=1,N                                                      204
      IF (DABS(G(I)).GT.1.0D-8) GO TO 46                               205
   45 CONTINUE                                                         206
      GO TO 33                                                         207
   46 IF (NPOS.EQ.N) GO TO 3                                           208
      DIRECT=0.0                                                       209 0164
      DO 16 I=1,N                                                      210 0165
   16 DIRECT=DIRECT+G(I)*GPRED(I)                                      211 0166
      WRITE (6,110) DIRECT                                             212 0167
      IF (DIRECT.LE.0.0) GO TO 3                                       213 0168
      NUM=0.0                                                          214 0169
      DEN=0.0                                                          215 0170
      DO 17 I=1,N                                                      216 0171
      NUM=NUM+(X(I)-XBASE(I))**2                                       217 0172
   17 DEN=DEN+(GPRED(I))**2                                            218 0173
      MAG=DSQRT(NUM/DEN)                                               219
      DO 18 I=1,N                                                      220 0174
```

```
0000000001111111111222222222233333333334444444444555555555566666666667777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456789

   18 X(I)=X(I)-MAG*GPRED(I)                                          221
C                                                                     222
C     PERFORM A LINEAR SEARCH FOR A MINIMUM IN THE PREDICTED NEGATIVE  223
C     GRADIENT DIRECTION.                                             224
C                                                                     225
      I=1                                                             226
      CALL FCT(X,F,G,GXS,NXMAX,I,NF,NG,NGX)                           227
      WRITE (6,103) NF,NG,NGX                                         228 0178
      WRITE (6,104)                                                   229 0179
      WRITE (6,100) (X(I),I=1,N)                                      230 0180
      WRITE (6,105) F                                                 231 0181
   41 DO 42 I=1,N                                                     232
   42 XBAMIN(I)=XBEST(I)                                              233
      IF (F.GT.FBEST) GO TO 23                                        234 0182
C                                                                     235
C     CONTINUE DOUBLING THE DISTANCE UNTIL F CEASES TO BECOME BETTER.  236
C                                                                     237
   19 DO 20 I=1,N                                                     238 0183
   20 XBEST(I)=X(I)                                                   239 0184
      FBEST=F                                                         240 0185
      DO 21 I=1,N                                                     241 0186
   21 X(I)=XBAMIN(I)+2.0*(X(I)-XBAMIN(I))                             242
      I=1                                                             243
      CALL FCT(X,F,G,GXS,NXMAX,I,NF,NG,NGX)                           244
      WRITE (6,103) NF,NG,NGX                                         245 0189
      WRITE (6,114)                                                   246 0190
      WRITE (6,100) (X(I),I=1,N)                                      247 0191
      WRITE (6,105) F                                                 248 0192
      IF (F.LT.FBEST) GO TO 19                                        249 0193
      DO 22 I=1,N                                                     250 0194
   22 X(I)=XBEST(I)                                                   251 0195
      F=FBEST                                                         252 0196
      GO TO 2                                                         253 0197
C                                                                     254
C     CONTINUE HALVING THE DISTANCE UNTIL F BECOMES BETTER.           255
C                                                                     256
   23 DO 24 I=1,N                                                     257 0198
   24 X(I)=XBAMIN(I)+.50*(X(I)-XBAMIN(I))                             258
      I=1                                                             259
      CALL FCT(X,F,G,GXS,NXMAX,I,NF,NG,NGX)                           260
      WRITE (6,103) NF,NG,NGX                                         261 0201
      WRITE (6,113)                                                   262 0202
      WRITE (6,100) (X(I),I=1,N)                                      263 0203
      WRITE (6,105) F                                                 264 0204
      IF (F.GT.FBEST) GO TO 23                                        265 0205
      DO 25 I=1,N                                                     266 0206
   25 XBEST(I)=X(I)                                                   267 0207
      FBEST=F                                                         268 0208
      GO TO 2                                                         269 0209
   60 DO 61 I=1,N                                                     270
   61 XBAMIN(I)=XBEST(I)                                              271
      I=2                                                             272
      CALL FCT(XBAMIN,F,G,GXS,NXMAX,I,NF,NG,NGX)                      273
      WRITE (6,103) NF,NG,NGX                                         274
      WRITE (6,118)                                                   275
```

```
00000000011111111112222222222333333333344444444445555555555666666666677777777778
12345678901234567890123456789012345678901234567890123456789012345678901234557890

      WRITE (6,100) (G(I),I=1,N)                                          276
      DIRECT=0.0                                                          277
      DO 62 I=1,N                                                         278
   62 DIRECT=DIRECT+G(I)*GPRED(I)                                         279
      WRITE (6,110) DIRECT                                                280
      IF (DIRECT.GT.0.0D0) GO TO 23                                       281
      DO 63 I=1,N                                                         282
   63 X(I)=XBAMIN(I)                                                      283
      GO TO 3                                                             284
   30 WRITE (6,116)                                                       285
      CALL EXIT                                                           286
   33 WRITE (6,109)                                                       287
      CALL EXIT                                                           288 0241
      RETURN                                                              289 0242
      END                                                                 290 0243
      SUBROUTINE FCT(X,F,G,GXS,NXMAX,II,NF,NG,NGX)                        291
      IMPLICIT INTEGER*2 (I-N),REAL*8 (A-H,O-Z)                           292
      DIMENSION X(50),G(10),GXS(55)                                       293
      COMMON /BLK3/ IF,IG(10),IGXS(55),N                                  294
      GO TO (1,2,3),II                                                    295 0247
    1 NF=NF+1                                                             296 0248
C                                                                         297 0249
C     INSERT THE EQUATION FOR F AS A FUNCTION OF X.                       298 0250
C                                                                         299 0251
      F=DVAL(IF,X)                                                        300
      RETURN                                                              301 0253
C                                                                         302 0254
C                                                                         303 0255
C                                                                         304 0256
C     INSERT THE EQUATIONS FOR G IMMEDIATELY FOLLOWING THE STATEMENT      305 0257
C     2 NG=NG+1.                                                          306 0258
    2 NG=NG+1                                                             307 0259
      DO 4 I=1,N                                                          308
    4 G(I)=DVAL(IG(I),X)                                                  309
      RETURN                                                              310 0264
C                                                                         311 0265
C     INSERT THE EQUATIONS FOR GX IMMEDIATELY FOLLOWING THE STATEMENT     312 0266
C     3 NGX=NGX+1                                                         313 0267
    3 NGX=NGX+1                                                           314 0268
      DO 5 I=1,N                                                          315
      DO 5 J=I,N                                                          316
      IJ=((J-1)*J)/2+I                                                    317
    5 GXS(IJ)=DVAL(IGXS(IJ),X)                                            318
      RETURN                                                              319 0274
      END                                                                 320 0275
```

```
00000000011111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      SUBROUTINE EIGEN(A,R,N,MV)                                          EIGEN041
C                                                                         EIGEN001
C     ...................................................................EIGEN002
C                                                                         EIGEN003
C        SUBROUTINE EIGEN                                                 EIGEN004
C                                                                         EIGEN005
C        PURPOSE                                                          EIGEN006
C           COMPUTE EIGENVALUES AND EIGENVECTORS OF A REAL SYMMETRIC      EIGEN007
C           MATRIX                                                        EIGEN008
C                                                                         EIGEN009
C        USAGE                                                            EIGEN010
C           CALL EIGEN(A,R,N,MV)                                          EIGEN011
C                                                                         EIGEN012
C        DESCRIPTION OF PARAMETERS                                        EIGEN013
C           A - ORIGINAL MATRIX (SYMMETRIC), DESTROYED IN COMPUTATION.    EIGEN014
C               RESULTANT EIGENVALUES ARE DEVELOPED IN DIAGONAL OF        EIGEN015
C               MATRIX A IN DESCENDING ORDER.                             EIGEN016
C           R - RESULTANT MATRIX OF EIGENVECTORS (STORED COLUMNWISE,       EIGEN017
C               IN SAME SEQUENCE AS EIGENVALUES)                          EIGEN018
C           N - ORDER OF MATRICES A AND R                                 EIGEN019
C           MV- INPUT CODE                                                EIGEN020
C                 0   COMPUTE EIGENVALUES AND EIGENVECTORS               EIGEN021
C                 1   COMPUTE EIGENVALUES ONLY (R NEED NOT BE           EIGEN022
C                     DIMENSIONED BUT MUST STILL APPEAR IN CALLING      EIGEN023
C                     SEQUENCE)                                         EIGEN024
C                                                                         EIGEN025
C        REMARKS                                                          EIGEN026
C           ORIGINAL MATRIX A MUST BE REAL SYMMETRIC (STORAGE MODE=1)     EIGEN027
C           MATRIX A CANNOT BE IN THE SAME LOCATION AS MATRIX R           EIGEN028
C                                                                         EIGEN029
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED                    EIGEN030
C           NONE                                                          EIGEN031
C                                                                         EIGEN032
C        METHOD                                                           EIGEN033
C           DIAGONALIZATION METHOD ORIGINATED BY JACOBI AND ADAPTED       EIGEN034
C           BY VON NEUMANN FOR LARGE COMPUTERS AS FOUND IN 'MATHEMATICAL  EIGEN035
C           METHODS FOR DIGITAL COMPUTERS', EDITED BY A. RALSTON AND      EIGEN036
C           H.S. WILF, JOHN WILEY AND SONS, NEW YORK, 1962, CHAPTER 7     EIGEN037
C                                                                         EIGEN038
C     ...................................................................EIGEN039
C                                                                         EIGEN040
      DIMENSION A(1),R(1)                                                 EIGEN042
C                                                                         EIGEN043
C     ...................................................................EIGEN044
C                                                                         EIGEN045
C        IF A DOUBLE PRECISION VERSION OF THIS ROUTINE IS DESIRED, THE    EIGEN046
C        C IN COLUMN 1 SHOULD BE REMOVED FROM THE DOUBLE PRECISION        EIGEN047
C        STATEMENT WHICH FOLLOWS.                                         EIGEN048
C                                                                         EIGEN049
C     DOUBLE PRECISION A,R,ANORM,ANRMX,THR,X,Y,SINX,SINX2,COSX,           EIGEN050
C    1                 COSX2,SINCS,RANGE                                  EIGENM01
      DOUBLE PRECISION A,R,ANORM,ANRMX,THR,X,Y,SINX,SINX2,COSX,
     1    COSX2,SINCS,RANGE,DABS,DSQRT
C                                                                         EIGEN052
C        THE C MUST ALSO BE REMOVED FROM DOUBLE PRECISION STATEMENTS      EIGEN053
```

```
0000000001111111111222222222233333333334444444444555555555566666666667777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456790

C         APPEARING IN OTHER ROUTINES USED IN CONJUNCTION WITH THIS      EIGEN054
C         ROUTINE.                                                       EIGEN055
C                                                                        EIGEN056
C         THE DOUBLE PRECISION VERSION OF THIS SUBROUTINE MUST ALSO      EIGEN057
C         CONTAIN DOUBLE PRECISION FORTRAN FUNCTIONS.  SQRT IN STATEMENTSEIGEN058
C         40, 68, 75, AND 78 MUST BE CHANGED TO DSQRT.  ABS IN STATEMENT EIGEN059
C         62 MUST BE CHANGED TO DABS. THE CONSTANT IN STATEMENT 5 SHOULD EIGENM02
C         BE CHANGED TO 1.0D-12.                                         EIGENM03
C                                                                        EIGEN061
C         .....EIGEN062
C                                                                        EIGEN063
C         GENERATE IDENTITY MATRIX                                       EIGEN064
C                                                                        EIGEN065
    5 RANGE=1.0-12
      IF(MV-1) 10,25,10                                                  EIGEN066
   10 IQ=-N                                                              EIGEN067
      DO 20 J=1,N                                                        EIGEN068
      IQ=IQ+N                                                            EIGEN069
      DO 20 I=1,N                                                        EIGEN070
      IJ=IQ+I                                                            EIGEN071
      R(IJ)=0.0                                                          EIGEN072
      IF(I-J) 20,15,20                                                   EIGEN073
   15 R(IJ)=1.0                                                          EIGEN074
   20 CONTINUE                                                           EIGEN075
C                                                                        EIGEN076
C         COMPUTE INITIAL AND FINAL NORMS (ANORM AND ANORMX)             EIGEN077
C                                                                        EIGEN078
   25 ANORM=0.0                                                          EIGEN079
      DO 35 I=1,N                                                        EIGEN080
      DO 35 J=I,N                                                        EIGEN081
      IF(I-J) 30,35,30                                                   EIGEN082
   30 IA=I+(J*J-J)/2                                                     EIGEN083
      ANORM=ANORM+A(IA)*A(IA)                                            EIGEN084
   35 CONTINUE                                                           EIGEN085
      IF(ANORM) 165,165,40                                               EIGEN086
   40 ANORM=1.414*DSQRT(ANORM)
      ANRMX=ANORM*RANGE/FLOAT(N)                                         EIGENM05
C                                                                        EIGEN089
C         INITIALIZE INDICATORS AND COMPUTE THRESHOLD, THR               EIGEN090
C                                                                        EIGEN091
      IND=0                                                              EIGEN092
      THR=ANORM                                                          EIGEN093
   45 THR=THR/FLOAT(N)                                                   EIGEN094
   50 L=1                                                                EIGEN095
   55 M=L+1                                                              EIGEN096
C                                                                        EIGEN097
C         COMPUTE SIN AND COS                                            EIGEN098
C                                                                        EIGEN099
   60 MQ=(M*M-M)/2                                                       EIGEN100
      LQ=(L*L-L)/2                                                       EIGEN101
      LM=L+MQ                                                            EIGEN102
   62 IF(DABS(A(LM))-THR) 130,65,65
   65 IND=1                                                              EIGEN104
      LL=L+LQ                                                            EIGEN105
      MM=M+MQ                                                            EIGEN106
```

```
0000000001111111111222222222233333333334444444444555555555566666666667777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456789

      X=0.5*(A(LL)-A(MM))                                              EIGEN107
   68 Y=-A(LM)/DSQRT(A(LM)*A(LM)+X*X)
      IF(X) 70,75,75                                                   EIGEN109
   70 Y=-Y                                                             EIGEN110
   75 SINX=Y/DSQRT(2.0*(1.0+(DSQRT(1.0-Y*Y))))
      SINX2=SINX*SINX                                                  EIGEN112
   78 COSX=DSQRT(1.0-SINX2)
      COSX2=COSX*COSX                                                  EIGEN114
      SINCS =SINX*COSX                                                 EIGEN115
C                                                                      EIGEN116
C        ROTATE L AND M COLUMNS                                        EIGEN117
C                                                                      EIGEN118
      ILQ=N*(L-1)                                                      EIGEN119
      IMQ=N*(M-1)                                                      EIGEN120
      DO 125 I=1,N                                                     EIGEN121
      IQ=(I*I-I)/2                                                     EIGEN122
      IF(I-L) 80,115,80                                                EIGEN123
   80 IF(I-M) 85,115,90                                                EIGEN124
   85 IM=I+MQ                                                          EIGEN125
      GO TO 95                                                         EIGEN126
   90 IM=M+IQ                                                          EIGEN127
   95 IF(I-L) 100,105,105                                              EIGEN128
  100 IL=I+LQ                                                          EIGEN129
      GO TO 110                                                        EIGEN130
  105 IL=L+IQ                                                          EIGEN131
  110 X=A(IL)*COSX-A(IM)*SINX                                          EIGEN132
      A(IM)=A(IL)*SINX+A(IM)*COSX                                      EIGEN133
      A(IL)=X                                                          EIGEN134
  115 IF(MV-1) 120,125,120                                             EIGEN135
  120 ILR=ILQ+I                                                        EIGEN136
      IMR=IMQ+I                                                        EIGEN137
      X=R(ILR)*COSX-R(IMR)*SINX                                        EIGEN138
      R(IMR)=R(ILR)*SINX+R(IMR)*COSX                                   EIGEN139
      R(ILR)=X                                                         EIGEN140
  125 CONTINUE                                                         EIGEN141
      X=2.0*A(LM)*SINCS                                                EIGEN142
      Y=A(LL)*COSX2+A(MM)*SINX2-X                                      EIGEN143
      X=A(LL)*SINX2+A(MM)*COSX2+X                                      EIGEN144
      A(LM)=(A(LL)-A(MM))*SINCS+A(LM)*(COSX2-SINX2)                    EIGEN145
      A(LL)=Y                                                          EIGEN146
      A(MM)=X                                                          EIGEN147
C                                                                      EIGEN148
C        TESTS FOR COMPLETION                                          EIGEN149
C                                                                      EIGEN150
C        TEST FOR M = LAST COLUMN                                      EIGEN151
C                                                                      EIGEN152
  130 IF(M-N) 135,140,135                                              EIGEN153
  135 M=M+1                                                            EIGEN154
      GO TO 60                                                         EIGEN155
C                                                                      EIGEN156
C        TEST FOR L = SECOND FROM LAST COLUMN                          EIGEN157
C                                                                      EIGEN158
  140 IF(L-(N-1)) 145,150,145                                          EIGEN159
  145 L=L+1                                                            EIGEN160
      GO TO 55                                                         EIGEN161
```

```
00000000011111111112222222222333333333344444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

  150 IF(IND-1) 160,155,160                                          EIGEN162
  155 IND=0                                                          EIGEN163
      GO TO 50                                                       EIGEN164
C                                                                    EIGEN165
C        COMPARE THRESHOLD WITH FINAL NORM                           EIGEN166
C                                                                    EIGEN167
  160 IF(THR-ANRMX) 165,165,45                                       EIGEN168
C                                                                    EIGEN169
C        SORT EIGENVALUES AND EIGENVECTORS                           EIGEN170
C                                                                    EIGEN171
  165 IQ=-N                                                          EIGEN172
      DO 185 I=1,N                                                   EIGEN173
      IQ=IQ+N                                                        EIGEN174
      LL=I+(I*I-I)/2                                                 EIGEN175
      JQ=N*(I-2)                                                     EIGEN176
      DO 185 J=I,N                                                   EIGEN177
      JQ=JQ+N                                                        EIGEN178
      MM=J+(J*J-J)/2                                                 EIGEN179
      IF(A(LL)-A(MM)) 170,185,185                                    EIGEN180
  170 X=A(LL)                                                        EIGEN181
      A(LL)=A(MM)                                                    EIGEN182
      A(MM)=X                                                        EIGEN183
      IF(MV-1) 175,185,175                                           EIGEN184
  175 DO 180 K=1,N                                                   EIGEN185
      ILR=IQ+K                                                       EIGEN186
      IMR=JQ+K                                                       EIGEN187
      X=R(ILR)                                                       EIGEN188
      R(ILR)=R(IMR)                                                  EIGEN189
  180 R(IMR)=X                                                       EIGEN190
  185 CONTINUE                                                       EIGEN191
      RETURN                                                         EIGEN192
      END                                                            EIGEN193
```

APPENDIX C

CONDITIONS FOR CONVERGENCE OF GAUSS'S METHOD

The Newton-Raphson method as given by Equation (2-3) requires the computation of the entire matrix of second partial derivatives for the scalar function. This is a considerable amount of computational effort, and it would be desirable if the matrix of second partial derivatives could be approximated by a simpler equation.

Gauss's method does neglect some of the more complicated terms of the matrix of second partial derivatives. This can be seen by looking at Equations (2-42) and (2-43).

In this appendix the matrix of second partial derivatives is given by the sum of the symmetric matrices A and B.

$$Y_{XX} = A + B. \tag{C-1}$$

Gauss's equation can be obtained by neglecting the matrix B in Equation (C-1) and then substituting Equation (C-1) into Equation (2-3). The equation which results is

$$X_{i+1} - X_i = -A\,(X_i)^{-1}\,Y_X{}^T(X_i). \tag{C-2}$$

A matrix Q is now defined as

$$Q = A^{-1} B. \tag{C-3}$$

The conditions for convergence of Gauss's method will now be shown by a proof which is similar to one presented by Bekey and McGhee (19). The conclusions of this proof are:

1. Gauss's method will converge provided the starting point is sufficiently close to the extremum point, X_{EXT}, if all of the eigenvalues of the matrix $Q^T(X_{EXT})\ Q(X_{EXT})$ are less than one.
2. Gauss's method will diverge from the extremum point, X_{EXT}, if all of the eigenvalues of the matrix $Q^T(X_{EXT})\ Q(X_{EXT})$ are larger than one.
3. If some of the eigenvalues of $Q^T(X_{EXT})\ Q(X_{EXT})$ are larger than one and some are smaller than one, no final conclusions are drawn.

At an extremum point the gradient vector must be 0, and the Taylor series expansion for the scalar function about the point X_{EXT} becomes

$$Y(X) = Y(X_{EXT}) + \tfrac{1}{2}[X - X_{EXT}]^T\ Y_{XX}(X_{EXT})[X - X_{EXT}] + (\text{higher order terms}). \tag{C-4}$$

The gradient of Equation (C-4) is

$$Y_X(X) = [X - X_{EXT}]^T\ Y_{XX}(X_{EXT}) + (\text{higher order terms}). \tag{C-5}$$

The substitution of Equation (C-1) into Equation (C-5) yields

$$Y_X(X) = [X - X_{EXT}]^T [A(X_{EXT}) + B(X_{EXT})] + (\text{higher order terms}). \quad (C-6)$$

The substitution of Equation (C-6) into Equation (C-2) yields

$$X_{i+1} - X_i = -A(X_i)^{-1} [A(X_{EXT}) + B(X_{EXT})][X_i - X_{EXT}] + (\text{higher order terms}). \quad (C-7)$$

The equations for each of the elements in the matrix $A(X_i)^{-1}$ could be expanded in a Taylor series about the point X_{EXT} provided that A is not singular within a neighborhood of the extremum point. This yields

$$A(X_i)^{-1} = A \ (X_{EXT})^{-1} + (\text{higher order terms in } X_i - X_{EXT}). \quad (C-8)$$

Equation (C-8) can now be substituted into Equation (C-7). Since only values of X within an infinitesimal region of X_{EXT} are being considered, the higher order terms can be neglected, and the equation which results is

$$X_{i+1} - X_i = -[I + A \ (X_{EXT})^{-1} B(X_{EXT})][X_i - X_{EXT}]. \quad (C-9)$$

Equation (C-3) is substituted into Equation (C-9), and the resulting equation is rearranged to yield

$$X_{i+1} - X_{EXT} = -Q(X_{EXT})[X_i - X_{EXT}]. \qquad (C-10)$$

The coordinate transformation

$$W = E^T[X_i - X_{EXT}] \qquad (C-11)$$

is now introduced. In Equation (C-11), the matrix E is the eigenvector matrix of $Q^T(X_{EXT})\ Q(X_{EXT})$. Athans and Falb (9) demonstrate that

$$E^T E = I \qquad (C-12)$$

and

$$E^T\ Q^T(X_{EXT})\ Q(X_{EXT})\ E = \lambda. \qquad (C-13)$$

In Equation (C-12), λ is the eigenvalue matrix.

From Equations (C-11) and (C-12), it follows that

$$W^T W = [X_i - X_{EXT}]^T\ [X_i - X_{EXT}]. \qquad (C-14)$$

Equations (C-10) through (C-13) can be combined to yield

$$[X_{i+1} - X_{EXT}]^T [X_{i+1} - X_{EXT}] = W^T \lambda W. \qquad (C-15)$$

If all of the eigenvalues are less than one, Equations (C-15) and (C-14) can be combined to yield

$$[X_{i+1} - X_{EXT}]^T\ [X_{i+1} - X_{EXT}] = W^T \lambda W < W^T W$$

$$= [X_i - X_{EXT}]^T\ [X_i - X_{EXT}]. \qquad (C-16)$$

Equation (C-16) is the mathematical statement for conclusion number one given at the start of this proof.

If all of the eigenvalues are greater than one, Equations (C-15) and (C-14) can be combined to yield

$$[X_{i+1} - X_{EXT}]^T [X_{i+1} - X_{EXT}] = W^T \lambda W > W^T W$$

$$= [X_i - X_{EXT}]^T [X_i - X_{EXT}]. \qquad (C-17)$$

Equation (C-17) is the mathematical statement for conclusion number two given at the start of this proof.

APPENDIX D

SENSITIVITY EQUATIONS FOR DYNAMICAL SYSTEMS

It is assumed that a mathematical model for the system has been formulated and written in state variable notation as

$$\dot{X} = F\ (X,P,t). \tag{D-1}$$

In the above equation, X is an N dimensional state vector, P is an M dimensional parameter vector, and t is the independent variable, time.

The initial condition vector is partitioned as

$$X(t_0) = \begin{bmatrix} \psi \\ X_0 \end{bmatrix} \begin{matrix} \text{- free} \\ \\ \text{- specified.} \end{matrix} \tag{D-2}$$

In the above equation, ψ is a K dimensional vector of free initial conditions, and X_0 is an N-K dimensional vector of known initial conditions.

Equation (D-1) has been written as a total differential equation, but in actuality it should be a partial differential equation because the dependent variables, X, are functions of K + M + 1 independent variables,

$$X = X(P_1, \ldots, P_M, \psi_1, \ldots, \psi_K, t).$$

Hence, Equation (D-1) becomes

$$\frac{\partial X}{\partial t} = F(X, P, t). \tag{D-3}$$

If the partial derivative of Equation (D-3) is taken with respect to P_i and the order of the partial derivatives reversed, the following is obtained

$$\frac{\partial X_{P_i}}{\partial t} = F_X X_{P_i} + F_{P_i} \qquad i = 1, 2, \ldots, M. \tag{D-4}$$

The notation used in Equation (D-4) is

$$X_{P_i} = \frac{\partial X}{\partial P_i} \tag{D-5}$$

$$F_{P_i} = \frac{\partial F}{\partial P_i} \tag{D-6}$$

$$F_X = \begin{bmatrix} \frac{\partial F_1}{\partial X_1} & \cdots & \frac{\partial F_1}{\partial X_N} \\ \vdots & & \vdots \\ \frac{\partial F_N}{\partial X_1} & \cdots & \frac{\partial F_N}{\partial X_N} \end{bmatrix}. \tag{D-7}$$

If the partial derivative of Equation (D-3) is taken with respect to ψ_i, and the order of the partial derivatives reversed, the following is obtained:

$$\frac{\partial X_{\psi_i}}{\partial t} = F_X X_{\psi_i} \qquad i = 1, 2, \ldots, K. \tag{D-8}$$

The additional notation used in Equation (D-8) is

$$X_{\psi_i} = \frac{\partial X}{\partial \psi_i}. \qquad \text{(D-9)}$$

At time t_0 each of the initial conditions for the state variables is independent of the parameter guess. This yields,

$$X_{P_i} = 0; \quad i = 1, 2, \ldots, M. \qquad \text{(D-10)}$$

Since each of the free initial conditions is independent of the other initial conditions, one can conclude that

$$X_{\psi_i} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \leftarrow \text{row } i, \qquad i = 1, 2, \ldots, K. \qquad \text{(D-11)}$$

Equations (D-10) and (D-11) provide the initial conditions necessary for integrating Equations (D-4) and (D-8) forward in time from t_0 to t_f if a guess is made for the free portion of the initial condition given by Equation (D-2).

Equations (D-12) and (D-13) are generated by taking the partial derivative of Equations (D-4) and (D-8) with respect to P_j, and then reversing the order of the partial derivatives. Equations (D-14) and (D-15) are formed in a similar manner by taking the partial derivative with respect to ψ_j.

$$\frac{\partial X_{P_j P_i}}{\partial t} = F_X X_{P_j P_i} + \sum_{L=1}^{N} \left(\frac{\partial F_X}{\partial X_L} \frac{\partial X_L}{\partial P_j} + \frac{\partial F_X}{\partial P_j} \right) X_{P_i} +$$

$$+ F_{P_j P_i} + \sum_{L=1}^{N} \left(\frac{\partial F_{P_i}}{\partial X_L} \frac{\partial X_L}{\partial P_j} \right) \tag{D-12}$$

$$\frac{\partial X_{P_j \psi_i}}{\partial t} = F_X X_{P_j \psi_i} + \sum_{L=1}^{N} \left(\frac{\partial F_X}{\partial X_L} \frac{\partial X_L}{\partial P_j} + \frac{\partial F_X}{\partial P_j} \right) X_{\psi_i} \tag{D-13}$$

$$\frac{\partial X_{\psi_j P_i}}{\partial t} = F_X X_{\psi_j P_i} + \sum_{L=1}^{N} \left(\frac{\partial F_X}{\partial X_L} \frac{\partial X_L}{\partial \psi_j} \right) X_{P_i}$$

$$+ \sum_{L=1}^{N} \left(\frac{\partial F_{P_i}}{\partial X_L} \frac{\partial X_L}{\partial \psi_j} \right) \tag{D-14}$$

$$\frac{\partial X_{\psi_j \psi_i}}{\partial t} = F_X X_{\psi_j \psi_i} + \sum_{L=1}^{N} \left(\frac{\partial F_X}{\partial X_L} \frac{\partial X_L}{\partial \psi_j} \right) X_{\psi_i} . \tag{D-15}$$

The additional notation used in Equations (D-12) through (D-15) is

$$X_{P_j P_i} = \frac{\partial X_{P_i}}{\partial P_j} \qquad X_{P_j \psi_i} = \frac{\partial X_{\psi_i}}{\partial P_j}$$

$$X_{\psi_j P_i} = \frac{\partial X_{P_i}}{\partial \psi_j} \qquad X_{\psi_j \psi_i} = \frac{\partial X_{\psi_i}}{\partial \psi_j}$$

$$F_{P_j P_i} = \frac{\partial F_{P_i}}{\partial P_j} \quad \text{etc.}$$

The initial condition for Equations (D-12) through (D-15) is the zero vector since $X_{\psi_i}(t_0)$ and $X_{P_i}(t_0)$ remain constant regardless of how much the initial condition

vector and parameter vector change.

Equations (D-4), (D-8), and (D-12) through (D-15) provide the equations which can be integrated from time t_0 to t_f to provide values for the sensitivities of the state vector with respect to the free parameters.

APPENDIX E

COMPUTATIONAL ALGORITHM FOR SOLUTION OF TWO-POINT BOUNDARY VALUE PROBLEMS

In this appendix, a brief discussion of an algorithm for solving two-point boundary value problems by the method of seeking principal planes (TPSEEK) is presented. A listing of the algorithm is presented at the end of this appendix.

The TPSEEK algorithm is based upon the theoretical results which were presented in Chapters II and III.

The main program and subroutines TPSEEK, INPUT, SCAFCT, FCT, OUTP, and DRKF are listed at the end of this appendix. The subroutines FORTIN, PARDIF, DVAL, and ISTORE are also needed; and a description and listing of these subroutines has already been provided in Appendix A. The subroutine EIGEN is also needed, and it has been described and listed in Appendix B.

A brief description of each of the subroutines which are listed at the end of this appendix will now be provided.

Subroutine TPSEEK calculates the distance to each of the principal planes. It is basically the equivalent of the SEEK subprogram given in Appendix B for solving algebraic

minimization problems.

Subroutine INPUT is used for reading the input data from the data cards.

Subroutine SCAFCT evaluates the scalar function and the partial derivatives of the scalar function.

Subroutine DRKF is a fixed step size Runga Kutta numerical integration program.

Subroutine FCT provides the equations for the derivatives of the dynamical equations.

Subroutine OUTP prints the response of the dynamical equations.

Users Guide for the TPSEEK Algorithm

The user need only supply the appropriate data cards in order to use the TPSEEK algorithm.

The first data card has FORMAT (5I5) and contains the order of the dynamical system in columns 1 to 5, the number of free initial conditions in columns 6 to 10, the number of time steps between the print out of the dynamical response in columns 11 to 15, the number of steps to be used in the numerical integration in columns 16 to 20, and a one in column 25.

The second data card has FORMAT (3D15.7) and contains the initial time in columns 1 to 15, the final time in columns 16 to 30, and a zero in columns 31 to 45.

The state model must be set up so that the initial conditions of the first $N-N_{FREE}$ (N is the order of the

system, and N_{FREE} is the number of unknown initial conditions) state variables are known.

The third data card has FORMAT (5D15.7) and contains the initial conditions for the state vector. A guess is used for the unknown initial conditions.

The next N data cards contain the equations for the right-hand of the state model. The state variables must be denoted by X(1) through X(N). If the equations involve the independent variable, time; it must be denoted by X(N+1). The equations must be written in columns 7 through 72. If it is necessary to continue an equation on more than one data card, a C should be placed in column 6 of every data card except the last data card associated with that equation. Column 6 of the last data card associated with a continued equation must be left blank.

The last data card contains the equation for the performance index, and the notation for writing this equation is presented in the previous paragraph.

The dynamical equations for a two-point boundary value problem presented by Birta and Trushel in IEEE Transactions on Automatic Control, Vol. AC-12, are presented in Figure 39. The convergence characteristics of the method of seeking principal planes are also presented in this figure.

The data cards which are necessary for this example are presented in Figure 40.

Dynamical Equations

$$\dot{X}_1 = (1.0 - X^2_2) X_1 - X_2 - X_3/2.0$$

$$\dot{X}_2 = X_1$$

$$\dot{X}_3 = -2.0 X_1 - X_3 + X_3 X^2_2 - X_4$$

$$\dot{X}_4 = -2.0 X_2 + 2.0 X_1 X_2 X_3 + X_3$$

Scalar Performance Index

$$Y = X^2_3 + X^2_4$$

Initial Condition Guess

$$X_1 = 0.0 \quad X_2 = .10 \quad X_3 = 0.0 \quad X_4 = 0.0$$

Converged Initial Condition

$$X_1 = 0.0 \quad X_2 = .10 \quad X_3 = .081176 \quad X_4 = .66941$$

Initial Time = 0.0
Final Time = 4.0
Number of Steps = 40

Converged After:

10 Evaluations of the performance index
2 Evaluations of the gradient vector
7 Evaluations of the matrix of second partial derivatives

Figure 39. Results for the Birta and Trushel Example

```
      COLUMN     0 1     1     2 2     3          4          6
      NUMBER     5 0     5     0 5     0          5          0

1st. Data Card  4 2     2    40 1

2nd. Data Card    0.0D00          4.0D00      0.0D00

3rd. Data Card    0.0D00          .10D00      0.0D00      0.0D00

4th. Data Card (1.0-X(2) **2)*X(1)-X(2)-X(3)/2.0

5th. Data Card X(1)

6th. Data Card -2.0*X(1)-X(3)+X(3)*X(2)**2-X(4)

7th. Data Card -2.0*X(2)+2.0*X(1)*X(2)*X(3)+X(3)

8th Data Card  X(3)**2+X(4)**2
```

Figure 40. Data Cards Required for the Birta and Trushel Example in Chapter III

```
00000000011111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      CALL TPSEEK                                                          001
      CALL EXIT                                                            002
      END                                                                  003
      SUBROUTINE TPSEEK                                                    004
C     PURPOSE                                                              005
C       DETERMINE THE BEST VALUES FOR THE PARAMETERS AND THE FREE          006
C       INITIAL CONDITIONS BY THE METHOD OF SEEKING AXES PLANES.           007
C                                                                          008
C     THIS PROGRAM WAS WRITTEN AND PROGRAMED BY DENNIS UNRUH AT THE        009
C     DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING, OKLAHOMA STATE010
C     UNIVERSITY, STILLWATER, OKLAHOMA.  JULY 1969                       011
C                                                                        012
C       DELFPR -  THE PREDICTED DIFFERENCE BETWEEN THE VALUE OF F AT     013 0023
C        THE AXIS PLANE AND F AT THE BASE POINT.                         014 0024
C       DEN -  SEE NUM/DEN.                                              015 0043
C       DIRECT -  IF THIS NUMBER IS POSITIVE, THAN THE PREDICTED         016 0038
C        GRADIENT DIRECTION AND ACTUAL GRADIENT DIRECTION ARE WITHIN     017 0039
C        90 DEGREES OF EACH OTHER.                                       018 0040
C       EIGVAL -  EIGENVALUE VECTOR OF GX.                               019 0037
C       F -  FUNCTION BEING MINIMIZED.  F=F(X)                           020 0015
C       FBASE -  VALUE OF F AT BASE POINT                                021 0027
C       FBEST -  VALUE OF F AT BEST POINT.                               022 0028
C     FESMIN -  ESTIMATED MINIMUM FOR THE SCALAR FUNCTION.               023
C       G -  GRADIENT VECTOR OF FUNCTION F.  FOR EXAMPLE G(2)=PARTIAL F/024 0016
C        PARTIAL X(2).                                                   025 0017
C     GMAG  -  MAGNITUDE OF GRADIENT AFTER NPOS STEPS TO AXES PLANES.    026
C       GPRED -  PREDICTED VALUE OF GRADIENT AT AXIS PLAIN.              027 0027
C       GX -  MATRIX OF SECOND PARTIALS OF F.  I.E. GX(2,3)=PARTIAL G(2) 028 0018
C        /PARTIAL X(3).                                                  029 0019
C       GXS -  VECTOR WHICH CONTAINS GX IN SCIENTIFIC SUBROUTINE PACKAGE 030 0035
C        SYMMETRIC STORAGE MODE.                                         031 0036
C      N  -  THE NUMBER OF INDEPENDENT VARIABLES.                        032
C      NF  -  NUMBER OF TIMES F HAS BEEN CALCULATED.                     033
C      NG  -  NUMBER OF TIMES THE GRADIENT VECTOR HAS BEEN CALCULATED.   034
C      NGX  -  NUMBER OF TIMES MATRIX GX HAS BEEN CALCULATED.            035
C       NPOS -  NUMBER OF POSITIVE EIGENVALUES.                          036 0029
C       NUM/DEN -  THIS IS THE MAGNITUDE USED ON STEP NPOS+1 PROVIDED    037 0032
C        NPOS SUCCESSFUL STEPS TO THE AXIS PLAINS HAVE BEEN MADE.  THIS  038 0033
C        IS USED ONLY IF NNEG IS NOT ZERO.                               039 0034
C       NXMAX -  MAXIMUM NUMBER OF INDEPENDENT VARIABLES.                040 0044
C       PERRAT -  RATIO BETWEEN THE PREDICTED FUNCTION CHANGE AND THE    041 0025
C        ACTUAL FUNCTION CHANGE.                                         042 0026
C       R -  DUMMY VARIABLE NEEDED FOR CALL STATEMENT OF EIGEN.          043 0045
C       WORK -  WORKING VARIABLE.                                        044 0046
C       X -  INDEPENDENT VARIABLE VECTOR.                                045 0012
C     XBAMIN -  BASE POINT FROM WHICH A MINIMUM IS BEING SOUGHT BY A     046
C       LINEAR SEARCH IN THE PREDICTED NEGATIVE GRADIENT DIRECTION.      047
C       XBASE -  THE VALUE OF THE INDEPENDENT VARIABLE VECTOR AT THE     048 0013
C        BASE POINT.                                                     049 0014
C       XBEST -  THE VALUE OF THE INDEPENDENT VARIABLE AT THE POINT      050 0020
C        WHERE THE CALCULATED VALUE OF F WAS SMALLEST.                   051 0021
C       XNPOS -  THE VALUE OF THE INDEPENDENT VARIABLE AFTER NPOS        052 0041
C        SUCCESSFUL STEPS TO THE AXIS PLAINS.                            053 0042
      IMPLICIT INTEGER*2 (I-N),REAL*8 (A-H,O-Z)                          054
      INTEGER*4 MMM                                                      055
```

```
0000000001111111111222222222233333333334444444444555555555566666666667777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      REAL*8 NUM,DEN,MAG                                                    056
      NXMAX=10                                                              057 0047
C     DIMENSION X(NXMAX),G(NXMAX),XBEST(NXMAX),                             058
C     XBASE(NXMAX),GXS((NXMAX*(NXMAX+1))/2),R(1),EIGVAL(NXMAX),             059
C     GPRED(NXMAX),XBAMIN(NXMAX)                                            060
C     ,GXSS(NSMAX*(NXMAX+1))/2)                                             061
      DIMENSION X(10),G(10),XBEST(10),XBASE(10),GXS(55),                    062
     1R(1),EIGVAL(10),GPRED(10),XBAMIN(50)                                  063
     2,GXSS(55)                                                             064
      COMMON /GENER/ NSYS,NFREE,NFIXED,NSYSP1                               065
  100 FORMAT (8D15.7)                                                       066 0052
  101 FORMAT (' THE EIGENVALUE VECTOR IS')                                  067 0053
  102 FORMAT (6H NPOS=,I5)                                                  068
  103 FORMAT (4HONF=,I5,4H NG=,I5,5H NGX=,I5)                               069 0055
  104 FORMAT (' INDEPENDENT VECTOR X IS')                                   070
  105 FORMAT (3H F=,D15.7)                                                  071 0057
  106 FORMAT (' THE GRADIENT VECTOR IS')                                    072 0058
  107 FORMAT (' GXS(I),I=1,IJMAX')                                          073
  108 FORMAT (8H PERRAT=,D15.7)                                             074 0060
  109 FORMAT (' THE SOLUTION HAS CONVERGED')                                075 0061
  110 FORMAT (8HODIRECT=,D15.7)                                             076 0062
  111 FORMAT (7H FBEST=,D15.7)                                              077 0063
  112 FORMAT (' THE VALUE OF X TO AXIS PLAIN',I5,'   IS')                   078 0064
  113 FORMAT (' THE VALUE OF X AFTER HALVING IS')                           079 0065
  114 FORMAT (' THE VALUE OF X AFTER DOUBLING IS')                          080 0066
  115 FORMAT (I5,D15.7)                                                     081
  116 FORMAT (' THE FUNCTION VALUE IS ALREADY LESS THAN YOUR PREDICTED      082
     1MINIMUM')                                                             083
  117 FORMAT (3H N=,I5,8H FESMIN=,D15.7)                                    084
  118 FORMAT (' GRADIENT EVALUATED AT XBAMIN')                              085
      CALL INPUT(X,FESMIN)                                                  086
      N=NFREE                                                               087
      IJMAX=((N+1)*N)/2                                                     088
      NF=0                                                                  089 0067
      NG=0                                                                  090 0068
      NGX=0                                                                 091 0069
    3 I=3                                                                   092
      CALL SCAFCT(X,F,G,GXS,I)                                              093
      NGX=NGX+1                                                             094
      WRITE (6,103) NF,NG,NGX                                               095 0085
      WRITE (6,104)                                                         096 0072
      WRITE (6,100) (X(I),I=1,N)                                            097 0073
      WRITE (6,105) F                                                       098 0074
      DO 1 I=1,N                                                            099 0075
    1 XBEST(I)=X(I)                                                         100 0076
      FBEST=F                                                               101 0077
      WRITE (6,106)                                                         102 0082
      WRITE (6,100) (G(I),I=1,N)                                            103 0083
      WRITE (6,107)                                                         104 0088
      WRITE (6,100) (GXS(I),I=1,IJMAX)                                      105
      DO 5 I=1,N                                                            106
    5 XBASE(I)=X(I)                                                         107
      FBASE=F                                                               108 0092
C                                                                           109 0093
C     DETERMINE EIGENVALUES OF GX AND THE NUMBER NPOS.                      110
```

```
00000000011111111112222222222333333333344444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

C                                                                          111 0095
      DO 7 I=1,IJMAX                                                       112
    7 GXSS(I)=GXS(I)                                                       113
      MMM=N                                                                114
      CALL EIGEN(GXSS,R,MMM,1)                                             115
      DO 6 I=1,N                                                           116 0101
      II=(I*(I+1))/2                                                       117 0102
    6 EIGVAL(I)=GXSS(II)                                                   118
      NPOS=N                                                               119 0105
      DO 8 I=1,N                                                           120 0106
      IF (1.0D05*EIGVAL(I).GT.EIGVAL(1)) GO TO 8                           121
      NPOS=NPOS-1                                                          122 0108
    8 CONTINUE                                                             123 0115
      IF (EIGVAL(1).LE.0.0D0) NPOS=0                                       124
      WRITE (6,101)                                                        125 0117
      WRITE (6,100) (EIGVAL(I),I=1,N)                                      126 0118
      WRITE (6,102) NPOS                                                   127
C                                                                          128
C     IF NPOS IS EQUAL TO ZERO , A LINEAR SEARCH IS PERFORMED FOR A        129
C     MINIMUM IN THE NEGATIVE GRADIENT DIRECTION.                          130
C                                                                          131
      IF (NPOS.NE.0) GO TO 52                                              132
      DO 53 I=1,N                                                          133
   53 XBAMIN(I)=X(I)                                                       134
      NUM=FBEST-FESMIN                                                     135
      IF (NUM.LT.0.0D0) GO TO 30                                           136
      DEN=0.0D0                                                            137
      DO 50 I=1,N                                                          138
   50 DEN=DEN+G(I)*G(I)                                                    139
      DO 51 I=1,N                                                          140
   51 X(I)=X(I)-(NUM/DEN)*G(I)                                             141
      GO TO 23                                                             142
C                                                                          143 0120
C     DETERMINE VALUES FOR X TO THE SUCCESSIVE AXIS PLANES.                144 0121
C                                                                          145 0122
   52 DO 14 K=1,NPOS                                                       146
      DO 9 I=1,N                                                           147 0125
      GPRED(I)=G(I)                                                        148 0126
      DO 9 J=1,N                                                           149 0127
      IJ=((J-1)*J)/2+I                                                     150
      IF (J.LT.I) IJ=((I-1)*I)/2+J                                         151
    9 GPRED(I)=GPRED(I)+GXS(IJ)*(X(J)-XBASE(J))                            152
      DO 10 I=1,N                                                          153 0129
   10 X(I)=X(I)-GPRED(I)/EIGVAL(K)                                         154 0130
      I=1                                                                  155
      CALL SCAFCT(X,F,G,GXS,I)                                             156
      NF=NF+1                                                              157
      WRITE (6,103) NF,NG,NGX                                              158 0132
      WRITE (6,112) K                                                      159 0133
      WRITE (6,100) (X(I),I=1,N)                                           160 0134
      WRITE (6,105) F                                                      161 0135
      IF (F.GT.FBEST.AND.K.EQ.1) GO TO 41                                  162
      IF (F.GT.FBEST) GO TO 60                                             163
C                                                                          164 0140
C     CALCULATE PERRAT                                                     165 0141
```

```
00000000011111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

C                                                                          166 0142
      DELFPR=0.0                                                           167 0143
      DO 13 I=1,N                                                          168 0144
      WORK=0.0                                                             169 0145
      DO 12 J=1,N                                                          170 0146
      IJ=((J-1)*J)/2+I                                                     171
      IF (J.LT.I) IJ=((I-1)*I)/2+J                                         172
   12 WORK=WORK+GXS(IJ)*(X(J)-XBASE(J))                                    173
   13 DELFPR=DELFPR+(G(I)+.50*WORK)*(X(I)-XBASE(I))                        174 0148
      PERRAT =(F-FBASE)/DELFPR                                             175 0149
      WRITE (6,108) PERRAT                                                 176 0150
      IF (DABS(PERRAT-1.0D0).GT..1D0) GO TO 3                              177
      DO 11 I=1,N                                                          178 0137
   11 XBEST(I)=X(I)                                                        179 0138
      FBEST=F                                                              180 0139
   14 CONTINUE                                                             181 0152
      DO 15 I=1,N                                                          182 0154
      GPRED(I)=G(I)                                                        183
      DO 15 J=1,N                                                          184 0156
      IJ=((J-1)*J)/2+I                                                     185
      IF (J.LT.I) IJ=((I-1)*I)/2+J                                         186
   15 GPRED(I)=GPRED(I)+GXS(IJ)*(X(J)-XBASE(J))                            187
      I=2                                                                  188
      CALL SCAFCT(X,F,G,GXS,I)                                             189
      NG=NG+1                                                              190
      WRITE (6,103) NF,NG,NGX                                              191 0159
      WRITE (6,106)                                                        192 0162
      WRITE (6,100) (G(I),I=1,N)                                           193 0163
C                                                                          194
C     CHECK FOR CONVERGENCE.                                               195
C                                                                          196
      DO 45 I=1,N                                                          197
      IF (DABS(G(I)).GT.1.0D-8)  GO TO 46                                  198
   45 CONTINUE                                                             199
      GO TO 33                                                             200
   46 IF (NPOS.EQ.N) GO TO 3                                               201
      DIRECT=0.0                                                           202 0164
      DO 16 I=1,N                                                          203 0165
   16 DIRECT=DIRECT+G(I)*GPRED(I)                                          204 0166
      WRITE (6,110) DIRECT                                                 205 0167
      IF (DIRECT.LE.0.0) GO TO 3                                           206 0168
      NUM=0.0                                                              207 0169
      DEN=0.0                                                              208 0170
      DO 17 I=1,N                                                          209 0171
      NUM=NUM+(X(I)-XBASE(I))**2                                           210 0172
   17 DEN=DEN+(GPRED(I))**2                                                211 0173
      MAG=DSQRT(NUM/DEN)                                                   212
      DO 18 I=1,N                                                          213 0174
   18 X(I)=X(I)-MAG*GPRED(I)                                               214
C                                                                          215
C     PERFORM A LINEAR SEARCH FOR A MINIMUM IN THE PREDICTED NEGATIVE      216
C     GRADIENT DIRECTION.                                                  217
C                                                                          218
      I=1                                                                  219
      CALL SCAFCT(X,F,G,GXS,I)                                             220
```

```
00000000011111111112222222222333333333344444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      NF=NF+1                                                          221
      WRITE (6,103) NF,NG,NGX                                          222 0178
      WRITE (6,104)                                                    223 0179
      WRITE (6,100) (X(I),I=1,N)                                       224 0180
      WRITE (6,105) F                                                  225 0181
   41 DO 42 I=1,N                                                      226
   42 XBAMIN(I)=XBEST(I)                                               227
      IF (F.GT.FBEST) GO TO 23                                         228 0182
C                                                                      229
C     CONTINUE DOUBLING THE DISTANCE UNTIL F CEASES TO BECOME BETTER.  230
C                                                                      231
   19 DO 20 I=1,N                                                      232 0183
   20 XBEST(I)=X(I)                                                    233 0184
      FBEST=F                                                          234 0185
      DO 21 I=1,N                                                      235 0186
   21 X(I)=XBAMIN(I)+2.0*(X(I)-XBAMIN(I))                              236
      I=1                                                              237
      CALL SCAFCT(X,F,G,GXS,I)                                         238
      NF=NF+1                                                          239
      WRITE (6,103) NF,NG,NGX                                          240 0189
      WRITE (6,114)                                                    241 0190
      WRITE (6,100) (X(I),I=1,N)                                       242 0191
      WRITE (6,105) F                                                  243 0192
      IF (F.LT.FBEST) GO TO 19                                         244 0193
      DO 22 I=1,N                                                      245 0194
   22 X(I)=XBEST(I)                                                    246 0195
      F=FBEST                                                          247 0196
      GO TO 3                                                          248
C                                                                      249
C     CONTINUE HALVING THE DISTANCE UNTIL F BECOMES BETTER.            250
C                                                                      251
   23 DO 24 I=1,N                                                      252 0198
   24 X(I)=XBAMIN(I)+.50*(X(I)-XBAMIN(I))                              253
      I=1                                                              254
      CALL SCAFCT(X,F,G,GXS,I)                                         255
      NF=NF+1                                                          256
      WRITE (6,103) NF,NG,NGX                                          257 0201
      WRITE (6,113)                                                    258 0202
      WRITE (6,100) (X(I),I=1,N)                                       259 0203
      WRITE (6,105) F                                                  260 0204
      IF (F.GT.FBEST) GO TO 23                                         261 0205
      DO 25 I=1,N                                                      262 0206
   25 XBEST(I)=X(I)                                                    263 0207
      FBEST=F                                                          264 0208
      GO TO 3                                                          265
   60 DO 61 I=1,N                                                      266
   61 XBAMIN(I)=XBEST(I)                                               267
      I=2                                                              268
      CALL SCAFCT(XBAMIN,F,G,GXS,I)                                    269
      NG=NG+1                                                          270
      WRITE (6,103) NF,NG,NGX                                          271
      WRITE (6,118)                                                    272
      WRITE (6,100) (G(I),I=1,N)                                       273
      DIRECT=0.0                                                       274
      DO 62 I=1,N                                                      275
```

```
00000000011111111112222222222333333333344444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

   62 DIRECT=DIRECT*G(I)*GPRED(I)                                          276
      WRITE (6,110) DIRECT                                                 277
      IF (DIRECT.GT.0.0D0) GO TO 23                                        278
      DO 63 I=1,N                                                          279
   63 X(I)=XBAMIN(I)                                                       280
      GO TO 3                                                              281
   30 WRITE (6,116)                                                        282
      CALL EXIT                                                            283
   33 WRITE (6,109)                                                        284
      CALL EXIT                                                            285 0241
      RETURN                                                               286 0242
      END                                                                  287 0243
      SUBROUTINE INPUT(X,FESMIN)                                           288
      IMPLICIT INTEGER*2(I-N),REAL*8(A-H,O-Z)                              289
C     PURPOSE                                                              290
C       READ DATA CARDS.                                                   291
C         DATA CARD ONE CONTAINS NSYS,NFREE,NPAR,NWRITE,NSTEP,NOPT         292
C         DATA CARD TWO CONTAINS PRMT(1),PRMT(2),FESMIN                    293
C         DATA CARD THREE CONTAINS THE INITIAL CONDITION GUESS FOR         294
C          COMPONENTS 1 THROUGH NSYS OF Y.                                 295
C         DATA CARD FOUR CONTAINS THE INITIAL GUESS FOR THE PARAMETERS.    296
C          IF NPAR=0 LEAVE THIS DATA CARD OUT.                             297
C         THE SYSTEM EQUATIONS ARE ENTERED ON CARDS 5 THROUGH NSYS+4.      298
C         THE EQUATION FOR THE INTEGRAN OF THE SCALAR PERFORMANCE INDEX    299
C          IS ENTERED ON THE LAST DATA CARD.                               300
C     IF NOPT=1 EQUATIONS ARE ENTERED ON DATA CARDS.  IF NOPT=2 EQUATION301
C     ARE ENTERED ON DATA CARDS AND THEIR SYMBOLIC FORM IS PUNCHED.        302
C     IF NOPT=3 EQUATIONS ARE ENTERED IN SYMBOLIC FORM.                    303
      DIMENSION X(10)                                                      304
      COMMON /GENER/ NSYS,NFREE,NFIXED,NSYSP1                              305
      COMMON /INFC/ IF(10),IFX(10,10),IFXXS(10,55)                         306
      COMMON /INOU/ NWRITE                                                 307
      COMMON /SCIN/ YI(10),PRMT(5),IG,IGX(10),IGXXS(55)                    308
  100 FORMAT (8D15.7)                                                      309
  101 FORMAT (5D15.7)                                                      310
  102 FORMAT (10I5)                                                        311
  103 FORMAT (' NSYS,NFREE,NWRITE,NSTEP,NOPT')                             312
  104 FORMAT ('   PRMT(1)    PRMT(2)      FESMIN')                         313
  106 FORMAT (5(I5,F10.5))                                                 314
  108 FORMAT (' INITIAL CONDITION USED')                                   315
  110 FORMAT (' IF(',I4,')=')                                              316
  111 FORMAT (' IFX(',2I4,')=')                                            317
  112 FORMAT (' IFXXS(',2I4,')=')                                          318
  113 FORMAT (' IG=')                                                      319
  114 FORMAT (' IGX(',I4,')=')                                             320
  115 FORMAT (' IGXXS(',I4,')=')                                           321
      READ (5,102)  NSYS,NFREE,NWRITE,NSTEP,NOPT                           322
      WRITE (6,103)                                                        323
      WRITE (6,102) NSYS,NFREE,NWRITE,NSTEP,NOPT                           324
      READ(5,101) PRMT(1),PRMT(2),FESMIN                                   325
      WRITE (6,104)                                                        326
      WRITE (6,100) PRMT(1),PRMT(2),FESMIN                                 327
      READ(5,101) (YI(I),I=1,NSYS)                                         328
      WRITE (6,108)                                                        329
      WRITE (6,100) (YI(I),I=1,NSYS)                                       330
```

```
00000000011111111112222222222333333333344444444445555555555666666666677777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      NFIXED=NSYS-NFREE                                                    331
      NSYSP1=NSYS+1                                                        332
      RNSTEP=NSTEP                                                         333
      PRMT(3)=(PRMT(2)-PRMT(1))/RNSTEP                                     334
      DO 3 I=1,NFREE                                                       335
      II=NFIXED+I                                                          336
    3 X(I)=YI(II)                                                          337
      IF (NOPT.EQ.3) GO TO 20                                              338
      DO 7 I=1,NSYS                                                        339
      WRITE (6,110) I                                                      340
      CALL FORTIN(IF(I))                                                   341
      DO 7 K=1,NSYS                                                        342
      WRITE (6,111) I,K                                                    343
      CALL PARDIF(IF(I),IFX(I,K),K)                                        344
      DO 7 J=1,K                                                           345
      JK=((K-1)*K)/2+J                                                     346
      WRITE (6,112) I,JK                                                   347
    7 CALL PARDIF(IFX(I,K),IFXXS(I,JK),J)                                  348
      WRITE (6,113)                                                        349
      CALL FORTIN (IG)                                                     350
      DO 8 J=1,NSYS                                                        351
      WRITE (6,114) J                                                      352
      CALL PARDIF(IG,IGX(J),J)                                             353
      DO 8 I=1,J                                                           354
      IJ=((J-1)*J)/2+I                                                     355
      WRITE (6,115) IJ                                                     356
    8 CALL PARDIF(IGX(J),IGXXS(IJ),I)                                      357
      IF (NOPT.EQ.1) RETURN                                                358
      I=3                                                                  359
      CALL ISTORE (NAME,I)                                                 360
      RETURN                                                               361
   20 I=4                                                                  362
      CALL ISTORE (NAME,I)                                                 363
      RETURN                                                               364
      END                                                                  365
      SUBROUTINE SCAFCT (X,F,G,GXS,ILOG)                                   366
C     PURPOSE                                                              367
C       SETS UP THE INITIAL CONDITIONS FOR THE DYNAMICAL EQUATIONS.        368
C       EVALUATES THE SCALAR PERFORMANCE FUNCTION,ITS GRADIENT, AND        369
C       ITS MATRIX OF SECOND PARTIAL DERIVITIVES.                          370
      IMPLICIT INTEGER*2 (I-N),REAL*8 (A-H,O-Z)                            371
      INTEGER*4 NTOTAL,IHLF                                                372
      DIMENSION X(10),G(10),GXS(55),Y(210),DERY(210),AUX(210)              373
     1,GX(10),GXXS(55),B(10),Z(50)                                         374
      COMMON /GENER/ NSYS,NFREE,NFIXED,NSYSP1                              375
      COMMON /SCIN/ YI(10),PRMT(5),IG,IGX(10),IGXXS(55)                    376
      COMMON /SCFCOU/ ILOGIC                                               377
      EXTERNAL FCT,OUTP                                                    378
      DATA NDIFTO /0/                                                      379
  100 FORMAT (' NDIFTO=',I5)                                               380
      ILOGIC=ILOG                                                          381
      IF (NFIXED.EQ.0) GO TO 4                                             382
      DO 1 I=1,NFIXED                                                      383
    1 Y(I)=YI(I)                                                           384
    4 DO 2 I=1,NFREE                                                       385
```

```
00000000011111111112222222222333333333344444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      II=I+NFIXED                                                          386
    2 Y(II)=X(I)                                                           387
      GO TO (10,20,30), ILOGIC                                             388
C                                                                          389
C     DETERMINE THE VALUE FOR THE SCALAR PERFORMANCE FUNCTION, F.          390
C                                                                          391
   10 NTOTAL=NSYS                                                          392
      NDIFTO=NDIFTO+NTOTAL                                                 393
      WRITE (6,100) NDIFTO                                                 394
      CALL DRKF(PRMT,Y,DERY,NTOTAL,IHLF,FCT,OUTP,AUX)                      395
      GO TO 40                                                             396
C                                                                          397
C     DETERMINE THE VALUE FOR THE GRADIENT OF THE SCALAR PERFORMANCE       398
C     FUNCTION, G.                                                         399
C                                                                          400
   20 NTOTAL=(NFREE+1)*NSYS                                                401
      DO 21 I=NSYSP1,NTOTAL                                                402
   21 Y(I)=0.0                                                             403
      DO 22 I=1,NFREE                                                      404
      II=NSYS*I+NFIXED+I                                                   405
   22 Y(II)=1.0                                                            406
      NDIFTO=NDIFTO+NTOTAL                                                 407
      WRITE (6,100) NDIFTO                                                 408
   23 CALL DRKF(PRMT,Y,DERY,NTOTAL,IHLF,FCT,OUTP,AUX)                      409
      GO TO 40                                                             410
C                                                                          411
C     DETERMINE THE VALUES FOR THE ELEMENTS OF THE MATRIX OF SECOND        412
C     PARTIAL DERIVITIVES OF THE SCALAR PERFORMANCE FUNCTION.              413
C                                                                          414
   30 NTOTAL=(1+NFREE+((NFREE+1)*NFREE)/2)*NSYS                            415
      DO 31 I=NSYSP1,NTOTAL                                                416
   31 Y(I)=0.0                                                             417
      DO 32 I=1,NFREE                                                      418
      II=NSYS*I+NFIXED+I                                                   419
   32 Y(II)=1.0                                                            420
      NDIFTO=NDIFTO+NTOTAL                                                 421
      WRITE (6,100) NDIFTO                                                 422
   33 CALL DRKF(PRMT,Y,DERY,NTOTAL,IHLF,FCT,OUTP,AUX)                      423
   40 DO 41 I=1,NSYS                                                       424
   41 Z(I)=Y(I)                                                            425
      IF (ILOGIC.EQ.2) GO TO 42                                            426
      F=DVAL(IG,Z)                                                         427
      IF (ILOGIC.EQ.1) RETURN                                              428
   42 DO 43 I=1,NSYS                                                       429
   43 GX(I)=DVAL(IGX(I),Z)                                                 430
      DO 44 I=1,NFREE                                                      431
      G(I)=0.0                                                             432
      DO 44 J=1,NSYS                                                       433
      JJ=I*NSYS+J                                                          434
   44 G(I)=G(I)+GX(J)*Y(JJ)                                                435
      IF (ILOGIC.EQ.2) RETURN                                              436
      DO 50 J=1,NSYS                                                       437
      DO 50 I=1,J                                                          438
      IJ=((J-1)*J)/2+I                                                     439
   50 GXXS(IJ)=DVAL(IGXXS(IJ),Z)                                           440
```

80/80 LIST E-13

```
0000000001111111111222222222233333333334444444444555555555566666666667777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      DO 52 J=1,NFREE                                                    441
      DO 51 LI=1,NSYS                                                    442
      B(LI)=0.0                                                          443
      DO 51 LJ=1,NSYS                                                    444
      LIJ=((LJ-1)*LJ)/2+LI                                               445
      IF (LJ.LT.LI) LIJ=((LI-1)*LI)/2+LJ                                 446
      LJJ=J*NSYS+LJ                                                      447
   51 B(LI)=B(LI)+GXXS(LIJ)*Y(LJJ)                                       448
      DO 52 I=1,J                                                        449
      IJ=((J-1)*J)/2+I                                                   450
      GXS(IJ)=0.0                                                        451
      DO 52 K=1,NSYS                                                     452
      KK=I*NSYS+K                                                        453
      KKK=(NFREE+IJ)*NSYS+K                                              454
   52 GXS(IJ)=GXS(IJ)+Y(KK)*B(K)+GX(K)*Y(KKK)                            455
      RETURN                                                             456
      END                                                                457
      SUBROUTINE FCT(T,Y,DERY)                                           458
C     PURPOSE                                                            459
C       SUPPLYS THE EQUATIONS FOR THE DYNAMICAL SYSTEM.                  460
      IMPLICIT INTEGER*2(I-N),REAL*8 (A-H,O-Z)                           461
      DIMENSION Y(210),DERY(210),Z(50),FX(10,10),FXXS(10,55),            462
     1B(10,10)                                                           463
      COMMON /GENER/ NSYS,NFREE,NFIXED,NSYSP1                            464
      COMMON /INFC/ IF(10),IFX(10,10),IFXXS(10,55)                       465
      COMMON /SCFCOU/ ILOGIC                                             466
      DO 1 I=1,NSYS                                                      467
    1 Z(I)=Y(I)                                                          468
      Z(NSYSP1)=T                                                        469
C                                                                        470
C     DETERMINE THE VALUES FOR THE DERIVITIVE OF THE BASIC SYSTEM.       471
C                                                                        472
   10 DO 11 I=1,NSYS                                                     473
   11 DERY(I)=DVAL(IF(I),Z)                                              474
      IF (ILOGIC.EQ.1) RETURN                                            475
C                                                                        476
C     DETERMINE THE VALUES FOR THE DERIVITIVES OF THE FIRST ORDER        477
C     INFLUENCE COEFFICIENTS.                                            478
   20 DO 21 I=1,NSYS                                                     479
      DO 21 J=1,NSYS                                                     480
   21 FX(I,J)=DVAL(IFX(I,J),Z)                                           481
      DO 22 I=1,NSYS                                                     482
      DO 22 J=1,NFREE                                                    483
      IJ=J*NSYS+I                                                        484
      DERY(IJ)=0.0                                                       485
      DO 22 K=1,NSYS                                                     486
      JK=J*NSYS+K                                                        487
   22 DERY(IJ)=DERY(IJ)+FX(I,K)*Y(JK)                                    488
      IF (ILOGIC.EQ.2) RETURN                                            489
C                                                                        490
C     DETERMINE THE DERIVITIVES FOR THE SECOND ORDER INFLUENCE           491
C     COEFFICIENTS.                                                      492
C                                                                        493
      DO 25 I=1,NSYS                                                     494
      DO 25 K=1,NSYS                                                     495
```

```
00000000011111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      DO 25 J=1,K                                                          496
      JK=((K-1)*K)/2+J                                                     497
   25 FXXS(I,JK)=DVAL(IFXXS(I,JK),Z)                                       498
      DO 32 J=1,NFREE                                                      499
      DO 30 LI=1,NSYS                                                      500
      DO 30 LJ=1,NSYS                                                      501
      B(LI,LJ)=0.0                                                         502
      DO 30 LK=1,NSYS                                                      503
      JJ=J*NSYS+LK                                                         504
      LJLK=((LK-1)*LK)/2+LJ                                                505
      IF (LK.LT.LJ) LJLK=((LJ-1)*LJ)/2+LK                                  506
   30 B(LI,LJ)=B(LI,LJ)+Y(JJ)*FXXS(LI,LJLK)                                507
      DO 32 I=1,J                                                          508
      DO 32 K=1,NSYS                                                       509
      IJ=((J-1)*J)/2+I                                                     510
      KK=(NFREE+IJ)*NSYS+K                                                 511
      DERY(KK)=0.0                                                         512
      DO 32 L=1,NSYS                                                       513
      LL=(NFREE+((J-1)*J)/2+1)*NSYS+L                                      514
      LLL=I*NSYS+L                                                         515
   32 DERY(KK)=DERY(KK)+FX(K,L)*Y(LL)+B(K,L)*Y(LLL)                        516
      RETURN                                                               517
      END                                                                  518
      SUBROUTINE OUTP(T,Y,DERY,IHLF,NTOTAL,PRMT)                           519
C     PURPOSE                                                              520
C       WRITE THE SYSTEM RESPONSE AFTER EVERY NWRITE STEPS OF NUMERICAL    521
C       INTEGRATION.  NUMERICALLY INTEGRATE THE INTEGRAL EQUATIONS BY      522
C     USING THE TRAPAZOID RULE.                                            523
      IMPLICIT INTEGER*2(I-N),REAL*8 (A-H,O-Z)                             524
      INTEGER*4 IHLF,NTOTAL                                                525
      DIMENSION Y(210),DERY(210),PRMT(5)                                   526
      COMMON /GENER/ NSYS,NFREE,NFIXED,NSYSP1                              527
      COMMON /INOU/ NWRITE                                                 528
      COMMON /SCFCOU/ ILOGIC                                               529
  100 FORMAT (1H0,D15.7)                                                   530
  101 FORMAT (8D15.7)                                                      531
      GO TO (10,20,30),ILOGIC                                              532
   10 NINT=1                                                               533
      GO TO 35                                                             534
   20 NINT=1+NFREE                                                         535
      GO TO 35                                                             536
   30 NINT=1+NFREE+((NFREE+1)*NFREE)/2                                     537
   35 IF (T.EQ.PRMT(1)) GO TO 41                                           538
      IF (T.GT.PRMT(2)-PRMT(3)/2.0) GO TO 41                               539
      IF (ILOGIC.GT.1) RETURN                                              540
      IF (NCOUNT.EQ.NWRITE) GO TO 41                                       541
      NCOUNT=NCOUNT+1                                                      542
      RETURN                                                               543
C                                                                          544
C     RESET NCOUNT AND WRITE OUT THE VALUES AT THIS TIME.                  545
C                                                                          546
   41 NCOUNT=1                                                             547
      WRITE (6,100) T                                                      548
      DO 42 I=1,NINT                                                       549
      JJ=(I-1)*NSYS+1                                                      550
```

80/80 LIST E-15

```
00000000011111111112222222223333333334444444445555555556666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      II=1*NSYS                                                          551
   42 WRITE (6,101) (Y(J),J=JJ,II)                                       552
      RETURN                                                             553
      END                                                                554
```

```
00000000011111111112222222223333333333444444444455555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      SUBROUTINE DRKF (PRMT,Y,DERY,NDIM,IHLF,FCT,OUTP,AUX)
C                                                                       DRKGS001
C     ..................................................................DRKGS002
C                                                                       DRKGS003
C       SUBROUTINE DRKF
C                                                                       DRKGS005
C       PURPOSE                                                         DRKGS006
C          TO SOLVE A SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL       DRKGS007
C          EQUATIONS WITH GIVEN INITIAL VALUES.                         DRKGS008
C                                                                       DRKGS009
C       USAGE                                                           DRKGS010
C            CALL DRKF (PRMT,T,DERY,NDIM,IHLF,FCT,OUTP,AUX)
C          PARAMETERS FCT AND OUTP REQUIRE AN EXTERNAL STATEMENT.       DRKGS012
C                                                                       DRKGS013
C       DESCRIPTION OF PARAMETERS                                       DRKGS014
C          PRMT   - DOUBLE PRECISION INPUT AND OUTPUT VECTOR WITH       DRKGS015
C                   DIMENSION GREATER THAN OR EQUAL TO 5, WHICH         DRKGS016
C                   SPECIFIES THE PARAMETERS OF THE INTERVAL AND OF     DRKGS017
C                   ACCURACY AND WHICH SERVES FOR COMMUNICATION BETWEENDRKGS018
C                   OUTPUT SUBROUTINE (FURNISHED BY THE USER) AND       DRKGS019
C                   SUBROUTINE DRKGS. EXCEPT PRMT(5) THE COMPONENTS     DRKGS020
C                   ARE NOT DESTROYED BY SUBROUTINE DRKGS AND THEY ARE  DRKGS021
C          PRMT(1)- LOWER BOUND OF THE INTERVAL (INPUT),                DRKGS022
C          PRMT(2)- UPPER BOUND OF THE INTERVAL (INPUT),                DRKGS023
C          PRMT(3)- INITIAL INCREMENT OF THE INDEPENDENT VARIABLE       DRKGS024
C                   (INPUT),                                            DRKGS025
C          PRMT(4)- DUMMY PARAMETER WHICH NEED NOT BE SPECIFIED SINCE
C                   IT IS NOT USED IN THIS SUBROUTINE.  IT IS LEFT
C                   OVER FROM DRKGS.  SEE REMARKS GIVEN BELOW
C          PRMT(5)- NO INPUT PARAMETER. SUBROUTINE DRKGS INITIALIZES    DRKGS032
C                   PRMT(5)=0. IF THE USER WANTS TO TERMINATE           DRKGS033
C                   SUBROUTINE DRKGS AT ANY OUTPUT POINT, HE HAS TO     DRKGS034
C                   CHANGE PRMT(5) TO NON-ZERO BY MEANS OF SUBROUTINE   DRKGS035
C                   OUTP. FURTHER COMPONENTS OF VECTOR PRMT ARE         DRKGS036
C                   FEASIBLE IF ITS DIMENSION IS DEFINED GREATER        DRKGS037
C                   THAN 5. HOWEVER SUBROUTINE DRKGS DOES NOT REQUIRE   DRKGS038
C                   AND CHANGE THEM. NEVERTHELESS THEY MAY BE USEFUL    DRKGS039
C                   FOR HANDING RESULT VALUES TO THE MAIN PROGRAM       DRKGS040
C                   (CALLING DRKF ) WHICH ARE OBTAINED BY SPECIAL
C                   MANIPULATIONS WITH OUTPUT DATA IN SUBROUTINE OUTP.   DRKGS042
C          Y      - DOUBLE PRECISION INPUT VECTOR OF INITIAL VALUES     DRKGS043
C                   (DESTROYED). LATERON Y IS THE RESULTING VECTOR OF   DRKGS044
C                   DEPENDENT VARIABLES COMPUTED AT INTERMEDIATE        DRKGS045
C                   POINTS X.                                           DRKGS046
C          DERY     - DERY IS THE VECTOR OF
C                   DERIVATIVES, WHICH BELONG TO FUNCTION VALUES Y AT   DRKGS050
C                   INTERMEDIATE POINTS X.                              DRKGS051
C          NDIM   - AN INPUT VALUE, WHICH SPECIFIES THE NUMBER OF       DRKGS052
C                   EQUATIONS IN THE SYSTEM.                            DRKGS053
C          IHLF   - DUMMY PARAMETER WHICH NEED NOT BE SPECIFIED SINCE
C                   IT IS NOT USED IN THIS SUBROUTINE.  IT IS LEFT
C                   OVER FROM DRKGS.  SEE REMARKS GIVEN BELOW
C          FCT    - THE NAME OF AN EXTERNAL SUBROUTINE USED. THIS       DRKGS061
C                   SUBROUTINE COMPUTES THE RIGHT HAND SIDES DERY OF    DRKGS062
C                   THE SYSTEM TO GIVEN VALUES X AND Y. ITS PARAMETER   DRKGS063
```

```
0000000001111111111222222222233333333334444444444555555555566666666667777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456789
C                     LIST MUST BE X,Y,DERY. SUBROUTINE FCT SHOULD        DRKGS064
C                     NOT DESTROY X AND Y.                               DRKGS065
C            OUTP   - THE NAME OF AN EXTERNAL OUTPUT SUBROUTINE USED.    DRKGS066
C                     ITS PARAMETER LIST MUST BE X,Y,DERY,IHLF,NDIM,PRMT.DRKGS067
C                     NONE OF THESE PARAMETERS (EXCEPT, IF NECESSARY,     DRKGS068
C                             PRMT(5),...) SHOULD BE CHANGED BY
C                     SUBROUTINE OUTP. IF PRMT(5) IS CHANGED TO NON-ZERO,DRKGS070
C                     SUBROUTINE DRKGS IS TERMINATED.                    DRKGS071
C             AUX   - DOUBLE PRECISION AUXILIARY VECTOR STORAGE WITH
C               NDIM ROWS.
C                                                                        DRKGS074
C        REMARKS                                                         DRKGS075
C           THE PROCEDURE TERMINATES AND RETURNS TO CALLING PROGRAM, IF  DRKGS076
C           (1) SUBROUTINE OUTP HAS CHANGED PRMT(5) TO NON-ZERO.
C           (2) INITIAL INCREMENT IS EQUAL TO 0 OR HAS WRONG SIGN        DRKGS080
C           (3) THE WHOLE INTEGRATION INTERVAL IS WORKED THROUGH,        DRKGS082
C
C           THIS PROGRAM IS A REVISION OF THE SSP PROGRAM DRKGS.  IT
C           USES A FIXED STEP SIZE FOR ALL STEPS EXCEPTING PERHAPS THE
C           LAST STEP WHICH WILL BE ADJUSTED TO HIT THE DESIRED
C           FINAL TIME EXACTLY.  THE CALL STATEMENTS HAVE ALL BEEN LEFT
C           WITH THE SAME PARAMETERS AS IN DRKGS.
C                                                                        DRKGS084
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED                   DRKGS085
C           THE EXTERNAL SUBROUTINES FCT(X,Y,DERY) AND                   DRKGS086
C           OUTP(X,Y,DERY,IHLF,NDIM,PRMT) MUST BE FURNISHED BY THE USER.DRKGS087
C                                                                        DRKGS088
C        METHOD                                                          DRKGS089
C           EVALUATION IS DONE BY MEANS OF FOURTH ORDER RUNGE-KUTTA      DRKGS090
C           FORMULAE IN THE MODIFICATION DUE TO GILL.
C           TO GET FULL FLEXIBILITY IN OUTPUT, AN OUTPUT SUBROUTINE      DRKGS099
C           MUST BE FURNISHED BY THE USER.                               DRKGS100
C           FOR REFERENCE, SEE                                           DRKGS101
C           RALSTON/WILF, MATHEMATICAL METHODS FOR DIGITAL COMPUTERS,    DRKGS102
C           WILEY, NEW YORK/LONDON, 1960, PP.110-120.                    DRKGS103
C                                                                        DRKGS104
C     ...................................................................DRKGS105
C                                                                        DRKGS106
C                                                                        DRKGS108
C                                                                        DRKGS109
      DIMENSION Y(1),DERY(1),AUX(1),A(4),B(4),C(4),PRMT(1)
      DOUBLE PRECISION PRMT,Y,DERY,AUX,A,B,C,X,XEND,H,AJ,BJ,CJ,R1,R2
      X=PRMT(1)                                                          DRKGS115
      XEND=PRMT(2)                                                       DRKGS116
      H=PRMT(3)                                                          DRKGS117
      PRMT(5)=0.D0                                                       DRKGS118
      CALL FCT(X,Y,DERY)                                                 DRKGS119
C                                                                        DRKGS120
C     ERROR TEST                                                         DRKGS121
      IF (H*(XEND-X)) 40,40,2
C                                                                        DRKGS123
C     PREPARATIONS FOR RUNGE-KUTTA METHOD                                DRKGS124
    2 A(1)=.5D0                                                          DRKGS125
      A(2)=.29289321881345248D0                                          DRKGS126
      A(3)=1.7071067811865475D0                                          DRKGS127
```

```
00000000011111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      A(4)=.1666666666666667D0                                          DRKGS128
      B(1)=2.D0                                                         DRKGS129
      B(2)=1.D0                                                         DRKGS130
      B(3)=1.D0                                                         DRKGS131
      B(4)=2.D0                                                         DRKGS132
      C(1)=.5D0                                                         DRKGS133
      C(2)=.2928932188134524D0                                          DRKGS134
      C(3)=1.7071067811865475D0                                         DRKGS135
      C(4)=.5D0                                                         DRKGS136
C                                                                       DRKGS137
C     PREPARATIONS OF FIRST RUNGE-KUTTA STEP                            DRKGS138
      DO 3 I=1,NDIM                                                     DRKGS139
    3 AUX(I)=0.D0
      IEND=0                                                            DRKGS148
C                                                                       DRKGS149
C                                                                       DRKGS150
C     START OF A RUNGE-KUTTA STEP                                       DRKGS151
    4 IF((X+H-XEND)*H)7,6,5                                             DRKGS152
    5 H=XEND-X                                                          DRKGS153
    6 IEND=1                                                            DRKGS154
C                                                                       DRKGS155
C     RECORDING OF INITIAL VALUES OF THIS STEP                          DRKGS156
    7 CALL OUTP(X,Y,DERY,IREC,NDIM,PRMT)                                DRKGS157
      IF (PRMT(5).NE.0.D0) GO TO 40
C                                                                       DRKGS161
C                                                                       DRKGS162
C     START OF INNERMOST RUNGE-KUTTA LOOP                               DRKGS163
      J=1                                                               DRKGS164
   10 AJ=A(J)                                                           DRKGS165
      BJ=B(J)                                                           DRKGS166
      CJ=C(J)                                                           DRKGS167
      DO 11 I=1,NDIM                                                    DRKGS168
      R1=H*DERY(I)                                                      DRKGS169
      R2=AJ*(R1-BJ*AUX(I))
      Y(I)=Y(I)+R2                                                      DRKGS171
      R2=R2+R2+R2                                                       DRKGS172
   11 AUX(I)=AUX(I)+R2-CJ*R1
      IF(J-4)12,15,15                                                   DRKGS174
   12 J=J+1                                                             DRKGS175
      IF(J-3)13,14,13                                                   DRKGS176
   13 X=X+.5D0*H                                                        DRKGS177
   14 CALL FCT(X,Y,DERY)                                                DRKGS178
      GOTO 10                                                           DRKGS179
C     END OF INNERMOST RUNGE-KUTTA LOOP                                 DRKGS180
C                                                                       DRKGS181
   15 CALL FCT(X,Y,DERY)
   30 IF (IEND) 4,4,41
   41 CALL OUTP(X,Y,DERY,IHLF,NDIM,PRMT)
   40 RETURN                                                            DRKGS262
      END                                                               DRKGS263
```

APPENDIX F

COMPUTATIONAL ALGORITHM FOR PARAMETER IDENTIFICATION OF DYNAMICAL SYSTEMS

In this appendix a brief discussion of an algorithm for identifying the parameters of a dynamical system by the method of seeking principal planes (PRSEEK) is presented. A listing of the algorithm is presented at the end of this appendix.

The PRSEEK algorithm is based upon the theoretical results which were presented in Chapters II and IV.

The Main program and subroutines TPSEEK, INPUT, SCAFACT, FCT, and OUTP are listed at the end of this appendix. The subroutines FORTIN, PARDIF, DVAL, and ISTORE are also needed, and a description and listing of these subroutines has already been provided in Appendix A. The subroutine EIGEN is also needed, and it has been described and listed in Appendix B. The subroutine DRKF is also needed and it has been described and listed in Appendix E.

A brief description of each of the subroutines which are listed at the end of this appendix will now be provided.

Subroutine PRSEEK calculates the distance to each of the principal planes. It is basically the equivalent of the SEEK subprogram given in Appendix B for solving algebraic

minimization problems.

Subroutine INPUT is used for reading the input data from the data cards.

Subroutine SCAFCT evaluates the scalar function and the partial derivatives of the scalar function.

Subroutine FCT provides the equations for the derivatives of the dynamical equations.

Subroutine OUTP prints the response of the dynamical equations.

Users Guide for the PRSEEK Algorithm

The user need only supply the appropriate data cards in order to use the PRSEEK algorithm.

The first data card has FORMAT (5I5) and contains the order of the dynamical system in columns 1 to 5, the number of free initial conditions in columns 6 to 10, the number of free parameters in columns 11 to 15, the number of time steps between the print out of the dynamical response in columns 16 to 20, and the number of steps to be used in the numerical integration in columns 11 to 25.

The second data card has FORMAT (3D15.7) and contains the initial time in columns 1 to 15, the final time in columns 16 to 30, and an estimate for the minimum of the performance index in columns 31 to 45.

The state model must be set up so that the initial conditions of the first $N-N_{FREE}$ (N is the order of the system, and N_{FREE} is the number of unknown initial conditions)

state variables are known.

The third data card has FORMAT (5D15.7) and contains the initial conditions for the state vector. A guess is used for the unknown initial conditions.

The fourth data card has FORMAT (5D15.7) and contains a guess for the values of the parameters which are to be adjusted.

The next N data cards contain the equations for the right-hand side of the state model. The state variables must be denoted by X(1) through X(N). The parameters are denoted by X(N+1) through $X(N+N_{PAR})$. (N_{PAR} is the number of parameters free to be adjusted.) The independent variable, time, is denoted by $X(N+N_{PAR}+1)$. The equations must be written in columns 7 through 72. If it is necessary to continue an equation on more than one data card, a C should be placed in column 6 of every data card except the last data card associated with that equation. Column 6 of the last data card associated with a continued equation must be left blank.

The last data card contains the equation for the integrand of the performance index, and the notation for writing this equation is presented in the previous paragraph.

The dynamical equations and convergence characteristics for the spool valve system identification problem which was presented in Chapter IV are given in Figure 41.

The data cards which are necessary for working this

example are presented in Figure 42.

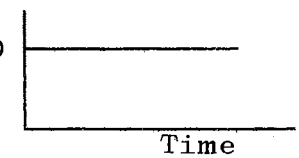
Dynamical Equations

$$\dot{X}_1 = X_2$$

$$\dot{X}_2 = -.36\ X_2 - .24\ X_1 - P_1\ X^3_1 - P_2\ X_1\ X_2 + 1.0$$

Performance Index

$$Y = \int_0^{5.0} (X - X_d)^2\ dt$$



Number of Time Steps = 20
Initial Time = 0.0
Final Time = 5.0
Number of Free Initial Conditions = None

Initial Condition Vector

$X_1 = 0.0$ $X_2 = 0.0$

Initial Parameter Guess

$P_1 = 0.0$ $P_2 = 0.0$

Converged Parameter Vector

$P_1 = .86122$ $P_2 = 1.3174$

Converged After:

21 Evaluations of the performance index
3 Evaluations of gradient vector
15 Evaluations of matrix of second partial derivatives

Figure 41. Equations and Results for the Spool Valve Example

COLUMN NUMBER	0 1 1 2 2 3 4 5 0 5 0 5 0 5
1st. Data Card	2 0 2 2 20
2nd. Data Card	0.0D00 5.0D00 0.0D00
3rd. Data Card	0.0D00 0.0D00
4th. Data Card	0.0D00 0.0D00
5th. Data Card	X(2)
6th. Data Card	-.36*X(2)-.24*X(1)-X(3)*X(1)**3-X(4)*X(1) *X(2)+1.0
7th. Data Card	(X(1)-1.0)**2

Figure 42. Data Cards Required for the Spool Valve Example in Chapter IV

```
00000000011111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      CALL PRSEEK                                                          001
      CALL EXIT                                                            002
      END                                                                  003
      SUBROUTINE PRSEEK                                                    004
C     PURPOSE                                                              005
C       DETERMINE THE BEST VALUES FOR THE PARAMETERS AND THE FREE          006
C       INITIAL CONDITIONS BY THE METHOD OF SEEKING AXES PLANES.           007
C                                                                          008
C     THIS PROGRAM WAS WRITTEN AND PROGRAMED BY DENNIS UNRUH AT THE        009
C     DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING, OKLAHOMA STATE   010
C     UNIVERSITY, STILLWATER, OKLAHOMA.  JULY 1969                         011
C                                                                          012
C       DELFPR -  THE PREDICTED DIFFERENCE BETWEEN THE VALUE OF F AT       013 0023
C        THE AXIS PLANE AND F AT THE BASE POINT.                           014 0024
C       DEN -  SEE NUM/DEN.                                                015 0043
C       DIRECT -  IF THIS NUMBER IS POSITIVE, THEN THE PREDICTED           016 0038
C        GRADIENT DIRECTION AND ACTUAL GRADIENT DIRECTION ARE WITHIN       017 0039
C        90 DEGREES OF EACH OTHER.                                         018 0040
C       EIGVAL -  EIGENVALUE VECTOR OF GX.                                 019 0037
C       F -  FUNCTION BEING MINIMIZED.  F=F(X)                             020 0015
C       FBASE -  VALUE OF F AT BASE POINT                                  021 0027
C       FBEST -  VALUE OF F AT BEST POINT.                                 022 0028
C     FESMIN -  ESTIMATED MINIMUM FOR THE SCALAR FUNCTION.                 023
C       G -  GRADIENT VECTOR OF FUNCTION F.  FOR EXAMPLE G(2)=PARTIAL F/   024 0016
C        PARTIAL X(2).                                                     025 0017
C     GMAG  -  MAGNITUDE OF GRADIENT AFTER NPOS STEPS TO AXIS PLANES.      026
C       GPRED -  PREDICTED VALUE OF GRADIENT AT AXIS PLAIN.                027 0022
C       GX -  MATRIX OF SECOND PARTIALS OF F.  I.E. GX(2,3)=PARTIAL G(2)   028 0018
C        /PARTIAL X(3).                                                    029 0019
C       GXS -  VECTOR WHICH CONTAINS GX IN SCIENTIFIC SUBROUTINE PACKAGE   030 0035
C        SYMMETRIC STORAGE MODE.                                           031 0036
C      N  -  THE NUMBER OF INDEPENDENT VARIABLES.                          032
C      NF  -  NUMBER OF TIMES F HAS BEEN CALCULATED.                       033
C      NG  -  NUMBER OF TIMES THE GRADIENT VECTOR HAS BEEN CALCULATED.     034
C      NGX  -  NUMBER OF TIMES MATRIX GX HAS BEEN CALCULATED.              035
C       NPOS -  NUMBER OF POSITIVE EIGENVALUES.                            036 0029
C       NUM/DEN -  THIS IS THE MAGNITUDE USED ON STEP NPOS+1 PROVIDED      037 0032
C        NPOS SUCCESSFUL STEPS TO THE AXIS PLAINS HAVE BEEN MADE.  THIS    038 0033
C        IS USED ONLY IF NNEG IS NOT ZERO.                                 039 0034
C       NXMAX -  MAXIMUM NUMBER OF INDEPENDENT VARIABLES.                  040 0044
C       PERRAT -  RATIO BETWEEN THE PREDICTED FUNCTION CHANGE AND THE      041 0025
C        ACTUAL FUNCTION CHANGE.                                           042 0026
C       R -  DUMMY VARIABLE NEEDED FOR CALL STATEMENT OF EIGEN.            043 0045
C       WORK -  WORKING VARIABLE.                                          044 0046
C       X -  INDEPENDENT VARIABLE VECTOR.                                  045 0012
C     XBAMIN -  BASE POINT FROM WHICH A MINIMUM IS BEING SOUGHT BY A       046
C      LINEAR SEARCH IN THE PREDICTED NEGATIVE GRADIENT DIRECTION.         047
C       XBASE -  THE VALUE OF THE INDEPENDENT VARIABLE VECTOR AT THE       048 0013
C        BASE POINT.                                                       049 0014
C       XBEST -  THE VALUE OF THE INDEPENDENT VARIABLE AT THE POINT        050 0020
C        WHERE THE CALCULATED VALUE OF F WAS SMALLEST.                     051 0021
C       XNPOS -  THE VALUE OF THE INDEPENDENT VARIABLE AFTER NPOS          052 0041
C        SUCCESSFUL STEPS TO THE AXIS PLAINS.                              053 0042
      IMPLICIT INTEGER*2 (I-N),REAL*8 (A-H,O-Z)                            054
      INTEGER*4 MMM                                                        055
```

```
00000000011111111112222222222333333333344444444445555555556666666667777777778
12345678901234567890123456789012345678901234557890123456789012345678901234567890

      REAL*8 NUM,DEN,MAG                                                    056
      NXMAX=10                                                              057 0047
C     DIMENSION X(NXMAX),G(NXMAX),XBEST(NXMAX),                             058
C     XBASE(NXMAX),GXS((NXMAX*(NXMAX+1))/2),R(1),EIGVAL(NXMAX),             059
C     GPRED(NXMAX),XBAMIN(NXMAX)                                            060
C     ,GXSS(NSMAX*(NXMAX+1))/2)                                             061
      DIMENSION X(10),G(10),XBEST(10),XBASE(10),GXS(55),                    062
     1R(1),EIGVAL(10),GPRED(10),XBAMIN(50)                                  063
     2,GXSS(55)                                                             064
      COMMON /GENER/ NSYS,NFREE,NPAR,NFPNP,NFIXED,NSPNP,NSYSP1              065
  100 FORMAT (8D15.7)                                                       066 0052
  101 FORMAT (' THE EIGENVALUE VECTOR IS')                                  067 0053
  102 FORMAT (6H NPOS=,I5)                                                  068
  103 FORMAT (4HONF=,I5,4H NG=,I5,5H NGX=,I5)                               069 0055
  104 FORMAT (' INDEPENDENT VECTOR X IS')                                   070
  105 FORMAT (3H F=,D15.7)                                                  071 0057
  106 FORMAT (' THE GRADIENT VECTOR IS')                                    072 0058
  107 FORMAT (' GXS(I),I=1,IJMAX')                                          073
  108 FORMAT (8H PERRAT=,D15.7)                                             074 0060
  109 FORMAT (' THE SOLUTION HAS CONVERGED')                                075 0061
  110 FORMAT (8HODIRECT=,D15.7)                                             076 0062
  111 FORMAT (7H FBEST=,D15.7)                                              077 0063
  112 FORMAT (' THE VALUE OF X TO AXIS PLAIN',I5,'   IS')                   078 0064
  113 FORMAT (' THE VALUE OF X AFTER HALVING IS')                           079 0065
  114 FORMAT (' THE VALUE OF X AFTER DOUBLING IS')                          080 0066
  115 FORMAT (I5,D15.7)                                                     081
  116 FORMAT (' THE FUNCTION VALUE IS ALREADY LESS THAN YOUR PREDICTED      082
     1MINIMUM')                                                             083
  117 FORMAT (3H N=,I5,8H FESMIN=,D15.7)                                    084
  118 FORMAT (' GRADIENT EVALUATED AT XBAMIN')                              085
      CALL INPUT(X,FESMIN)                                                  086
      N=NFPNP                                                               087
      IJMAX=((N+1)*N)/2                                                     088
      NF=0                                                                  089 0067
      NG=0                                                                  090 0068
      NGX=0                                                                 091 0069
    3 I=3                                                                   092
      CALL SCAFCT(X,F,G,GXS,I)                                              093
      NGX=NGX+1                                                             094
      WRITE (6,103) NF,NG,NGX                                               095 0085
      WRITE (6,104)                                                         096 0072
      WRITE (6,100) (X(I),I=1,N)                                            097 0073
      WRITE (6,105) F                                                       098 0074
      DO 1 I=1,N                                                            099 0075
    1 XBEST(I)=X(I)                                                         100 0076
      FBEST=F                                                               101 0077
      WRITE (6,106)                                                         102 0082
      WRITE (6,100) (G(I),I=1,N)                                            103 0083
      WRITE (6,107)                                                         104 0088
      WRITE (6,100) (GXS(I),I=1,IJMAX)                                      105
      DO 5 I=1,N                                                            106
    5 XBASE(I)=X(I)                                                         107
      FBASE=F                                                               108 0092
C                                                                           109 0093
C     DETERMINE EIGENVALUES OF GX AND THE NUMBER NPOS.                      110
```

80/80 LIST F-7

```
0000000001111111111222222222233333333334444444444555555555566666666667777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456789
C                                                                          111 0095
      DO 7 I=1,IJMAX                                                       112
    7 GXSS(I)=GXS(I)                                                       113
      MMM=N                                                                114
      CALL EIGEN(GXSS,R,MMM,1)                                             115
      DO 6 I=1,N                                                           116 0101
      II=(I*(I+1))/2                                                       117 0102
    6 EIGVAL(I)=GXSS(II)                                                   118
      NNEG=N                                                               119 0104
      NPOS=N                                                               120 0105
      DO 8 I=1,N                                                           121 0106
      IF (1.0D05*EIGVAL(I).GT.EIGVAL(1)) GO TO 8                           122
      NPOS=NPOS-1                                                          123 0108
    8 CONTINUE                                                             124 0115
      IF (EIGVAL(1).LE.0.0D0) NPOS=0                                       125
      WRITE (6,101)                                                        126 0117
      WRITE (6,100) (EIGVAL(I),I=1,N)                                      127 0118
      WRITE (6,102) NPOS                                                   128
C                                                                          129
C     IF NPOS IS EQUAL TO ZERO , A LINEAR SEARCH IS PERFORMED FOR A        130
C     MINIMUM IN THE NEGATIVE GRADIENT DIRECTION.                          131
C                                                                          132
      IF (NPOS.NE.0) GO TO 52                                              133
      DO 53 I=1,N                                                          134
   53 XBAMIN(I)=X(I)                                                       135
      NUM=FBEST-FESMIN                                                     136
      IF (NUM.LT.0.0D0) GO TO 30                                           137
      DEN=0.0D0                                                            138
      DO 50 I=1,N                                                          139
   50 DEN=DEN+G(I)*G(I)                                                    140
      DO 51 I=1,N                                                          141
   51 X(I)=X(I)-(NUM/DEN)*G(I)                                             142
      GO TO 23                                                             143
C                                                                          144 0120
C     DETERMINE VALUES FOR X TO THE SUCCESSIVE AXIS PLAINS.                145 0121
C                                                                          146 0122
   52 DO 14 K=1,NPOS                                                       147
      DO 9 I=1,N                                                           148 0125
      GPRED(I)=G(I)                                                        149 0126
      DO 9 J=1,N                                                           150 0127
      IJ=((J-1)*J)/2+I                                                     151
      IF (J.LT.I) IJ=((I-1)*I)/2+J                                         152
    9 GPRED(I)=GPRED(I)+GXS(IJ)*(X(J)-XBASE(J))                            153
      DO 10 I=1,N                                                          154 0129
   10 X(I)=X(I)-GPRED(I)/EIGVAL(K)                                         155 0130
      I=1                                                                  156
      CALL SCAFCT(X,F,G,GXS,I)                                             157
      NF=NF+1                                                              158
      WRITE (6,103) NF,NG,NGX                                              159 0132
      WRITE (6,112) K                                                      160 0133
      WRITE (6,100) (X(I),I=1,N)                                           161 0134
      WRITE (6,105) F                                                      162 0135
      IF (F.GT.FBEST.AND.K.EQ.1) GO TO 41                                  163
      IF (F.GT.FBEST) GO TO 60                                             164
C                                                                          165 0140
```

80/80 LIST F-8

```
0000000001111111111222222222233333333334444444444555555555566666666667777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456789
C     CALCULATE PERRAT                                                    166 0141
C                                                                         167 0142
      DELFPR=0.0                                                          168 0143
      DO 13 I=1,N                                                         169 0144
      WORK=0.0                                                            170 0145
      DO 12 J=1,N                                                         171 0146
      IJ=((J-1)*J)/2+I                                                    172
      IF (J.LT.I) IJ=((I-1)*I)/2+J                                        173
   12 WORK=WORK+GXS(IJ)*(X(J)-XBASE(J))                                   174
   13 DELFPR=DELFPR+(G(I)+.50*WORK)*(X(I)-XBASE(I))                       175 0148
      PERRAT =(F-FBASE)/DELFPR                                            176 0149
      WRITE (6,108) PERRAT                                                177 0150
      IF (DABS(PERRAT-1.0D0).GT..1D0) GO TO 3                             178
      DO 11 I=1,N                                                         179 0137
   11 XBEST(I)=X(I)                                                       180 0138
      FBEST=F                                                             181 0139
   14 CONTINUE                                                            182 0152
      DO 15 I=1,N                                                         183 0154
      GPRED(I)=G(I)                                                       184
      DO 15 J=1,N                                                         185 0156
      IJ=((J-1)*J)/2+I                                                    186
      IF (J.LT.I) IJ=((I-1)*I)/2+J                                        187
   15 GPRED(I)=GPRED(I)+GXS(IJ)*(X(J)-XBASE(J))                           188
      I=2                                                                 189
      CALL SCAFCT(X,F,G,GXS,I)                                            190
      NG=NG+1                                                             191
      WRITE (6,103) NF,NG,NGX                                             192 0159
      WRITE (6,106)                                                       193 0162
      WRITE (6,100) (G(I),I=1,N)                                          194 0163
C                                                                         195
C     CHECK FOR CONVERGENCE.                                              196
C                                                                         197
      DO 45 I=1,N                                                         198
      IF (DABS(G(I)).GT.1.0D-8)  GO TO 46                                 199
   45 CONTINUE                                                            200
      GO TO 33                                                            201
   46 IF (NPOS.EQ.N) GO TO 3                                              202
      DIRECT=0.0                                                          203 0164
      DO 16 I=1,N                                                         204 0165
   16 DIRECT=DIRECT+G(I)*GPRED(I)                                         205 0166
      WRITE (6,110) DIRECT                                                206 0167
      IF (DIRECT.LE.0.0) GO TO 3                                          207 0168
      NUM=0.0                                                             208 0169
      DEN=0.0                                                             209 0170
      DO 17 I=1,N                                                         210 0171
      NUM=NUM+(X(I)-XBASE(I))**2                                          211 0172
   17 DEN=DEN+(GPRED(I))**2                                               212 0173
      MAG=DSQRT(NUM/DEN)                                                  213
      DO 18 I=1,N                                                         214 0174
   18 X(I)=X(I)-MAG*GPRED(I)                                              215
C                                                                         216
C     PERFORM A LINEAR SEARCH FOR A MINIMUM IN THE PREDICTED NEGATIVE     217
C     GRADIENT DIRECTION.                                                 218
C                                                                         219
      I=1                                                                 220
```

```
0000000001111111111222222222233333333334444444444555555555566666666667777777778
1234567890123456789012345678901234567890123456789012345678901234567890123456789
      CALL SCAFCT(X,F,G,GXS,I)                                          221
      NF=NF+1                                                           222
      WRITE (6,103) NF,NG,NGX                                           223 0178
      WRITE (6,104)                                                     224 0179
      WRITE (6,100) (X(I),I=1,N)                                        225 0180
      WRITE (6,105) F                                                   226 0181
   41 DO 42 I=1,N                                                       227
   42 XBAMIN(I)=XBEST(I)                                                228
      IF (F.GT.FBEST) GO TO 23                                          229 0182
C                                                                       230
C     CONTINUE DOUBLING THE DISTANCE UNTIL F CEASES TO BECOME BETTER.   231
C                                                                       232
   19 DO 20 I=1,N                                                       233 0183
   20 XBEST(I)=X(I)                                                     234 0184
      FBEST=F                                                           235 0185
      DO 21 I=1,N                                                       236 0186
   21 X(I)=XBAMIN(I)+2.0*(X(I)-XBAMIN(I))                               237
      I=1                                                               238
      CALL SCAFCT(X,F,G,GXS,I)                                          239
      NF=NF+1                                                           240
      WRITE (6,103) NF,NG,NGX                                           241 0189
      WRITE (6,114)                                                     242 0190
      WRITE (6,100) (X(I),I=1,N)                                        243 0191
      WRITE (6,105) F                                                   244 0192
      IF (F.LT.FBEST) GO TO 19                                          245 0193
      DO 22 I=1,N                                                       246 0194
   22 X(I)=XBEST(I)                                                     247 0195
      F=FBEST                                                           248 0196
      GO TO 3                                                           249
C                                                                       250
C     CONTINUE HALVING THE DISTANCE UNTIL F BECOMES BETTER.             251
C                                                                       252
   23 DO 24 I=1,N                                                       253 0198
   24 X(I)=XBAMIN(I)+.50*(X(I)-XBAMIN(I))                               254
      I=1                                                               255
      CALL SCAFCT(X,F,G,GXS,I)                                          256
      NF=NF+1                                                           257
      WRITE (6,103) NF,NG,NGX                                           258 0201
      WRITE (6,113)                                                     259 0202
      WRITE (6,100) (X(I),I=1,N)                                        260 0203
      WRITE (6,105) F                                                   261 0204
      IF (F.GT.FBEST) GO TO 23                                          262 0205
      DO 25 I=1,N                                                       263 0206
   25 XBEST(I)=X(I)                                                     264 0207
      FBEST=F                                                           265 0208
      GO TO 3                                                           266
   60 DO 61 I=1,N                                                       267
   61 XBAMIN(I)=XBEST(I)                                                268
      I=2                                                               269
      CALL SCAFCT(XBAMIN,F,G,GXS,I)                                     270
      NG=NG+1                                                           271
      WRITE (6,103) NF,NG,NGX                                           272
      WRITE (6,118)                                                     273
      WRITE (6,100) (G(I),I=1,N)                                        274
      DIRECT=0.0                                                        275
```

80/80 LIST F-10

```
00000000011111111112222222222333333333344444444445555555555666666666677777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      DO 62 I=1,N                                                          276
   62 DIRECT=DIRECT+G(I)*GPRED(I)                                          277
      WRITE (6,110) DIRECT                                                 278
      IF (DIRECT.GT.0.0D0) GO TO 23                                        279
      DO 63 I=1,N                                                          280
   63 X(I)=XBAMIN(I)                                                       281
      GO TO 3                                                              282
   30 WRITE (6,116)                                                        283
      CALL EXIT                                                            284
   33 WRITE (6,109)                                                        285
      CALL EXIT                                                            286 0241
      RETURN                                                               287 0242
      END                                                                  288 0243
      SUBROUTINE INPUT(X,FESMIN)                                           289
      IMPLICIT INTEGER*2(I-N),REAL*8(A-H,O-Z)                              290
C     PURPOSE                                                              291
C       READ DATA CARDS.                                                   292
C         DATA CARD ONE CONTAINS NSYS,NFREE,NPAR,NWRITE,NSTEP              293
C         DATA CARD TWO CONTAINS PRMT(1),PRMT(2),FESMIN                    294
C         DATA CARD THREE CONTAINS THE INITIAL CONDITION GUESS FOR         295
C          COMPONENTS 1 THROUGH NSYS OF Y.                                 296
C         DATA CARD FOUR CONTAINS THE INITIAL GUESS FOR THE PARAMETERS.    297
C          IF NPAR=0 LEAVE THIS DATA CARD OUT.                             298
C         THE SYSTEM EQUATIONS ARE ENTERED ON CARDS 5 THROUGH NSYS+4.      299
C         THE EQUATION FOR THE INTEGRAN OF THE SCALAR PERFORMANCE INDEX    300
C          IS ENTERED ON THE LAST DATA CARD.                               301
      DIMENSION X(10)                                                      302
      COMMON /GENER/ NSYS,NFREE,NPAR,NFPNP,NFIXED,NSPNP,NSYSP1             303
      COMMON /INFC/ IF(10),IFX(10,10),IFXXS(10,55)                         304
      COMMON /INOU/ IG,IGX(10),IGXXS(55),NWRITE                            305
      COMMON /SCIN/ YI(10),PRMT(5)                                         306
      COMMON /IOFCSC/ P(5)                                                 307
  100 FORMAT (8D15.7)                                                      308
  101 FORMAT (5D15.7)                                                      309
  102 FORMAT (10I5)                                                        310
  103 FORMAT (' NSYS,NFREE,NPAR,NWRITE,NSTEP')                             311
  104 FORMAT ('   PRMT(1)     PRMT(2)       FESMIN')                       312
  106 FORMAT (5(I5,F10.5))                                                 313
  108 FORMAT (' INITIAL CONDITION USED')                                   314
  109 FORMAT (' PARAMETER VECTOR')                                         315
  110 FORMAT (' IF(',I4,')=')                                              316
  111 FORMAT (' IFX(',2I4,')=')                                            317
  112 FORMAT (' IFXXS(',2I4,')=')                                          318
  113 FORMAT (' IG=')                                                      319
  114 FORMAT (' IGX(',I4,')=')                                             320
  115 FORMAT (' IGXXS(',I4,')=')                                           321
      READ(5,102) NSYS,NFREE,NPAR,NWRITE,NSTEP                             322
      WRITE (6,103)                                                        323
      WRITE (6,102) NSYS,NFREE,NPAR,NWRITE,NSTEP                           324
      READ(5,101) PRMT(1),PRMT(2),FESMIN                                   325
      WRITE (6,104)                                                        326
      WRITE (6,100) PRMT(1),PRMT(2),FESMIN                                 327
      READ(5,101) (YI(I),I=1,NSYS)                                         328
      WRITE (6,108)                                                        329
      WRITE (6,100) (YI(I),I=1,NSYS)                                       330
```

```
00000000011111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      IF (NPAR.EQ.0) GO TO 2                                                    331
      READ (5,101) (P(I),I=1,NPAR)                                              332
      WRITE (6,109)                                                             333
      WRITE (6,101) (P(I),I=1,NPAR)                                             334
    2 NFPNP=NFREE+NPAR                                                          335
      NFIXED=NSYS-NFREE                                                         336
      NSPNP=NSYS+NPAR                                                           337
      NSYSP1=NSYS+1                                                             338
      RNSTEP=NSTEP                                                              339
      PRMT(3)=(PRMT(2)-PRMT(1))/RNSTEP                                          340
      IF (NFREE.EQ.0) GO TO 4                                                   341
      DO 3 I=1,NFREE                                                            342
      II=NFIXED+I                                                               343
    3 X(I)=YI(II)                                                               344
    4 IF (NPAR.EQ.0) GO TO 6                                                    345
      DO 5 I=1,NPAR                                                             346
      II=NFREE+I                                                                347
    5 X(II)=P(I)                                                                348
    6 DO 7 I=1,NSYS                                                             349
      WRITE (6,110) I                                                           350
      CALL FORTIN(IF(I))                                                        351
      DO 7 K=1,NSPNP                                                            352
      WRITE (6,111) I,K                                                         353
      CALL PARDIF(IF(I),IFX(I,K),K)                                             354
      DO 7 J=1,K                                                                355
      JK=((K-1)*K)/2+J                                                          356
      WRITE (6,112) I,JK                                                        357
    7 CALL PARDIF(IFX(I,K),IFXXS(I,JK),J)                                       358
      WRITE (6,113)                                                             359
      CALL FORTIN (IG)                                                          360
      DO 8 J=1,NSYS                                                             361
      WRITE (6,114) J                                                           362
      CALL PARDIF(IG,IGX(J),J)                                                  363
      DO 8 I=1,J                                                                364
      IJ=((J-1)*J)/2+I                                                          365
      WRITE (6,115) IJ                                                          366
    8 CALL PARDIF(IGX(J),IGXXS(IJ),I)                                           367
      RETURN                                                                    368
      END                                                                       369
      SUBROUTINE FCT(T,Y,DERY)                                                  370
C     PURPOSE                                                                   371
C       SUPPLYS THE EQUATIONS FOR THE DYNAMICAL SYSTEM.                         372
      IMPLICIT INTEGER*2(I-N),REAL*8 (A-H,O-Z)                                  373
      DIMENSION Y(210),DERY(210),Z(50),FX(10,10),FXXS(10,55),                   374
     1B(10,10),C(10)                                                            375
      COMMON /GENER/ NSYS,NFREE,NPAR,NFPNP,NFIXED,NSPNP,NSYSP1                  376
      COMMON /INFC/ IF(10),IFX(10,10),IFXXS(10,55)                              377
      COMMON /SCFCOU/ ILOGIC                                                    378
      COMMON /IOFCSC/ P(5)                                                      379
      DO 1 I=1,NSYS                                                             380
    1 Z(I)=Y(I)                                                                 381
      IF (NPAR.EQ.0) GO TO 3                                                    382
      DO 2 I=1,NPAR                                                             383
      II=NSYS+I                                                                 384
    2 Z(II)=P(I)                                                                385
```

```
00000000011111111112222222222333333333344444444445555555555666666666677777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

     3 II=NSYS+NPAR+I                                                          386
       Z(II)=T                                                                 387
C                                                                              388
C      DETERMINE THE VALUES FOR THE DERIVITIVE OF THE BASIC SYSTEM.            389
C                                                                              390
    10 DO 11 I=1,NSYS                                                          391
    11 DERY(I)=DVAL(IF(I),Z)                                                   392
       IF (ILOGIC.EQ.1) RETURN                                                 393
C                                                                              394
C      DETERMINE THE VALUES FOR THE DERIVITIVES OF THE FIRST ORDER             395
C      INFLUENCE COEFFICIENTS.                                                 396
    20 DO 21 I=1,NSYS                                                          397
       DO 21 J=1,NSYS                                                          398
    21 FX(I,J)=DVAL(IFX(I,J),Z)                                                399
       DO 22 I=1,NSYS                                                          400
       DO 22 J=1,NFPNP                                                         401
       IJ=J*NSYS+I                                                             402
       DERY(IJ)=0.0                                                            403
       DO 22 K=1,NSYS                                                          404
       JK=J*NSYS+K                                                             405
    22 DERY(IJ)=DERY(IJ)+FX(I,K)*Y(JK)                                         406
       IF (NPAR.EQ.0) GO TO 24                                                 407
       DO 23 J=1, NPAR                                                         408
       DO 23 I=1,NSYS                                                          409
       IJ=(NFREE+J)*NSYS+I                                                     410
       II=NSYS+J                                                               411
    23 DERY(IJ)=DERY(IJ)+DVAL(IFX(I,II),Z)                                     412
    24 IF (ILOGIC.EQ.2) RETURN                                                 413
C                                                                              414
C      DETERMINE THE DERIVITIVES FOR THE SECOND ORDER INFLUENCE                415
C      COEFFICIENTS.                                                           416
C                                                                              417
       DO 25 I=1,NSYS                                                          418
       DO 25 K=1,NSPNP                                                         419
       DO 25 J=1,K                                                             420
       JK=((K-1)*K)/2+J                                                        421
    25 FXXS(I,JK)=DVAL(IFXXS(I,JK),Z)                                          422
       DO 32 J=1,NFPNP                                                         423
       DO 30 LI=1,NSYS                                                         424
       DO 30 LJ=1,NSYS                                                         425
       B(LI,LJ)=0.0                                                            426
       IF (J.LE.NFREE) GO TO 26                                                427
       JJ=NSYS+J-NFREE                                                         428
       LJJJ=((JJ-1)*JJ)/2+LJ                                                   429
       B(LI,LJ)=FXXS(LI,LJJJ)                                                  430
    26 CONTINUE                                                                431
       DO 30 LK=1,NSYS                                                         432
       JJ=J*NSYS+LK                                                            433
       LJLK=((LK-1)*LK)/2+LJ                                                   434
       IF (LK.LT.LJ) LJLK=((LJ-1)*LJ)/2+LK                                     435
    30 B(LI,LJ)=B(LI,LJ)+Y(JJ)*FXXS(LI,LJLK)                                   436
       DO 32 I=1,J                                                             437
       IF (J.LE.NFREE.OR.I.LE.NFREE) GO TO 33                                  438
       DO 31 K=1,NSYS                                                          439
       JJ=NSYS+J-NFREE                                                         440
```

```
00000000011111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      II=NSYS+I-NFREE                                                    441
      IIJJ=((JJ-1)*JJ)/2+II                                              442
      C(K)=FXXS(K,IIJJ)                                                  443
      DO 31 LI=1,NSYS                                                    444
      LII=((II-1)*II)/2+LI                                               445
      IF (LI.GT.II) LII=((LI-1)*LI)/2+II                                 446
      LIJ=J*NSYS+LI                                                      447
   31 C(K)=C(K)+FXXS(K,LII)*Y(LIJ)                                       448
   33 CONTINUE                                                           449
      DO 32 K=1,NSYS                                                     450
      IJ=((J-1)*J)/2+I                                                   451
      KK=(NFPNP+IJ)*NSYS+K                                               452
      DERY(KK)=0.0                                                       453
      IF (J.GT.NFREE.AND.I.GT.NFREE) DERY(KK)=C(K)                       454
      DO 32 L=1,NSYS                                                     455
      LL=(NFPNP+((J-1)*J)/2+I)*NSYS+L                                    456
      LLL=I*NSYS+L                                                       457
   32 DERY(KK)=DERY(KK)+FX(K,L)*Y(LL)+B(K,L)*Y(LLL)                      458
      RETURN                                                             459
      END                                                                460
      SUBROUTINE SCAFCT (X,F,G,GXS,ILOG)                                 461
C     PURPOSE                                                            462
C       SETS UP THE INITIAL CONDITIONS FOR THE DYNAMICAL EQUATIONS.      463
C       EVALUATES THE SCALAR PERFORMANCE FUNCTION,ITS GRADIENT, AND      464
C       ITS MATRIX OF SECOND PARTIAL DERIVITIVES.                        465
      IMPLICIT INTEGER*2 (I-N),REAL*8 (A-H,O-Z)                          466
      INTEGER*4 NTOTAL,IHLF                                              467
      DIMENSION X(10),G(10),GXS(55),Y(210),DERY(210),AUX(210)            468
      COMMON /GENER/ NSYS,NFREE,NPAR,NFPNP,NFIXED,NSPNP,NSYSP1           469
      COMMON/SCOU/ GRAL(21)                                              470
      COMMON /SCIN/ YI(10),PRMT(5)                                       471
      COMMON /SCFCOU/ ILOGIC                                             472
      COMMON /IOFCSC/ P(5)                                               473
      EXTERNAL FCT,OUTP                                                  474
      DATA NDIFTO /0/                                                    475
  100 FORMAT (' NDIFTO=',I5)                                             476
      ILOGIC=ILOG                                                        477
      IF (NFIXED.EQ.0) GO TO 4                                           478
      DO 1 I=1,NFIXED                                                    479
    1 Y(I)=YI(I)                                                         480
    4 IF (NFREE.EQ.0) GO TO 3                                            481
      DO 2 I=1,NFREE                                                     482
      II=I+NFIXED                                                        483
    2 Y(II)=X(I)                                                         484
    3 IF (NPAR.EQ.0) GO TO 6                                             485
      DO 5 I=1,NPAR                                                      486
      II=NFREE+I                                                         487
    5 P(I)=X(II)                                                         488
    6 GO TO (10,20,30), ILOGIC                                           489
C                                                                        490
C     DETERMINE THE VALUE FOR THE SCALAR PERFORMANCE FUNCTION, F.        491
C                                                                        492
   10 NTOTAL=NSYS                                                        493
      NDIFTO=NDIFTO+NTOTAL                                               494
      WRITE (6,100) NDIFTO                                               495
```

```
00000000011111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      CALL DRKF(PRMT,Y,DERY,NTOTAL,IHLF,FCT,OUTP,AUX)                          496
      F=GRAL(1)                                                               497
      RETURN                                                                  498
C                                                                             499
C     DETERMINE THE VALUE FOR THE GRADIENT OF THE SCALAR PERFORMANCE          500
C     FUNCTION, G.                                                            501
C                                                                             502
   20 NTOTAL=(NFPNP+1)*NSYS                                                   503
      DO 21 I=NSYSP1,NTOTAL                                                   504
   21 Y(I)=0.0                                                                505
      IF (NFREE.EQ.0) GO TO 23                                                506
      DO 22 I=1,NFREE                                                         507
      II=NSYS*I+NFIXED+I                                                      508
   22 Y(II)=1.0                                                               509
      NDIFTO=NDIFTO+NTOTAL                                                    510
      WRITE (6,100) NDIFTO                                                    511
   23 CALL DRKF(PRMT,Y,DERY,NTOTAL,IHLF,FCT,OUTP,AUX)                         512
      DO 24 I=1,NFPNP                                                         513
   24 G(I)=GRAL(I+1)                                                          514
      RETURN                                                                  515
C                                                                             516
C     DETERMINE THE VALUES FOR THE ELEMENTS OF THE MATRIX OF SECOND           517
C     PARTIAL DERIVITIVES OF THE SCALAR PERFORMANCE FUNCTION.                 518
C                                                                             519
   30 NTOTAL=(1+NFPNP+((NFPNP+1)*NFPNP)/2)*NSYS                               520
      DO 31 I=NSYSP1,NTOTAL                                                   521
   31 Y(I)=0.0                                                                522
      IF (NFREE.EQ.0) GO TO 33                                                523
      DO 32 I=1,NFREE                                                         524
      II=NSYS*I+NFIXED+I                                                      525
   32 Y(II)=1.0                                                               526
      NDIFTO=NDIFTO+NTOTAL                                                    527
      WRITE (6,100) NDIFTO                                                    528
   33 CALL DRKF(PRMT,Y,DERY,NTOTAL,IHLF,FCT,OUTP,AUX)                         529
      F=GRAL(1)                                                               530
      DO 34 J=1,NFPNP                                                         531
      G(J)=GRAL(J+1)                                                          532
      DO 34 I=1,J                                                             533
      IJ=((J-1)*J)/2+I                                                        534
      II=NFPNP+1+IJ                                                           535
   34 GXS(IJ)=GRAL(II)                                                        536
      RETURN                                                                  537
      END                                                                     538
      SUBROUTINE OUTP(T,Y,DERY,IHLF,NTOTAL,PRMT)                              539
C     PURPOSE                                                                 540
C       WRITE THE SYSTEM RESPONSE AFTER EVERY NWRITE STEPS OF NUMERICAL       541
C       INTEGRATION.  NUMERICALLY INTEGRATE THE INTEGRAL EQUATIONS BY         542
C     USING THE TRAPAZOID RULE.                                               543
      IMPLICIT INTEGER*2(I-N),REAL*8 (A-H,O-Z)                                544
      INTEGER*4 IHLF,NTOTAL                                                   545
      DIMENSION Y(210),DERY(210),Z(50),GRAN(21),GRANPR(21),                   546
     1GX(10),GXXS(55),B(10),PRMT(5)                                           547
      COMMON /GENER/ NSYS,NFREE,NPAR,NFPNP,NFIXED,NSPNP,NSYSP1                548
      COMMON/SCOU/ GRAL(21)                                                   549
      COMMON /INOU/ IG,IGX(10),IGXXS(55),NWRITE                               550
```

```
00000000011111111112222222222333333333344444444445555555555666666666677777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

      COMMON /SCFCOU/ ILOGIC                                          551
      COMMON /IOFCSC/ P(5)                                            552
  100 FORMAT (1HO,D15.7)                                              553
  101 FORMAT (8D15.7)                                                 554
  102 FORMAT (' THE INTEGRAL VECTOR IS')                              555
      DO 1 I=1,NSYS                                                   556
    1 Z(I)=Y(I)                                                       557
      IF (NPAR.EQ.0) GO TO 3                                          558
      DO 2 I=1,NPAR                                                   559
      II=NSYS+I                                                       560
    2 Z(II)=P(I)                                                      561
    3 II=NSYS+NPAR+1                                                  562
      Z(II)=T                                                         563
C                                                                     564
C     SET UP THE VALUE FOR THE INTEGRAN OF THE SCALAR PERFORMANCE     565
C     FUNCTION.                                                       566
      GRAN(1)=DVALUE(G,Z)                                             567
      NINT=1                                                          568
      IF (ILOGIC.EQ.1) GO TO 30                                       569
C                                                                     570
C     SET UP THE VALUES FOR THE INTEGRAN OF THE GRADIENT OF THE SCALAR 571
C     PERFORMANCE FUNCTION.                                           572
C                                                                     573
      DO 10 I=1,NSYS                                                  574
   10 GX(I)=DVALUE(GX(I),Z)                                           575
      NINT=NFPNP+1                                                    576
      DO 11 I=2,NINT                                                  577
      GRAN(I)=0.0                                                     578
      DO 11 J=1,NSYS                                                  579
      JJ=(I-1)*NSYS+J                                                 580
   11 GRAN(I)=GRAN(I)+GX(J)*Y(JJ)                                     581
      IF (ILOGIC.EQ.2) GO TO 30                                       582
C                                                                     583
C     SET UP THE VALUES FOR THE INTEGRAN OF THE MATRIX OF SECOND      584
C     PARTIAL DERIVITIVES OF THE SCALAR FUNCTION.                     585
C                                                                     586
      DO 20 J=1,NSYS                                                  587
      DO 20 I=1,J                                                     588
      IJ=((J-1)*J)/2 +I                                               589
   20 GXXS(IJ)=DVALUE(GXXS(IJ),Z)                                     590
      DO 22 J=1,NFPNP                                                 591
      DO 21 LI=1,NSYS                                                 592
      B(LI)=0.0                                                       593
      DO 21 LJ=1,NSYS                                                 594
      LIJ=((LJ-1)*LJ)/2+LI                                            595
      IF (LJ.LT.LI) LIJ=((LI-1)*LI)/2+LJ                              596
      LJJ=J*NSYS+LJ                                                   597
   21 B(LI)=B(LI)+GXXS(LIJ)*Y(LJJ)                                    598
      DO 22 I=1,J                                                     599
      IJ=((J-1)*J)/2+I                                                600
      II=1+NFPNP+IJ                                                   601
      GRAN(II)=0.0                                                    602
      DO 22 K=1,NSYS                                                  603
      KK=I*NSYS+K                                                     604
      KKK=(NFPNP+IJ)*NSYS+K                                           605
```

```
0000000001111111111222222222233333333334444444444555555555566666666677777777788
1234567890123456789012345678901234567890123456789012345678901234567890123456789
     GRAN(II)=GRAN(II)+Y(KK)*B(K)+GX(K)*Y(KKK)                                  606
     NINT=1+NFPNP+((NFPNP+1)*NFPNP)/2                                           607
  30 IF (T.EQ.PRMT(1)) GO TO 36                                                 608
     DO 34 I=1,NINT                                                             609
     GRAL(I)=GRAL(I)+.50*(GRAN(I)+GRANPR(I))*(T-TPR)                            610
  34 GRANPR(I)=GRAN(I)                                                          611
     TPR=T                                                                      612
     IF (T.EQ.PRMT(2)) GO TO 41                                                 613
     IF (NCOUNT.EQ.NWRITE) GO TO 41                                             614
     NCOUNT=NCOUNT+1                                                            615
     RETURN                                                                     616
  36 DO 37 I=1,NINT                                                             617
     GRAL(I)=0.0                                                                618
  37 GRANPR(I)=GRAN(I)                                                          619
     TPR=T                                                                      620
C                                                                               621
C    RESET NCOUNT AND WRITE OUT THE VALUES AT THIS TIME.                        622
C                                                                               623
  41 NCOUNT=1                                                                   624
     WRITE (6,100) T                                                            625
     DO 42 I=1,NINT                                                             626
     JJ=(I-1)*NSYS+1                                                            627
     II=I*NSYS                                                                  628
  42 WRITE (6,101) (Y(J),J=JJ,II)                                               629
     WRITE (6,102)                                                              630
     WRITE (6,101) (GRAL(I),I=1,NINT)                                           631
     RETURN                                                                     632
     END                                                                        633
```

VITA

Dennis Ray Unruh

Candidate for the Degree of

Doctor of Philosophy

Thesis: DETERMINATION OF OPTIMAL PARAMETERS FOR DYNAMICAL SYSTEMS

Major Field: Engineering

Biographical:

Personal Data: Born in Great Bend, Kansas, November 21, 1943, the son of Mr. and Mrs. Harvey Unruh.

Education: Graduated from Pawnee Rock High School, Pawnee Rock, Kansas in June, 1961; attended Bethel College, North Newton, Kansas from 1961 to 1964; received the Bachelor of Science degree from Kansas State University, Manhattan, Kansas in June, 1966; received the Master of Science degree from Oklahoma State University, Stillwater, Oklahoma, in May, 1968; completed requirements for the Doctor of Philosophy degree in May, 1970.

Professional Experience: Employed by Cessna Aircraft Co., Industrial Products Division, as a design engineer from June to September of 1965 and 1966; employed by Oklahoma State University as a graduate research assistant from September, 1966 to September, 1969; employed by Cessna Aircraft Co., Industrial Products Division, as a project engineer from September, 1969 to present.