ON THE FEASIBILITY OF ADAPTIVE CONTROL WITHOUT IDENTIFICATION

ON THE FEASIBILITY OF ADAPTIVE CONTROL WITHOUT IDENTIFICATION

by

MUHAMMAD JAVED IQLEEM, B.Sc., B.Sc. (Elect. Eng.)

A Thesis

Submitted to the Faculty of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Master of Engineering

McMaster University

February 1967

MASTER OF ENGINEERING — McMASTER UNIVERSITY

(Electrical Engineering) — Hamilton, Ontario

TITLE: On the Feasibility of Adaptive Control Without Identification

AUTHOR: Muhammad Javed Iqleem B.Sc. (Panjab University-Pakistan)
B.Sc. Electrical Eng. (West Pakistan University of Engineering and Technology-Pakistan)

SUPERVISOR: Dr. N. K. Sinha

NUMBER OF PAGES: vii, 110

SCOPE AND CONTENTS:-

One of the two basic philosophies underlying adaptive control is that the transfer function of the plant must be first determined and then the values of an adjustable controller varied for optimizing a given index of performance. The process of identifying the plant characteristics is popularly known as Identification Problem and constitutes a major problem in the realization of an adaptive system of this type.

The other philosophy is that a complete knowledge of the plant is not necessary for the optimum adjustments of the parameter of control. The system is caused to measure its own performance against a figure of merit and drives its performance towards optimum. This approach is becoming popular because of the many difficulties associated with the identification problem and a number of "hill climbing" techniques have been proposed based on this philosophy.

In this thesis, three such techniques (steepest descent, conjugate gradients and parallel tangents) have been analysed with a view to determine the most efficient and quickest way to determine the parameters of a controller for optimum performance.

ACKNOWLEDGEMENTS

The author wishes to acknowledge with gratitude, the guidance and encouragement of Dr. N. K. Sinha in preparation of this thesis. The author is also grateful to the Canadian Government for providing a scholarship under the Colombo Plan to carry out this work.

Thanks are also due to R. H. Sheikh and B. Sahay for their helpful suggestions in the Computer Programming, and to Miss Marie Kaneary for typing this manuscript.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

CHAPTER 1

INTRODUCTION

The essential feature of everyautomatic control system is feedback. Application of feedback techniques has enabled man to construct a large variety of automatic control devices by which he can control physical phenomena in a desired fashion. These systems, however, have shown a most significant limitation: They perform a particular task under desired or anticipated operating conditions; should the operating conditions change, they show restricted compatability. As a matter of fact, when the parameters of the devices to be controlled are invariant, and when the operating characteristics of the controller elements can be expected to be unchanging, even feedback may be unnecessary. The feedback is useful because it enables a specified level of automatic control performance to be maintained despite small changes in the controlled parameters, but when such changes become large the performance of the system deteriorates.

The last decade has been noticeable in the field of automatic control system for the development of the concept of adaptive control—— or control in which the system is capable of modifying its own parameters so as to remain efficient despite the changes in the environments. Adaptation to unpredictable conditions is in fact one of the basic requirements of a control system. This problem has certainly long been

recognised, but until recently not enough work was carried out in this direction. The growing interest in the field of adaptive control is evident from the numerous papers published in the last few years [1], [2], [3], [4], [5], [6], [7], [8]. Current interest in the adaptability of a control system is the result of progress being made in space, nuclear and other complex industrial technologies, where an attempt has been made to overcome the limitations of conventional design philosophy and meet the higher performance requirements.

There are two basic approaches underlying adaptive control. One is that the transfer function of the plant must be first determined and then the values of the parameters of an adjustable controller can be obtained for optimizing a given index of performance or figure of merit. The process of determining the characteristics of the plant is called "plant identification" and constitutes a major problem in the realization of an adaptive system of this type.

The other philosophy is that a complete knowledge of the plant to be controlled is not necessary for the optimum adjustments of the parameters of the control. The system is caused to measure its own performance against a figure of merit and drives its performance towards optimum. A number of"hill-climbing techniques" based on this philosophy have been proposed [9], [10]. This approach is becoming popular [11], [12] because of the many difficulties associated with the identification problem.

In this thesis, a very important aspect of adaptive control without identification is investigated. This is: how quickly can the variable parameters of the controller be adjusted in order to optimize

the performance of the system. Some of the optimizing techniques proposed in the past few years have been analyzed in the present work with a view to find the most efficient and quickest way to determine the parameters of a controller for optimum performance. The optimization calculations in this work have been carried out on an IBM-7040 computer using WATFOR and IBM compilers.

CHAPTER 2

THE CONCEPT OF ADAPTIVE CONTROL

2.1 INTRODUCTION:

With the progress being made in space, nuclear, and other industrial technologies, there is a growing need for automatic control systems which are capable of changing their own parameters in order to remain efficient in spite of large changes in their environments. Adaptive control has been viewed as the instrumentation realization of a prime characteristic of the human being in a control task. This has led to a good deal of work during the past few years on adaptive control systems.

In the literature on adaptive control systems one finds that different research groups have used their own terms and definitions, many of them overlap each other. A general scheme of classification of adaptive control systems has been proposed by Sinha [13], which clarifies the confusion created by the different overlapping definitions.

2.2 DEFINITION OF ADAPTIVE CONTROL:

Control systems can be divided into two main classes: adaptive and non-adaptive. Adaptive control systems may be defined as those which are capable of modifying their own parameters with changes in environments in such a manner that their performance is optimized on the basis of a proscribed criterion. Non-adaptive systems do not have this facility.

The changes in the environments of a control system can be either in the statistical properties of the input or in the plant dynamics, or both. Whenever such changes take place they must be identified and the corresponding compensatory adjustments made in the controller for the optimum operation.

Figure (1) shows the block diagram of an adaptive control system. All adaptive control systems perform some of the following operations____ measurement, identification, determination of optimum control strategy and modification of the controller.

2.3 IDENTIFICATION PROBLEM:

Most control systems consist of two sub-systems: plant and controller. The plant is considered to be the mechanism to be controlled and has little design freedom in most cases. The controller is that part of the system which is designed with a view toward making the entire system work properly. Evidently the success with which a given plant can be controlled in a desired fashion depends on how accurately its dynamic characteristics are known. The basic control system in its simplest form is shown in Figure (2). Knowing the plant transfer function, any desired input-output relationship can in principle be obtained by designing a controller with transfer function $G_c(s)\ \frac{1}{G_p(s)}$. $G_c(s)$ represents the desired transfer function. This approach fails in almost all the cases, becuase

(i) $G_p(s)$ is varying during the course of normal operation.

(ii) $G_p(s)$ is not accurately known in advance of design.

(iii) $G_p(s)$ is non-minimum-phase.

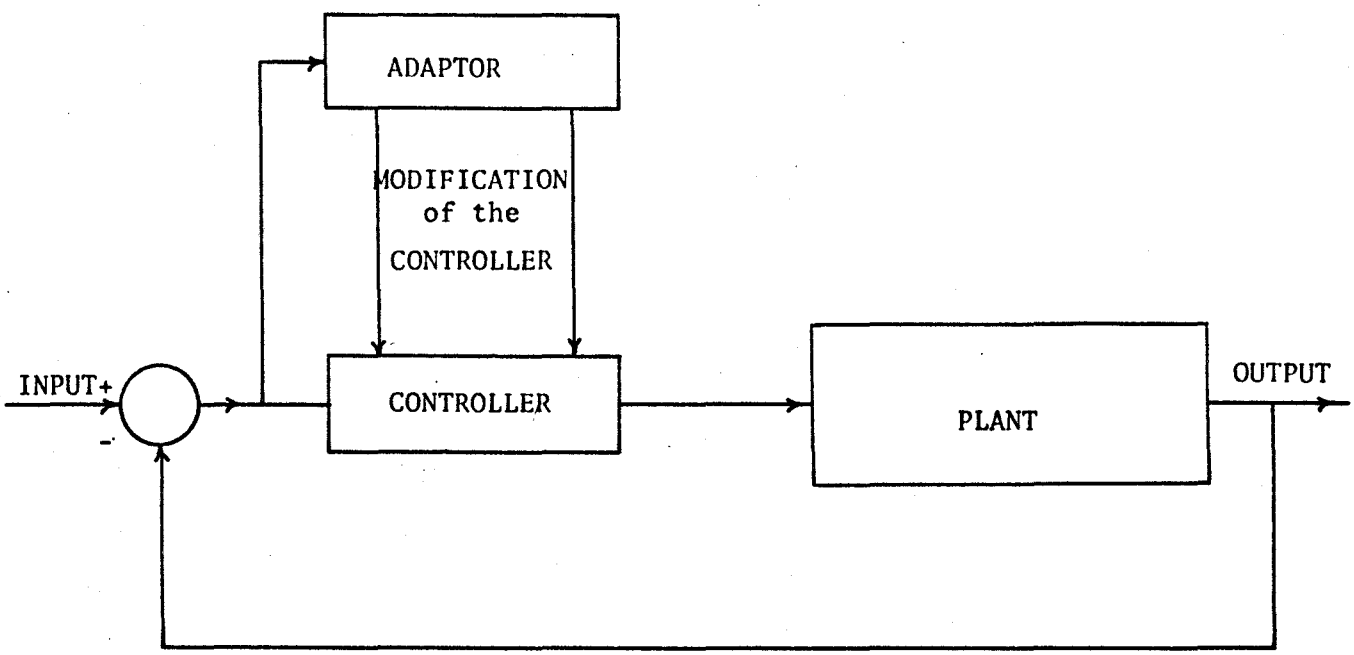


FIGURE 1: Block diagram of a basic adaptive control system

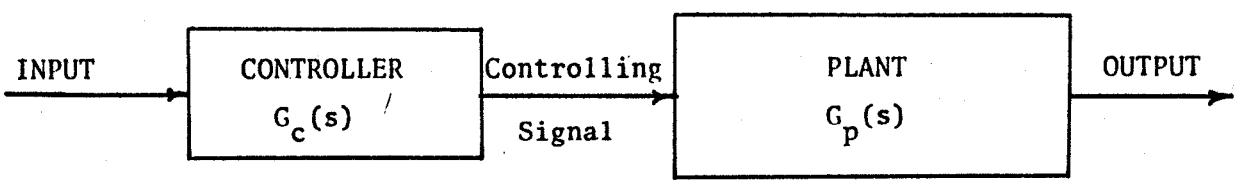


FIGURE 2: Control problem

To overcome these difficulties, the concept of feedback was introduced. This is shown in Figure (3). Such an arrangement can overcome the problems mentioned above only if the overall transmission is independent of $G_p(s)$.

The overall transfer function is now:

$$\frac{G_c(s)\ G_p(s)}{1 + G_c(s)\ G_p(s)\ H(s)}$$

For overall transmission to be independent of $G_p(s)$, we should have:

$$\left| G_c(s)\ G_p(s)\ H(s) \right| \gg 1$$

This condition reduces the overall transfer function to $\simeq \frac{1}{H(s)}$.
Unfortunately this condition cannot be fulfilled over a wide range of frequencies due to considerations of stability and unavoidable random noise. In either case, if the plant to be controlled is quite large, or if the control system is to reach optimum performance, it is desirable that the design of the controller be based upon as much knowledge of the process dynamics as possible.

The identification problem, i.e., the process of characterising the plant dynamics is a major problem in adaptive control systems. A lot of work has been done during the recent years and a detailed account is presented in reference [4].

2.4 DIFFICULTY OF IDENTIFICATION:

The general identification problem involves the use of measured data for the determination of certain unknown parameters. The study of

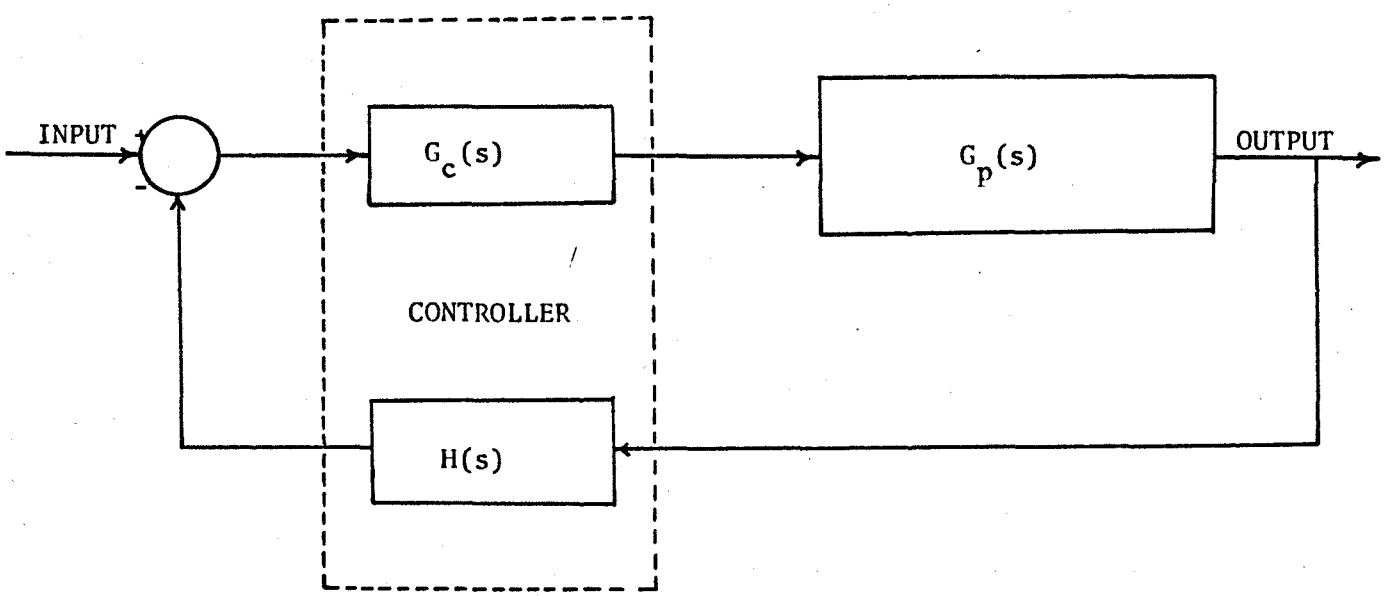


FIGURE 3: Feedback controller

identification in effect demands an answer to such questions as:

(i) What ways are available for characterising the process dynamics?

(ii) What signals (if permissible) are allowed to determine such characteristics?

(iii) How should data from such excitations be processed to obtain the desired characteristics of process dynamics?

(iv) What accuracy can be anticipated?

Unfortunately, the present state of the art of engineering analysis and design can not answer these questions completely, although considerable research work has been done in this field.

The process to be identified may, in general, be non-linear and time varying with multiple inputs and outputs. The difficulty of identification depends mainly upon whether the process is linear or non-linear; it is usual to assume that the given process is linear in this regard the designer cannot help; given a process he has no other choice but to assume the process to be linear and then establish the nature of non-linearities. It is also assumed that the process is time invariant. Accordingly, success in identification depends upon how correct these assumptions made at the outset are. A single input and a single output is another common assumption. It can be said that research efforts in this area have been made on a restrictive basis because of the amount of the work involved in identification.

From a practical point of view, any identification should meet the following requirements:

(i) identification should be made in the presence of normal operating

signals and noise disturbances,

(ii) any test performed on the process must not unduly distrub the normal operation.

These conditions, in effect, do not allow a direct application of an external signal or removal of the plant during the test. Another consideration which demands attention is that identification must be made quickly if it is to have any use in the adaptive control system. By their very nature, adaptive systems demand quick solution of the identification problem and it is seldom possible to allow long time intervals merely for the process dynamics to be identified.

2.5 ADAPTIVE CONTROL WITHOUT IDENTIFICATION:

Because of the numerous difficulties mentioned above, another approach [11], [12] has become popular during the recent years. This approach does not demand the identification of plant dynamics and the system can be made adaptive on the principle of performance measure, wherein the need for controller adjustment is detected through a measurement of the system performance. A very good example of an adaptive system without identification is a pilot flying an airplane. The pilot is constantly identifying the plane's behaviour without doing so in terms of coefficients of a differential equation or transfer function. What the pilot senses is simply the current performance of his plane along with its current sensitivity to the control action to adjust the current performance to the desired performance. Because the human pilot is an excellent adaptive system (within his limitations and speed), the

research efforts have been directed to lie along lines characterising his operation. This gives rise to three basic constituents of the adaptive principle:

(i) The definition of an optimum condition of operation or a figure of merit.

(ii) The comparison of the actual performance with the desired performance.

(iii) The adjustment of the system in order to drive the actual performance towards the desired performance.

The first quantity is the designer's decision, and the remaining two are automatic operations to be accomplished by the system.

This approach also overcomes the many imperfections of the conventional design philosophy. Present conventional procedures for automatic control systems are, in the language of a psychologist, completely structured. That is the system can only cope with problems foreseen and allowed for by the designer. Also, it is almost essential to have the controlled variable directly available for measurement and manipulation in order to have the desired control. In adaptive control systems, the designer provides the system with a means of continuously monitoring its own performance in relation to a given figure of merit and a means of adjusting the variable parameters so as to reach the desired optimum, rather than organising the system to meet anticipated inputs and parameter variations.

2.6 THE FIGURE OF MERIT OR PERFORMANCE CRITERION:

Perhaps the most important aspect of the overall performance of

adaptive control is the figure of merit against which a system measures its performance. This choice is made by the designer and consequently the final response of the system can be no better than the criterion. The performance of a system is generally a function of stability, sensitivity, accuracy and transient response, etc. The exact specifications are dictated by the required system performance. Certain characteristics are more important in some systems than the other. Transient response is by far the most important attribute of all the physical systems and designers of such systems are often faced with the problem of optimizing the transient behaviour. Various criteria have been put forward in the past decade but unfortunately most of them are either impractical, incomplete or designed to solve a very specific problem. Transient response characteristics are usually defined on the basis of a step input. Graham and Lathrop [14] have developed several performance criteria for optimizing the transient response. These criteria resolve the conflict that exists between rise time, peak overshoot and settling time. Of the different criteria discussed in [14], integral of time-multiplied absolute-value of error $\int_0^\infty |e| t dt$ (ITAE) and integral of time multiplied square of error $\int_0^\infty e^2 t \, dt$ (ITSE) appear to be most suitable. ITAE has been extensively used and yields good results because of its selectivity and ease of mechanisation on an analog computer. ITSE appears to be equally reliable, though it has not been employed as much as ITAE.

While ITAE and ITSE are fairly reasonable performance criteria on any objective and operational basis, they require considerable computation. Also it is to be noted that these performance criteria appear to be insensitive over a wide variety of applications--nevertheless it

is not always so. In any given system there is always the possibility of improving its performance by the measurement of performance criterion and systematic adjustment of the adjustable parameters.

Use of integral criterion as a figure of merit in adaptive systems has another important drawback. The performance criteria like ITAE or ITSE can affect the stability of the system. This follows from the fact that in real system it would not be possible to measure the performance ITSE ($\int_0^\infty e^2\ t\ dt$) because this requires integration over an infinite interval of time. A real measurement would be a truncated form of ITSE where the upper limit of the integral would be T. T is taken to be reasonably large--about four to five times the largest time constant present in the system. Also, almost all the optimization techniques require the determination of partial derivatives of the performance criterion with respect to adjustable parameters. The adjustment of the controller parameters depends on the partial derivatives and therefore there is a definite time lag of T units in the adjustment for each variable parameter (since partial derivatives can only be determined if the value of performance criterion is known). This can cause some instability if the plant parameters have changed in the mean time. This will be dealt with in more detail in Chapter 3.

2.7 RESPONSE SURFACE:

The problem of maintaining the system response at an optimum is one of adjusting a set of controller parameters. If there are more than one parameter to be adjusted as is the usual case, the problem of

optimization becomes n dimensional. Each of the parameters to be adjusted may be thought of as defining a dimension in an n-fold optimizing space. Contours of constant performance criterion or figure of merit exist in this space. Such a presentation for two dimensional case is shown in Figure (4). A point which represents the setting of the adjustable parameters is to be moved across these contours towards the optimum, which is the minimum value of figure of merit. The nature of the response surface plays an important part in the method employed to obtain the optimum. A particular method may not work properly if the response surface is not regular. Therefore, it is not wrong to say that half the battle is won if a proper figure of merit is chosen which gives a regular response surface. Ideally the best figure of merit is one which gives the instantaneous value of the performance measure rather than the present integral criterion, but we are not aware of such a criterion at present.

The response surface can have more than one minimum and the method of optimization may lead to a local minimum which is not the true minimum. There is no definite solution to this problem at the present stage. The optimum obtained in a particular case depends very much on the starting point. The problem of finding the true minimum if the response surface is not unimodal is being studied by the people working in the field of adaptive control and is not undertaken in the present work.

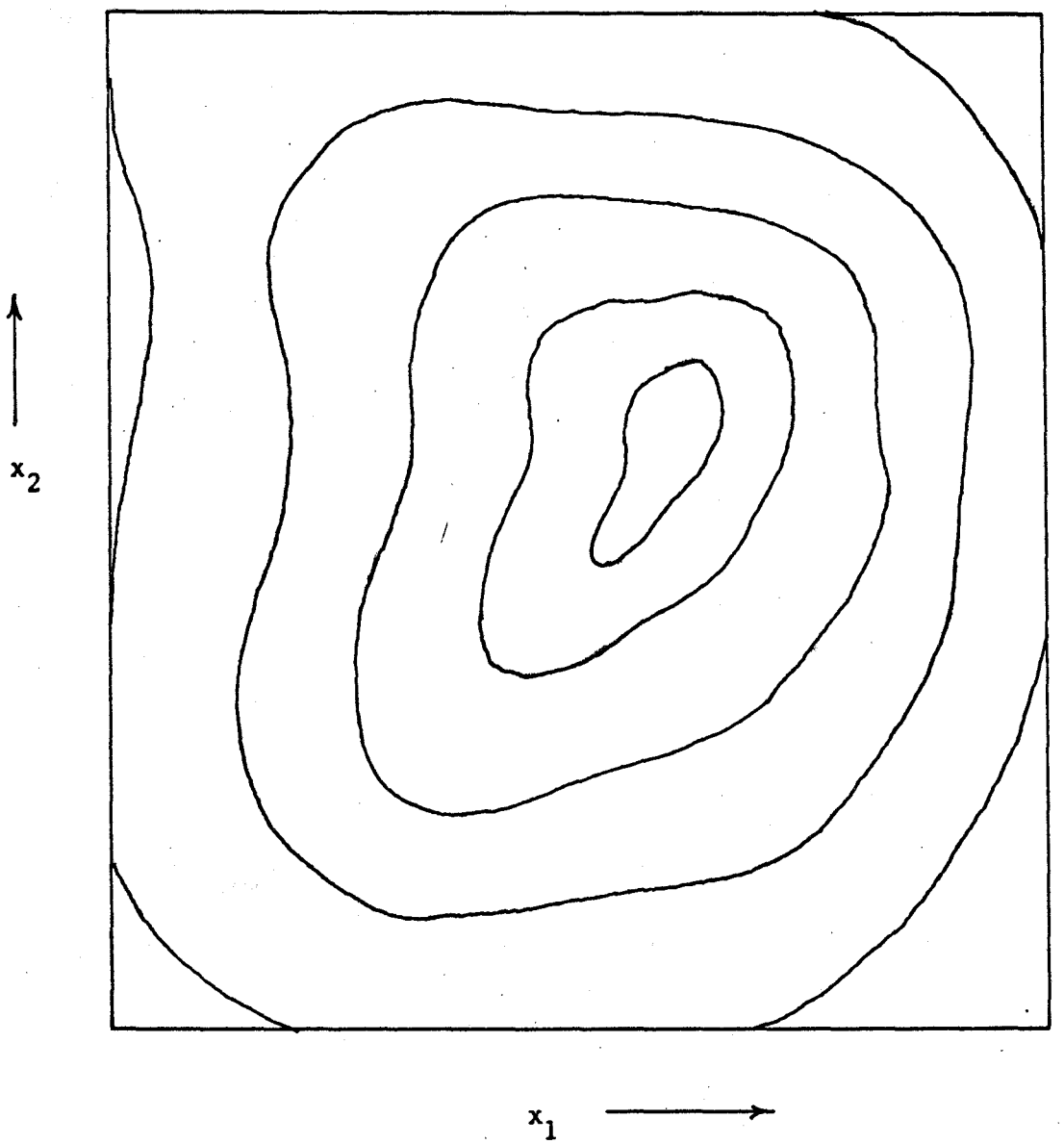


FIGURE 4: Response surface showing constant FM contours

x_1 and x_2 are the adjustable parameters

CHAPTER 3

OPTIMIZATION TECHNIQUES

3.1 INTRODUCTION:

A problem which arises in many ways is to find the maximum or minimum value of a function. The most obvious application in control engineering is in adjusting the parameters of a feedback system so as to minimize some measure of performance of the system: this is the problem of optimizing. Optimization occupies an important place in the practical world of engineering, trade and even commerce.

The optimization problem can arise on a plant which is not as yet built, where the design of the plant and the controller can be varied easily to suit the desired control. This problem can equally well arise on a plant which is built and already in operation. It is this class of problems that we are mainly concerned with, where the design of the plant cannot be changed and access is only to the controller parameters, which can be adjusted to make the system work in an optimum fashion.

3.2 REVIEW OF CLASSICAL METHODS FOR OPTIMIZATION:

Until World War 2, the only mathematical methods for handling the optimization problem were classical differential and variational calculus. These include the well known theory of maxima and minima and

calculus of variations.

The theory of maxima and minima is concerned with the problem of finding the values of each of the independent variables x_1, x_2,------, x_n at which some specified function of the n-variables $F(x_1, x_2, \text{--------}, x_n)$ reaches either a maximum or a minimum (extremum). This problem may be interpreted geometrically as the problem of finding a point in an n-dimensional space at which the desired function has an extremum. This geometrical interpretation is quite helpful in understanding the problem, particularly when there are only two independent variables. A representation of such a problem is shown in Figure (4). The independent variables are x_1 and x_2 while the dependent variable $F(x_1, x_2)$ is represented by the contour lines.

A detailed treatment is given by Leitman [15]; only the limitations of this approach will be pointed out here. In its simplest form it is valid only if there are no constraints or, if there are, these can be eliminated by substitution into the optimizing equation.

The calculus of variations is concerned with optimization problem under more general conditions than those considered in the theory of maxima and minima. This method is employed when the optimum value of a function F is determined, not by selecting proper values for a series of variables, but rather by finding the proper form of a function which yields the optimum value of F.

The three fundamental problems in the calculus of variations are the Lagrange, Mayer and Bolza problems. It is easily possible to transform a problem of one type into either of the other two, and many optimum control problems can be formulated as one of these three fundamental

problems. Quite a few books on control theory treat these problems in detail [15], [16], [17], [18].

Finding the optimum of a function by the calculus of variations generally leads to a two-point boundary value problem. Analytical solutions for such problems are only possible in special cases. The resulting Euler-Lagrange differential equations are usually non-linear and numerical methods are applied to obtain the approximate solution.

The difficulty in solving cases with the constraints makes the calculus of variations approach less attractive in finding the optimum, but it is still employed in applied aerodynamics and flight mechanics. Two analytical techniques have been developed during recent years for solving problems posed in terms of calculus of variations. One of these is Bellman's Dynamic Programming [19] and the other is Pontryagin's Maximum Principle [20]. Their application to control problems can be found in almost all advanced texts on control theory.

3.3 LINEAR PROGRAMMING:

We have examined the classical methods for optimization and seen their limitations. An alternative approach, and one developed rather recently, is to use a numerical iterative method. This has been most successful in the case of linear systems, where it is called Linear Programming.

A problem in linear programming consists of the determination of a set of parameter values (a program), subject to given linear constraints, and having the property of being optimal in the sense that a given linear value function has the extremum value. The linear programming is

applicable to well defined, linear processes and to processes which are essentially static in nature.

3.4 HILL CLIMBING:

In adaptive control systems where identification is not convenient or desirable, we resort to techniques which optimize some suitable function of the system such as a performance criterion. Hill climbing techniques, as they are popularly known, are in general applicable to any function where a number of variables can be adjusted to give it a maximum or minimum value. Figure (4) can be interpreted as representing a hill with the largest value of the figure of merit corresponding to the hill top. These arguments also apply to the bottom of a valley.

The simplest way which suggests itself for solving a hill climbing problem is to vary each parameter in turn. For example, if we are seeking a minimum of $F(x_1, x_2)$, we might vary x_1, as shown in Figure (5) until no further improvement is obtained. Then we vary x_2 along bc until the minimum on this line is obtained. Again we vary x_1, and in this way we reach the minimum along the path abcde. When there are more than two parameters to be adjusted, a minor modification in the method described is used. Instead of varying the parameters as described in the order $x_1, x_2, \text{----}, x_p, x_1, x_2, \text{----}$, we evaluate every $\partial F/\partial x_i$ and choose to vary that x_i which gives the largest derivative. This method is also used in Southwell's relaxation method.

The example in Figure (5) is ideal because the contours are nearly circular. It may be pointed out here that almost any hill climbing method will work properly when the contours are roughly

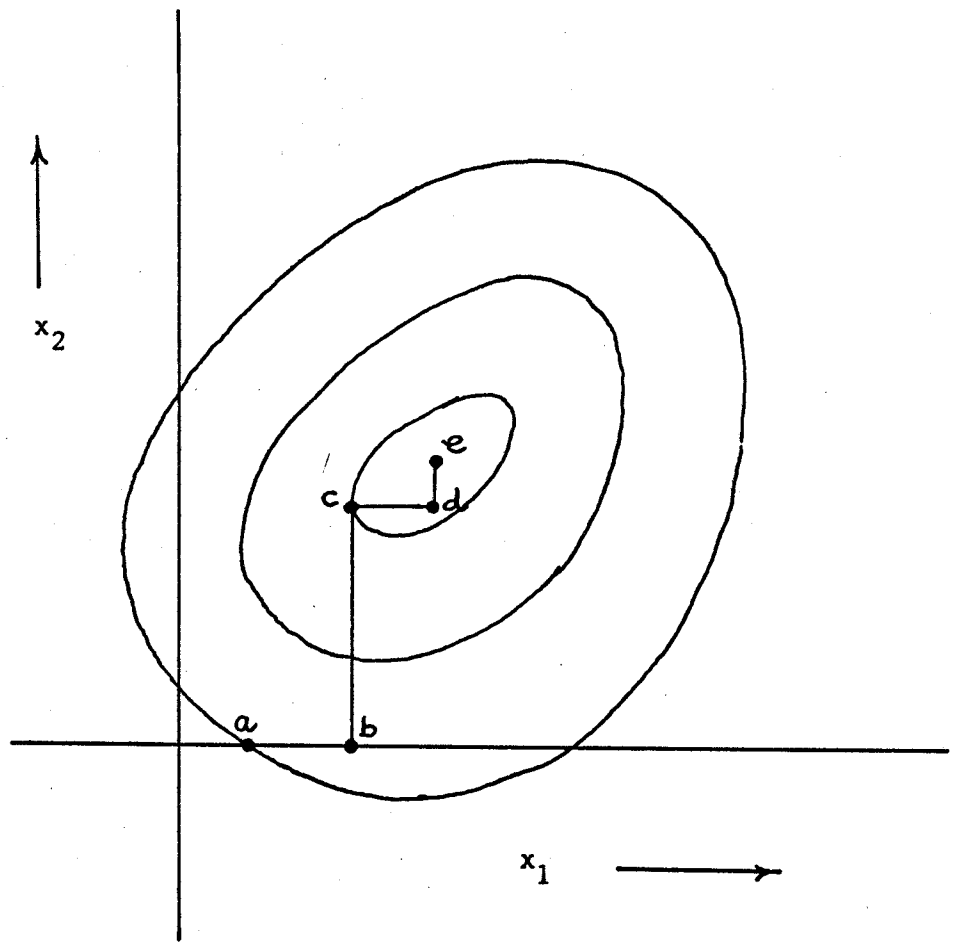


FIGURE 5: Basic hill climbing on circular contours

circular, and there is not much difference in efficiency between most methods in these circumstances. A more probable response surface is shown in Figure (6).

The contours are elliptical and the minimum is located at "m". The procedure described above leads to the path abcdefgh_____, and evidently the progress is slow.

3.5 METHOD OF STEEPEST DESCENT OR GRADIENT METHOD:

A better method to find the minimum than that mentioned above is the method of steepest descent or gradient method. The basic concept is quite simple and dates back to Cauchy. If the function to be optimized is known, we can, in two dimensional geometric terms, simply follow the direction of the steepest slope until the minimum is reached. To follow this direction, we tend to move normal to the contour lines. What is done mathematically is that the partial derivatives of the function $F(x_1, x_2)$ are first found. Then the direction of the steepest descent (which is the direction defined by the gradient vector of F) is found by computing the unit vector U with components

$$U_i = \frac{\partial F}{\partial x_i} \left\{ \sum_{i=1}^{2} \left(\frac{\partial F}{\partial x_i} \right)^2 \right\}^{-\frac{1}{2}} \qquad \text{.....(3.1)}$$

This vector U is normal to the local contour surface. Having determined the direction of steepest descent at an arbitrary starting point, a step of some length e is taken. The direction of the steepest descent is again worked out and another step of length e is taken and so on. If e is small and constant,a continuous path of steepest descent joining the

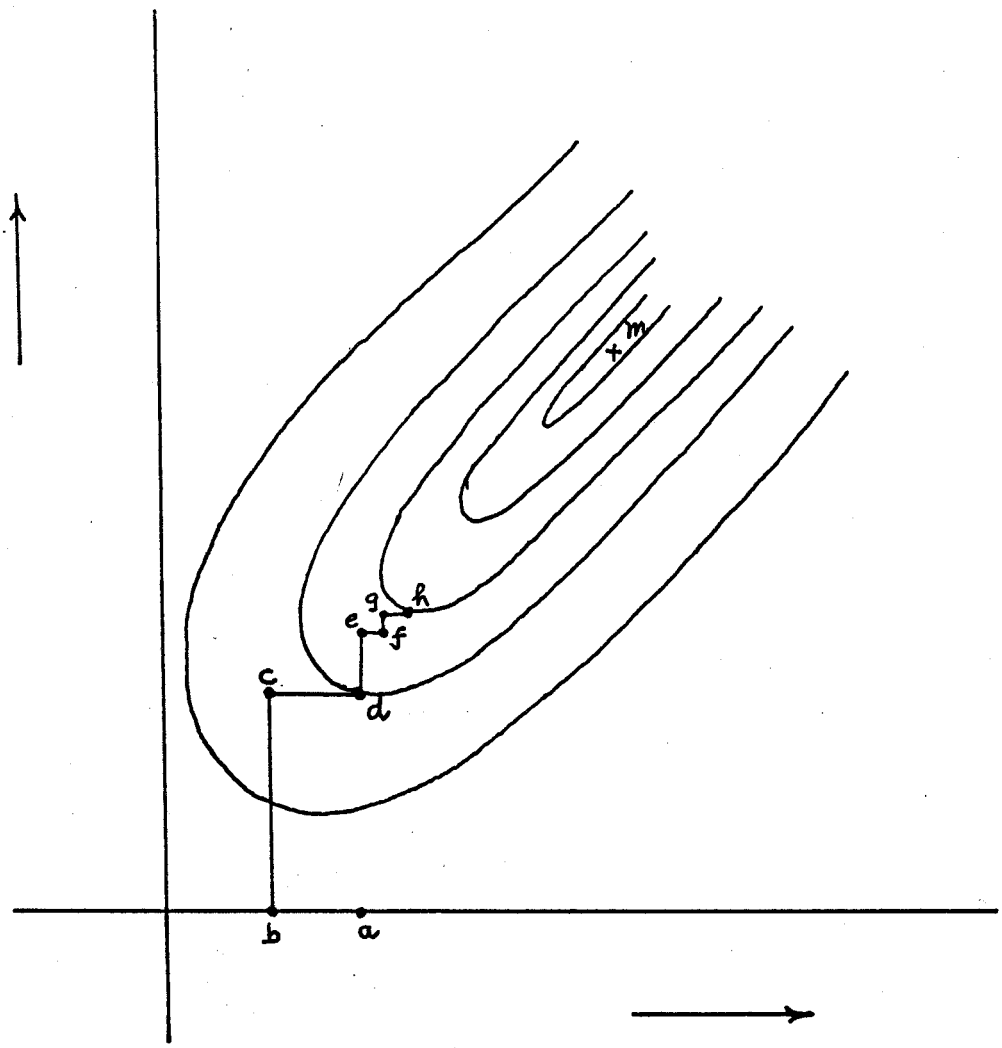


FIGURE 6: Basic hill climbing on elliptical contours

starting point to the minimum point, results. This procedure has obvious limitations and a discrete version of steepest descent is resorted to in practical cases. In such cases like adaptive control systems, the function to be optimized is not known analytically and the value of this function cannot be determined immediately after a parameter change has been made. The discrete version of steepest descent is best illustrated by considering an example. The function to be optimized is taken as ITSE, then:

$$F(x_1, x_2) = \int_0^T e^2 (x_1, x_2)\, t\, dt \qquad .....(3.2)$$

To find the value of F at any instant we have to evaluate the integral from t = o to t = T. The partial derivatives of F cannot be determined analytically, so they are obtained numerically as follows: First x_1 is varied by an amount dx_1 and $\partial F/\partial x_1$ is estimated from:

$$\frac{\partial F}{\partial x_1} = \frac{F(x_1 + dx_1, x_2) - F(x_1, x_2)}{dx_1} \qquad .....(3.3)$$

The other derivative is estimated in the same way. Evidently each partial derivative requires the same length of time T to evaluate, because the value of $F(x_1 + dx_1, x_2)$ is not known immediately after the change in parameter x_1 is made. After the estimation of partial derivatives, the direction of steepest descent or gradient is found by (3.1).

Steps are then taken from an initial point in the direction of gradient, assuming that gradient does not change, until the lowest point on this line is found. The direction of the gradient is again computed and steps are taken from this point in the new direction. This results

in a discrete approximation of steepest descent as shown in Figure (7). It can be seen from Figure (7) that progress is made by successive steps in two directions. These directions are fixed by the choice of starting point "a" and may not bear any relation to the direction of any ridges present in the response surface. If the starting point happens to be on the side of the ridge, the method may not progress well. It may be pointed out here that in case of two parameters, the method just described is essentially equivalent to the method of varying each parameter in turn. When there are more than two parameters, successive directions of steepest descent are normal to each other, but n successive directions are not necessarily mutually orthogonal. Thus, when there are more than two parameters, the method is not equivalent to the method of varying one parameter at a time.

Thus far, we have not included constraints. The presence of constraints is a difficult complication. The inclusion of constraints is treated in detail in reference [17]. There are many modifications to the method of steepest descent; the two most commonly employed are due to Booth [21] and Rosenbrock [22]. These modifications usually give some improvement but difficulties still persist.

The method of gradients is usually employed in most optimization problems. As pointed out before, this requires knowledge of partial derivatives which are impractical for analytical determination. They are more commonly determined by the use of equation (3.3). This numerical evaluation is not as simple as it might appear.

The choice of dx_1 in equation (3.3) must be a compromise between two conflicting requirements. If dx_1 is too large, equation

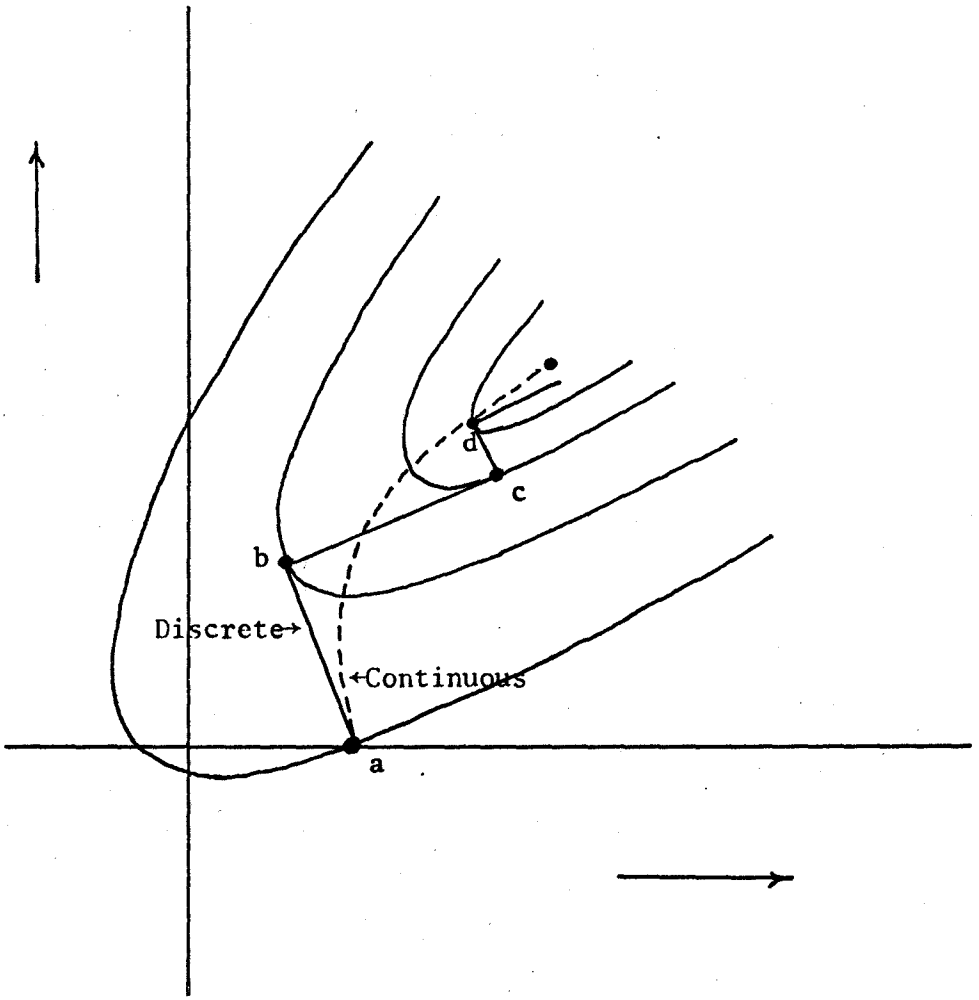


FIGURE 7: Continuous and discrete steepest descent paths

(3.3) will not give a correct estimate of the partial derivative; on the other hand if it is too small, the derivative will be almost zero. The magnitude of dx_1 taken depends on the ingenuity of the designer, however, it is usual to take it as 1% of the parameter in most applications [7].

Narendra and Streeter [12] have suggested an alternative approach for calculating the partial derivatives using correlation techniques, when the unidentified plant is subject to a random input. This technique shows promise, but enough work has not been done yet employing it.

It has been practically demonstrated that some optimization techniques, which are otherwise good, may not work as efficiently as a particular method for a particular problem. Even with hill climbing techniques, some modifications work better than others. This is because of the fact that most modifications have been found by emperical experiment, and theoretical guidance on the fundamental problem is lacking, apart from one or two elementary results [19]. There has been no general method which will work reliably in the majority of circumstances.

Most of the work in optimization has been done on a theoretical basis, optimizing functions which are seldom encountered in practice. In adaptive control systems, we optimize functions like ITSE and ITAE, etc., but very little work has been done in this area. The next two Chapters are devoted to the investigation of two recently developed methods, namely:

(i) Method of conjugate gradients.

(ii) Method of parallel tangents.

It is claimed that these methods are more or less general for most applications. These are applied to optimize ITSE criterion in an adaptive system and to a highly realistic theoretical function. In both the cases, the optimization is carried out in two dimensions.

CHAPTER 4

METHOD OF CONJUGATE GRADIENTS

4.1 INTRODUCTION:

The problem of finding an optimum of the performance criterion F constitutes a major problem in the design of adaptive control systems. Where the plant identification is not convenient, the function F is an observable response depending on variables x_1, x_2, ----, x_n under the control of the designer. As mentioned in Chapter 3, the derivatives in such cases are determined numerically and iterative methods are employed in which one starts with a trial solution and obtains successive improvements. In the past few years several promising minimization techniques have been developed. Important among these are Fletcher and Reeves [23] adaptation of the conjugate gradient method of Hestenes and Stiefel [24] and the parallel tangent method of Buehler, Shah and Kempthorne [25].

The method of conjugate gradients is based on an elegant n-step iterative procedure for solving a set of linear equations:

$$Ax = K \qquad \text{.....(4.1)}$$

where A is an n x n symmetric positive definite matrix of coefficients, x is an n x 1 vector of unknowns and K is an n x 1 vector of constants. Starting from a trial solution, an algorithm is applied to give successive approximations to the solution and, if the computations are carried

out with complete accuracy, a solution is obtained (where it exists) after m iterations where m < n (the order of the system).

The method of conjugate gradients is a special case of the method of conjugate directions. A complete treatment of the method of conjugate gradients is given by Hestenes and Stiefel and by Beckman [26]. A brief account is given here.

It is assumed that a solution vector "h" of the system Ax = K exists. Let us suppose that a set of n "A-conjugate", or "A-orthogonal" vectors p_i, i = 0, 1, ____, n - 1, is available. This makes the inner product $\langle A\, p_i, p_j \rangle = 0$ where $i \neq j$. If A is positive definite, then $\langle A\, p_i, p_i \rangle > 0$. In this case, since the p_i are necessarily independent and span the n-dimensional space, the solution vector h can be written as:

$$h = c_o p_o + c_1 p_1 + \text{-------} + c_{n-1}\, p_{n-1}$$

The solution is readily known if c_i are known. c_i can be determined as:

$$\langle Ah, p_i \rangle = \langle K, p_i \rangle = c_i \langle Ap_i, p_i \rangle$$

which gives $$c_i = \frac{\langle K, p_i \rangle}{\langle Ap_i, p_i \rangle}$$

and $$h = \frac{\langle K, p_o \rangle}{\langle Ap_o, p_o \rangle} p_o + \frac{\langle K, p_1 \rangle}{\langle Ap_1, p_1 \rangle} p_1 + \text{----------}$$

$$\text{--------} + \frac{\langle K, p_{n-1} \rangle}{\langle Ap_{n-1}, p_{n-1} \rangle} p_{n-1}$$

In the method of conjugate gradients a particular set of "A conjugate" vectors p_i is developed and a solution found in terms of these. The same arguments apply to the minimization of a positive definite quadratic function of n variables in n steps. The method of conjugate gradients is a modification of the classical method of steepest descent which tells us that starting from a point x_o on the surface of an ellipsoid $F(x)$ = constant in n-dimensional space, the greatest instantaneous reduction in $F(x_o)$ is achieved by travelling from x_o in the direction of the gradient. In the method of conjugate gradients further improvement is introduced by generating directions p_o, p_1,-----, such that p_{i+1} is a linear combination of $-g_{i+1}$ (gradient vector) and p_o, p_1,-----, p_i so that the A-orthogonality condition $\langle Ap_i, p_j \rangle = 0$ for $i \neq j$ is satisfied.

4.2 GENERAL ALGORITHM OF CONJUGATE GRADIENT METHOD:

The above treatment leads to the following algorithm. Let x_o be an arbitrary starting point to the solution. Then the following formulas define the fundamental conjugate gradient iterative procedure.

$$
\left.
\begin{aligned}
x_o &= \text{arbitrary} \\
g_o &= g(x_o),\ g \text{ represents the gradient.} \\
p_o &= -g_o \\
x_{i+1} &= \text{position of the minimum of } F(x) \text{ on the} \\
&\quad \text{line through } x_i \text{ in the direction of } p_i \\
g_{i+1} &= g(x_{i+1}) \\
B_i &= \frac{g_{i+1}^2}{g_i^2} \\
p_{i+1} &= -g_{i+1} + \beta_i p_i
\end{aligned}
\right\} \qquad (4.2)
$$

It has been claimed [23] that this procedure locates the minimum of a positive definite quadratic function of n variables in at most n iterations (apart from rounding off errors etc.). For other functions like performance criterion of control system etc., which may not be quadratic and linear, this procedure is iterative rather than n step. It is clear from the procedure (4.2) that setting $\beta_i = 0$ results in the method of steepest descent. In applying the procedure (4.2) to general functions, the following points demand consideration.

(i) The Choice of Starting Point x_o:

For quadratic functions any choice of starting point is in principle equally satisfactory. But is is not true for general functions. For practical purposes, that starting point would be considered best which leads to the minimum as quickly as possible.

(ii) The Line Search:

Starting from x_i, the approximation x_{i+1} is obtained by travelling a certain distance along the vector p_i. The distance travelled is such as to minimize the function $F(x)$. To find the best point on this line, different fits have been put forwards. Buehler, Shah and Kempthorne [27] have used quadratic interpolation whereas Fletcher and Reeves suggest cubic fit. We have tried linear interpolation for simplicity. There is no doubt to the fact that better results and lesser iterations are obtained using quadratic or cubic interpolation.

(iii) The Convergence Criterion:

The iterations should stop if any of g_i is zero, because at the minimum all g_i should vanish. This, however, does not happen

because of round off errors. The more appropriate criterion is that iterations should stop when there is no considerable reduction in the value of the function. The convergence criterion depends on the choice of the designer and in particular cases it may well be possible to use some less stringent criterion.

These points are treated in detail by Fletcher and Reeves [23].

4.3 METHOD OF CONJUGATE GRADIENT APPLIED TO BEALE'S FUNCTION:

In order to investigate the performance of the method of conjugate gradients in case of general functions, a two dimensional example given by:

$$F(x_1, x_2) = \sum_{i=1}^{3} u_i^2$$

$$u_i = c_i - x_1(1 - x_2^{\,i})$$

$$(c_1, c_2, c_3) = (1.5, 2.25, 2.625)$$

is selected. This function is non-quadratic and non-linear and was selected by Beale [27] as a particularly awkward one. It has a minimum of zero at $(x_1, x_2) = (3, 0.5)$. Some contours are shown in Figure (8) The choice of scales in Figure (9)is suitable for graphical representation and yields better results since steepest descent, and its modifications like conjugate gradients are scale dependent. It has been observed that better performance results when the choice of scales is such that change in one variable is of the same order as that in the other.

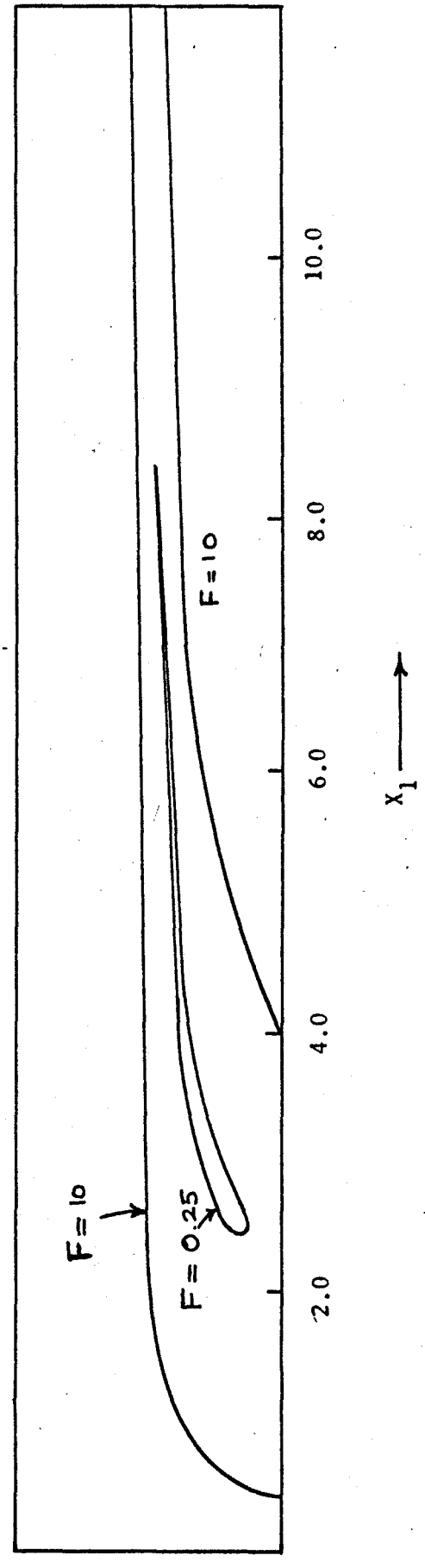


FIGURE 8: Contours of Beale's Function

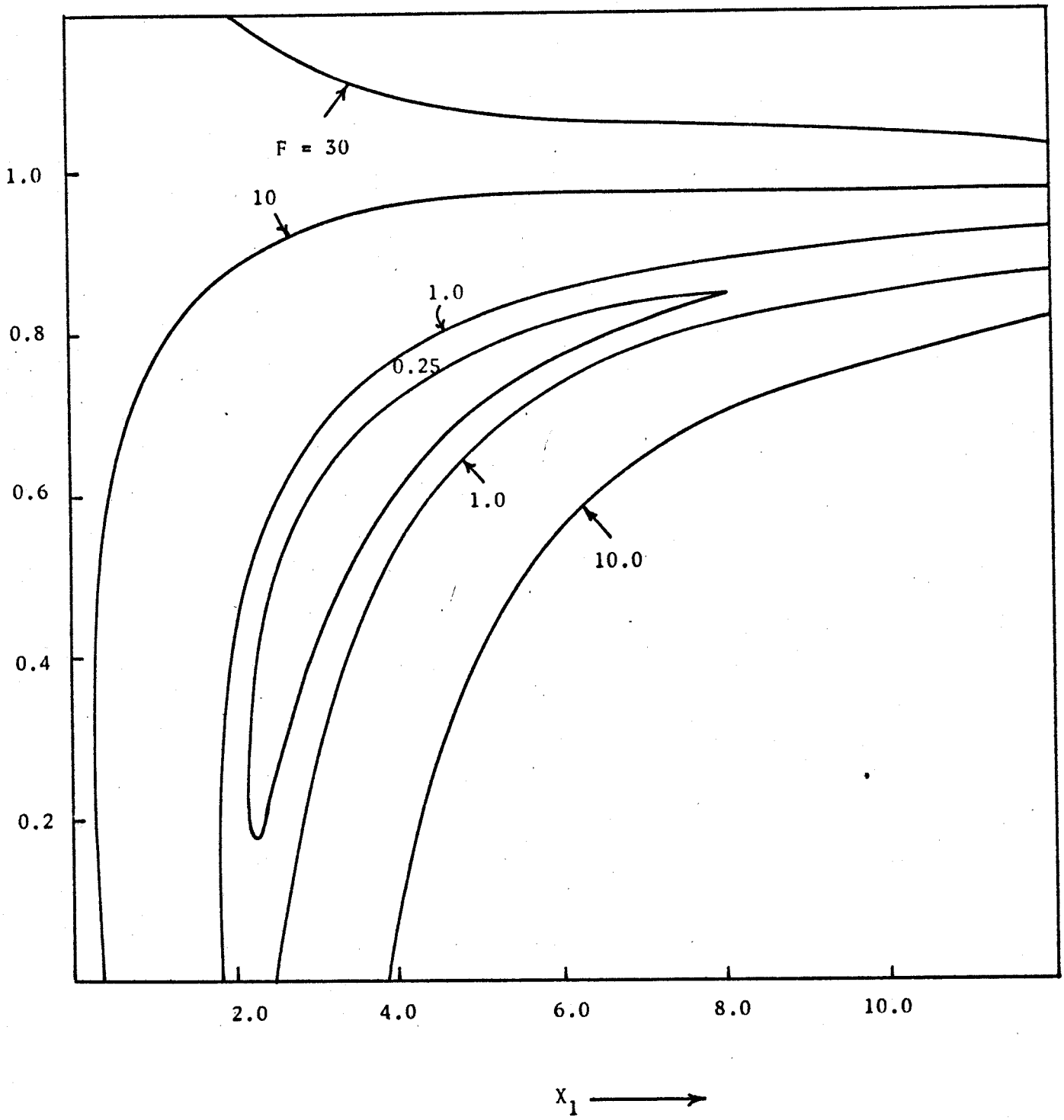


FIGURE 9: Contours of Beale's Function

The choice of scales shown here is used in numerical example.

Four starting points are tried with choice of scales as in Figure (9). For each starting point, the method of steepest descent is also applied in addition to conjugate gradient method. The results are shown in Tables 4.1-8. The calculations were carried out on an IBM 7040 using WATFOR compiler which is limited in the memory to half that of an IBM compiler but is ten times faster than the IBM compiler. Typical running time for 30 iterations was 40 to 60 seconds.

The computational details are similar to those employed by Buehler, Shah and Kempthorne [26] and are given below:

First iteration from any starting point $x_o(x_1, x_2)$ is achieved by travelling in the direction p_o along a line with a prescribed slope m_o, where $p_o = -m_o$. This line search involves the following steps:

Find $F(x_o)$

Change x_1 by Δx_1 (the choice of Δx_1 is usually small and $\Delta x_1 = 10^{-K}$ where K = 4, 5, or 6. In the present case study K = 4) and $\Delta x_2 = m\Delta x_1$

Find F here and check $\Delta\phi<o$

Proceed along this line successively doubling the step size until $\Delta\phi>o$. Estimate the minimum point on this line by a linear interpolation to the last two F-values.

From x_1 this process is repeated until g_i approach zero or not much reduction is obtained in the value of F, but the line searches are made in the directions $p_{i+1} = -m_{i+1} = \beta_i p_i$ where $\beta_i = \frac{m_{i+1}^2}{m_i^2}$. For steepest descent the method is identical to the one above with $\beta_i = o$. The slopes in this case study were determined analytically as $m_i = \frac{\partial F/\partial x_2}{\partial F/\partial x_1}$.

The flow chart and some computer programs are given in Appendix .

4.4 METHOD OF CONJUGATE GRADIENTS APPLIED TO A CONTROL SYSTEM:

To further study the behaviour of the method of conjugate gradients on practical functions, a system suggested by Gibson [28] is tried. The system is shown in Figure (10). The performance criterion or figure of merit (FM) is ITSE. This performance criterion has the desirable features of realibility and selectivity like ITAE and has been recommended for use by Gibson. This system has fairly nice response surface with closed constant FM contours in the stable region of ω_1 and ω_2. Like Beale's function it is a two-dimensional problem. The performance criterion ITSE has a minimum of 0.01870 at $(\omega_1, \omega_2) = (0.36, 17.0)$.

The whole system was first tried using the MIMIC programming [29] language on IBM 7040. MIMIC is a programming system written for a digital computer which, from the standpoint of the user, seems to make the sequential machine function like an analogue computer. This programming system has a very desirable feature of eliminating time and amplitude scaling. Moreover, MIMIC has the provision for carrying out hybrid calculations. It was found after careful consideration that MIMIC processor was not working properly. The same trouble was encountered by Hinchley [30].

The system was then solved on IBM 7040 using IBM compiler. WATFOR compiler could not be used because of memory considerations. The computational procedure is similar to the one used in Beale's function with two differences:

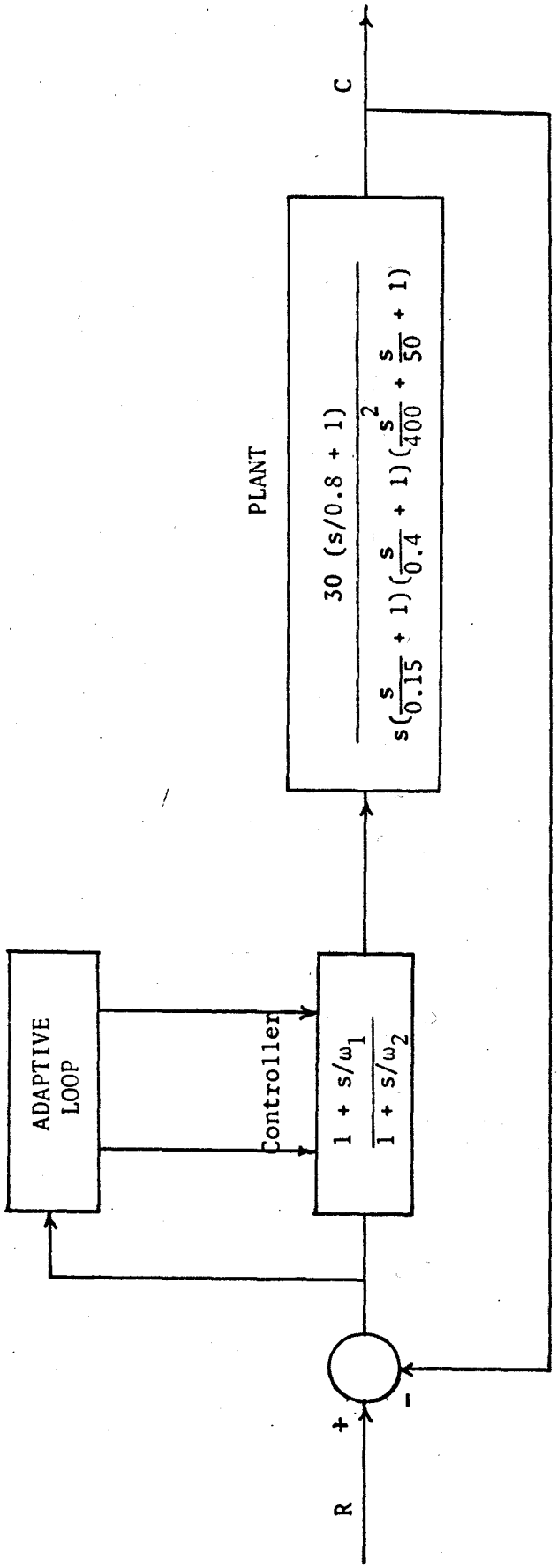


FIGURE 10: Gibson System

(i) The function $F(\omega_1, \omega_2)$ is obtained by the use of a subroutine. The overall transfer function is solved by breaking the sixth order differential equation of the system into six first order simultaneous differential equations and then applying the Runge-Kutta method of solving these.

(ii) Partial derivatives are calculated numerically by the local examination of the response surface using equation (3.3). The numerical evaluation of the partial derivatives demands careful consideration and their evaluation is carried out on the factors discussed in Chapter 3.

Four starting points are tried in this case, also, two with the method of steepest descent applied in addition to the method of conjugate gradient. The results are shown in Tables 4.9-16. From the results it can be seen that the method of steepest descent requires a large number of iterations both in the case of Beale's function and Gibson's system. The rate of convergence is slow and the number of iterations required in a particular case depends very much on the starting point. The method of conjugate gradients seems to require more iterations in some cases than the steepest descent, and it can be seen that no marked improvement is obtained with this method. This is further illustrated graphically in Chapter 6 by Figures (13-21).

TABLE 4.1

METHOD OF STEEPEST DESCENT

BEALE'S FUNCTION

(Starting Point $x_1 = 5.0$, $x_2 = 0.2$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	5.0	0.2	18.20472
2	3.36170	0.66020	0.30613
3	3.46400	0.58040	0.03054
4	3.43850	0.59108	0.01968
5	3.41299	0.59090	0.01861
6	3.41149	0.58618	0.01777
7	3.38599	0.58656	0.01689
8	3.38550	0.58084	0.01601
9	3.36640	0.02543	0.01535
10	3.36489	0.57790	0.01412
11	3.33939	0.57807	0.01379
12	3.13472	0.53822	0.00373
13	3.14102	0.53315	0.00273
14	3.12832	0.53280	0.00248
15	3.12800	0.52936	0.00220
16	3.11852	0.53027	0.00212
17	3.11782	0.52774	0.00195
18	3.10512	0.52792	0.00181
19	3.10662	0.52470	0.00163
20	3.10032	0.52526	0.00150
21	3.09402	0.52081	0.00134

TABLE 4.2

STEEPEST DESCENT

BEALE'S FUNCTION

(Starting Point $x_1 = 2.0$, $x_2 = 0.2$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	2.0	0.2	0.52978
2	2.40950	0.18977	0.26361
3	2.30720	0.24416	0.18940
4	2.51189	0.27235	0.13998
5	2.46080	0.30120	0.10194
6	2.87030	0.48218	0.00825
7	2.89580	0.47478	0.00201
8	2.90850	0.47326	0.00169
9	2.90840	0.47670	0.00149
10	2.91789	0.47640	0.00131
11	2.91859	0.47938	0.00116
12	2.92809	0.47907	0.00103
13	2.92760	0.48154	0.00091
14	2.93389	0.48150	0.00081
15	2.93618	0.48444	0.00070
16	2.94250	0.48384	0.00061
17	2.94359	0.48626	0.00055
18	2.94989	0.48567	0.00048
19	2.94960	0.48730	0.00042

TABLE 4.3

STEEPEST DESCENT

BEALE'S FUNCTION

(Starting Point $x_1 = 5.0$, $x_2 = 0.8$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	5.0	0.8	0.48672
2	5.02550	0.74299	0.15853
3	5.01280	0.75840	0.13844
4	5.01330	0.75428	0.13593
5	4.98780	0.75505	0.13516
6	4.98710	0.75245	0.13434
7	4.98080	0.75319	0.13406
8	4.97770	0.75139	0.13383
9	4.97460	0.75228	0.13358
10	4.95550	0.74863	0.13301
11	4.95240	0.75048	0.13219
12	4.91410	0.75095	0.13102
13	4.91380	0.74817	0.12979
14	3.68510	0.65302	0.06874
15	3.71060	0.62739	0.04243
16	3.69790	0.63203	0.03966
17	3.49320	0.61177	0.02949
18	3.50270	0.60061	0.02444
19	3.49000	0.60202	0.02367
20	3.47090	0.59295	0.02256
21	3.45820	0.59598	0.02121
22	3.04870	0.48861	0.01314
23	3.01040	0.50295	0.000020
24	3.01059	0.50262	0.000018
25	3.00950	0.50261	0.000016

TABLE 4.4

STEEPEST DESCENT

BEALE'S FUNCTION

(Starting Point $x_1 = 2.0$, $x_2 = 0.8$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	2.0	0.8	6.27010
2	2.61430	0.27643	0.18479
3	2.46080	0.32702	0.08973
4	2.56310	0.33272	0.06951
5	2.55040	0.36922	0.05525
6	2.62701	0.36813	0.04266
7	2.63010	0.39900	0.03420
8	2.68123	0.39630	0.02642
9	2.73240	0.43999	0.01878
10	2.78350	0.43136	0.01131
11	2.78340	0.44542	0.00991
12	2.80896	0.44289	0.00788
13	2.82806	0.45887	0.00615
14	2.85356	0.45531	0.00475
15	2.85246	0.46115	0.00412
16	2.87156	0.46113	0.00359
17	2.87006	0.46695	0.00315
18	2.88276	0.46620	0.00272
19	2.88906	0.47380	0.00236
20	2.90176	0.47194	0.00187
21	2.90486	0.47612	0.00162
22	2.91436	0.47553	0.00141
23	2.91586	0.47870	0.00124
24	2.92536	0.47835	0.00110
25	2.92506	0.48168	0.00099

TABLE 4.5

CONJUGATE GRADIENTS

BEALE'S FUNCTION

(Starting Point $x_1 = 2.0$, $x_2 = 0.2$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	2.0	0.2	0.52978
2	2.40950	0.18977	0.26361
3	2.39680	0.34030	0.12780
4	2.55030	0.32672	0.07468
5	2.54999	0.37574	0.05625
6	2.65230	0.36994	0.04226
7	2.65080	0.40011	0.02942
8	2.70189	0.40040	0.02408
9	2.70284	0.41762	0.01999
10	2.75394	0.41700	0.01677
11	2.75384	0.43544	0.01303
12	2.77934	0.43373	0.01074
13	2.77784	0.44002	0.01020
14	2.80334	0.43976	0.00870
15	2.80589	0.44981	0.00760
16	2.83139	0.44826	0.00638
17	2.83138	0.46100	0.00614
18	2.85688	0.45685	0.00442
19	2.85693	0.46252	0.00385
20	2.87603	0.46235	0.00337
21	2.87560	0.46744	0.00287
22	2.88778	0.46740	0.00252
23	2.88873	0.47153	0.00225
24	2.90143	0.47110	0.00197

TABLE 4.6

CONJUGATE GRADIENTS

BEALE'S FUNCTION

(Starting Point $x_1 = 5.0$, $x_2 = 0.8$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	5.0	0.8	0.48672
2	5.02550	0.74298	0.15853
3	5.01920	0.75478	0.13628
4	4.71210	0.74328	0.12736
5	4.71199	0.72824	0.12090
6	4.70569	0.73517	0.11657
7	4.70800	0.73475	0.11655
8	4.70569	0.73312	0.11631
9	4.69299	0.73448	0.11582
10	4.69329	0.73197	0.11519
11	4.68699	0.73314	0.11512
12	4.68769	0.73288	0.11511
13	4.68299	0.73102	0.11486
14	4.58069	0.72883	0.11023
15	4.58100	0.72355	0.10780
16	4.57470	0.72469	0.10743
17	4.57540	0.72424	0.10740
18	4.56270	0.72087	0.10687
19	4.56580	0.72233	0.10675
20	4.56110	0.72356	0.10647
21	4.56180	0.72323	0.10645

TABLE 4.7

CONJUGATE GRADIENTS

BEALE'S FUNCTION

(Starting Point $x_1 = 5.0$, $x_2 = 0.2$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	5.0	0.2	18.20472
2	1.72330	0.29204	1.42454
3	2.54240	0.28219	0.13321
4	2.44009	0.31363	0.10074
5	2.54239	0.31439	0.08689
6	2.59349	0.38750	0.04321
7	2.69579	0.38698	0.03234
8	2.68309	0.40522	0.02374
9	2.72140	0.40565	0.02162
10	2.72610	0.42823	0.01677
11	2.77720	0.42785	0.01253
12	2.77650	0.43780	0.01038
13	2.80200	0.43794	0.00922
14	2.80072	0.44726	0.00800
15	2.82622	0.44718	0.00666
16	2.82602	0.45398	0.00591
17	2.85152	0.45394	0.00507
18	2.85132	0.46093	0.00419
19	2.87042	0.46091	0.00362
20	2.86787	0.46377	0.00329
21	2.88057	0.46399	0.00308
22	2.88127	0.46827	0.00259
23	2.89397	0.46837	0.00237
24	2.89427	0.47356	0.00203
25	2.91977	0.47346	0.00167

TABLE 4.8

CONJUGATE GRADIENTS

BEALE'S FUNCTION

(Starting Point $x_1 = 2.0$, $x_2 = 0.8$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	2.0	0.8	6.27010
2	2.61430	0.27643	0.18470
3	2.51199	0.32318	0.07918
4	2.49930	0.33008	0.07739
5	2.90880	0.43752	0.03055
6	3.31830	0.56714	0.01179
7	3.30560	0.57016	0.01123
8	3.30590	0.56724	0.01087
9	3.26760	0.56567	0.00992
10	3.26750	0.55864	0.00865
11	3.25480	0.55967	0.00818
12	3.25550	0.55634	0.00796
13	3.24398	0.55709	0.00750
14	3.24260	0.55443	0.00728
15	3.21710	0.55446	0.00683
16	3.21836	0.54966	0.00605
17	3.19286	0.54909	0.00557
18	3.19413	0.54451	0.00490
19	3.17503	0.54431	0.00451
20	3.17566	0.54003	0.00409
21	3.16296	0.54078	0.00382
22	3.16265	0.53712	0.00372

TABLE 4.9

STEEPEST DESCENT

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.41$, $\omega_2 = 11.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.41	11.0	0.025100
2	0.30770	11.11272	0.020940
3	0.32040	11.16241	0.020816
4	0.31410	11.21918	0.020763
5	0.33200	11.49548	0.020579
6	0.32050	11.53123	0.020436
7	0.31420	11.64940	0.020370
8	0.32700	11.70720	0.020290
9	0.32060	11.73768	0.020250
10	0.31110	12.97310	0.019804
11	0.33660	12.97902	0.019450
12	0.33030	13.01110	0.019420
13	0.32400	13.27087	0.019379
14	0.33670	13.28418	0.019300
15	0.33360	13.29429	0.019290
16	0.33209	13.30840	0.019287
17	0.35760	14.93219	0.018938
18	0.34489	14.94100	0.018840
19	0.34560	14.93831	0.018840

TABLE 4.10

STEEPEST DESCENT

(Starting Point $\omega_1 = 0.41$, $\omega_2 = 20.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.41	20.0000	0.019695
2	0.37169	20.00380	0.018940
3	0.37639	19.99561	0.018920
4	0.37230	19.99545	0.018918
5	0.37609	19.99512	0.018913

TABLE 4.11

STEEPEST DESCENT

GIBSON'S SYSTEM

(Starting point $\omega_1 = 0.31$, $\omega_2 = 20.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.31	20.0	0.026307
2	0.38670	19.97714	0.019008
3	0.37399	19.97600	0.018933
4	0.37710	19.94619	0.018928
5	0.37560	19.94452	0.018917
6	0.37549	19.94426	0.018916

TABLE 4.12

STEEPEST DESCENT

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.31$, $\omega_2 = 10.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.31	10.0	0.022460
2	0.29089	10.51047	0.022039
3	0.31639	10.54910	0.021587
4	0.30369	10.68751	0.021475
5	0.31639	10.74097	0.021320
6	0.30369	10.91876	0.021211
7	0.31640	10.95692	0.021047
8	0.30370	11.26434	0.020880
9	0.32280	11.29986	0.020675
10	0.31650	11.34355	0.020625
11	0.34200	12.08583	0.020165
12	0.32289	12.12183	0.019951
13	0.34840	12.98325	0.019617
14	0.32930	13.00912	0.019423
15	0.33880	13.14385	0.019380
16	0.33249	13.15878	0.019342
17	0.33569	13.20049	0.019341

TABLE 4.13

CONJUGATE GRADIENT

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.31$, $\omega_2 = 10.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.31	10.0	0.022460
2	0.29089	10.51047	0.022039
3	0.31200	10.54847	0.021588
4	0.30370	10.68290	0.021476
5	0.31640	10.73879	0.021323
6	0.30370	10.91204	0.021219
7	0.31640	10.94989	0.021056
8	0.30370	11.23972	0.020901
9	0.32279	11.27576	0.020701
10	0.31330	11.34033	0.020650
11	0.32599	11.42836	0.020560
12	0.31650	11.47411	0.020500
13	0.32920	11.64916	0.020368
14	0.31970	11.68465	0.020300
15	0.37079	14.21058	0.019560
16	0.34530	14.22722	0.018997
17	0.34060	14.23486	0.018989

TABLE 4.14

CONJUGATE GRADIENTS

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.31$, $\omega_2 = 20.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.31	20.0	0.026307
2	0.38670	19.97715	0.019008
3	0.37400	19.97590	0.018923
4	0.37710	19.94620	0.018910
5	0.37560	19.94450	0.018909

TABLE 4.15

CONJUGATE GRADIENTS

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.41$, $\omega_2 = 20.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.41	20.0	0.019695
2	0.37169	20.00380	0.018940
3	0.37640	19.99561	0.018922
4	0.37630	19.99545	0.018921

TABLE 4.16

CONJUGATE GRADIENTS

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.41$, $\omega_2 = 11.0$

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.41	11.0	0.025100
2	0.30769	11.11272	0.020940
3	0.32040	11.16238	0.020816
4	0.31410	11.21831	0.020763
5	0.33200	11.48106	0.020595
6	0.32050	11.51627	0.020450
7	0.31420	11.62939	0.020386
8	0.32690	11.68780	0.020311
9	0.32060	11.71774	0.020270
10	0.31110	12.54191	0.019921
11	0.33020	12.55330	0.019665
12	0.32709	12.57107	0.019650
13	0.32080	12.91814	0.019550
14	0.33499	12.93340	0.019450
15	0.33040	12.94685	0.019449

CHAPTER 5

METHOD OF PARALLEL TANGENTS

The method of parallel tangents or "PARTAN" as it is called, is another recently developed iterative method for finding the minimum of a function of several variables.

The master strategy of PARTAN is based on certain global properties of ellipsoids. It works like the method of steepest descent and it is possible to make it invariant to changes in the scale of measurements. For criterion functions with concentric ellipsoidal contours, PARTAN will find the optimum exactly after a fixed, small number of iterations. But even when the contours are not precisley elliptical, the technique has desirable features.

5.1 BASIC RESULTS:

A two-dimensional example is considered for simplicity. Suppose $F(x_1, x_2)$ has elliptical contours and P_o and P_2 are any two points in the x_1 - x_2 plane. From P_2 progress is made in the direction parallel to the tangent to the contour at P_o until an extremum of F is found. Then the tangents at P_o and P_3 will be parallel, and the centre of the system will be found at P_4, the extremal point on the line through P_o and P_3. This approach was first suggested by Finkel [25]. Minima are determined on two parallel lines.

For elliptical contours these minima occur at mid points of the chords and the theorem of parallel chords asserts that these minima are collinear with the centre of the ellipse. The approach can be extended to n-dimensions in which P_{2n-4} and P_{2n-1} have parallel tangents and the extremum of F is found at P_{2n}.

Let F now denote a function of n-variables and suppose that we are seeking a minimum. Starting from P_o, let P_1, P_2, P_3,------- denote the successive approximations, and let π_i be the hyperplane tangent to the contour of F at P_i.

5.2 PARTAN ALGORITHMS:

(i) Steepest Descent Partan:

From any point P_o proceed along a polygonal line, P_o P_2 P_3 P_4,-------, for which P_K is at the minimum of F on the extended line joining it with the preceding point. At even numbered points proceed in the direction of steepest descent (gradient descent). At odd numbered points, P_{2K+1} proceed in the direction determined by the line joining P_{2K-2} and P_{2K+1}. Steepest descent partan is in fact an n-dimensional generalization of the two dimensional procedure of two steepest descents followed by an acceleration step. This approach was suggested by Forsyth and Motzkin and is shown in Figure (11).

(ii) General Partan:

From any point P_o progress is made along a polygonal line P_o P_2 P_3 P_4------, for which P_K is at the minimum of F on the extended line joining it with the preceding point. The direction of

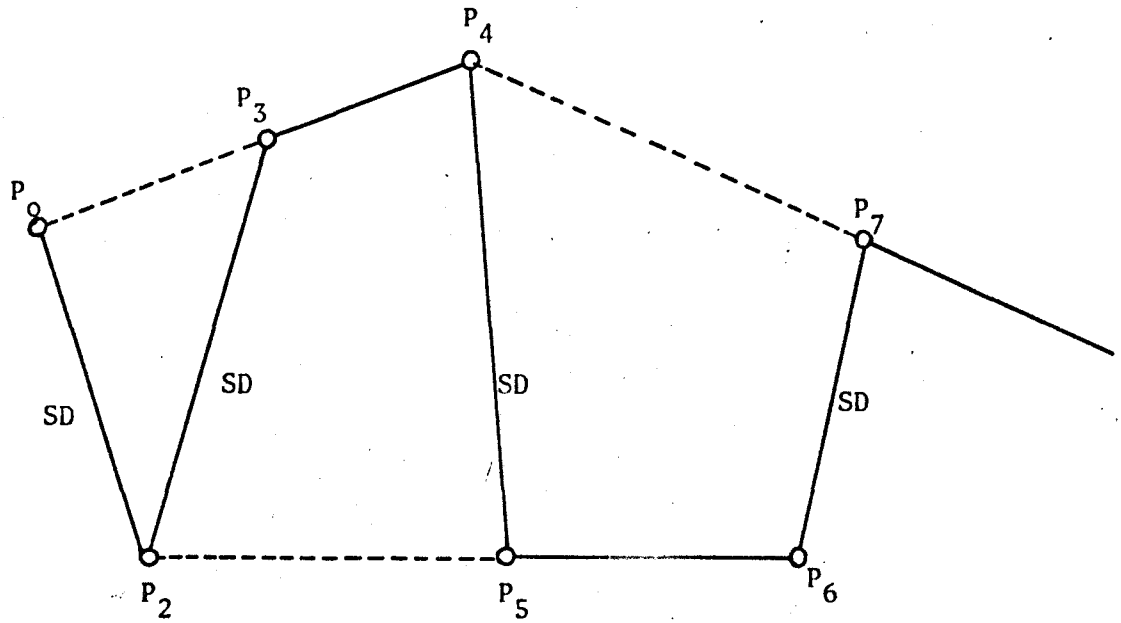


FIGURE 11: Steepest Descent Partan

SD dentoes the steepest descent path

P_o P_2 is arbitrary; P_2P_3 is any direction parallel to π_o; thereafter, for K = 1, 2, -----, n-1, P_{2K+2} is collinear with P_{2K-2} and P_{2K+1}, and for K = 2, 3, ----, n-1, P_{2K} P_{2K+1} is parallel to π_o, π_2, ----, π_{2K-2}. The procedure is represented in Figure (12).

(iii) Scale Invarient Partan:

This is a special case of the general Partan described above. In this case the arbitrary directions are resolved as follows: At P_o only x_1 is changed; at P_2, x_1 and x_2 are changed; at P_4, x_1, x_2 and x_3 are changed and so on.

An elaborate account of the method of parallel tangents is also given by Wilde [9]. Buehler, Shah and Kempthorne [25] have developed certain theorems about their method and it has been shown by them that if F is quadratic, the minimum is reached at P_{2n} or sooner, n being the number of dimensions. In non-quadratic or non-ideal cases, sufficient progress may not have been made at or before P_{2n}. One possible remedy is an iterated partan, in which iterations are started again with P_{2n} retermed as P_o. Another obvious option in the case of steepest descent partan is continued partan, in which one simply continues the alternating steepest descent and acceleration steps.

5.3 METHOD OF PARALLEL TANGENTS APPLIED TO BEALE'S FUNCTION:

Here again four starting points are tried with the same choice of scale as in the case of conjugate gradients. It has been seen and also demonstrated by Buehler, Shah and Kempthorne that the method of parallel tangents works quite satisfactorily in a particularly awkward situation with choice of scales as shown in Figure (8).

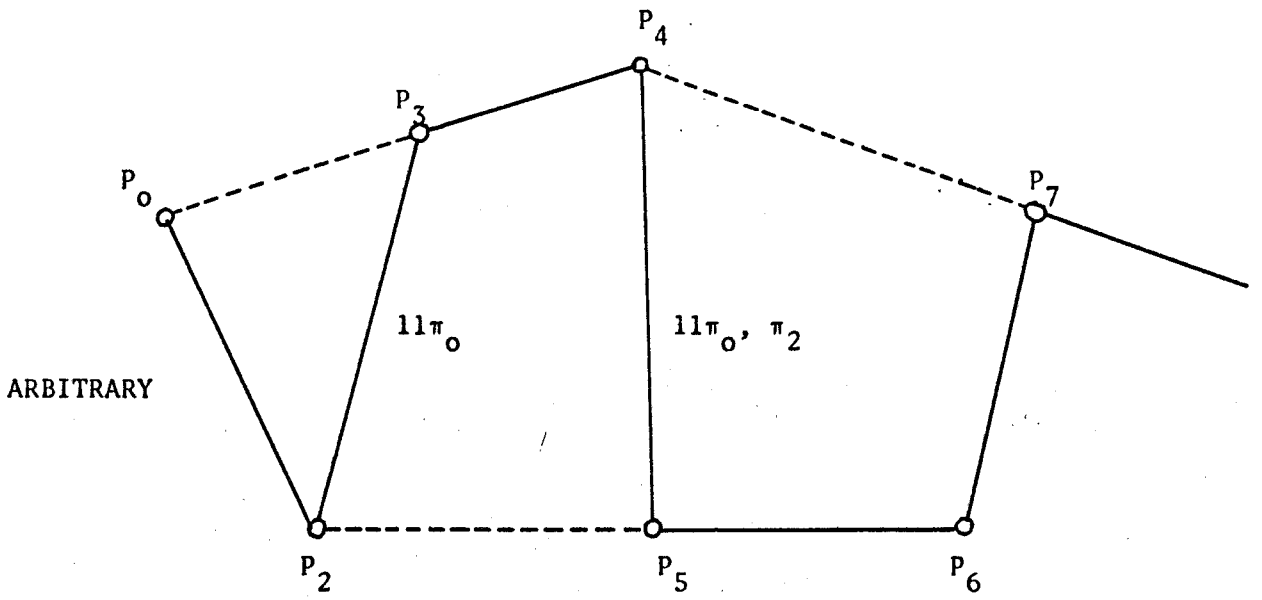


FIGURE 12: Continued Partan

π_i denotes the tangent plane at point P_i

The calculations on an IBM 7040 using WATFOR compiler do not require very large running time. Line search beginning at any point (x_1, x_2) and proceeding in a direction with prescribed slope m was accomplished as follows: Find $F(x_1, x_2)$. Change x_1 by $\Delta x_1 = 10^{-4}$ and change x_2 by $m\Delta x_1$. Find ϕ here and check $\Delta\phi<o$. Proceed along this line successively doubling the step size until $\Delta\phi>o$. Minimum point is estimated on this line by a linear interpolation rather than quadratic fit. This process is repeated until there is no appriciable change in the value of F or some other criterion is satisfied.

The step length Δx_1 is to be properly chosen as the line search depends on it. There is no set rule governing its choice, the best Δx_1 is one for which minimum is located in not many steps.

The slopes are determined from a mathematical expression. For iterated steepest descent partan, $m_o = \frac{\partial F/\partial x_2}{\partial F/\partial x_1}$. $m_2 = -\frac{1}{m_o}$ since lines $P_o P_2$ and $P_2 P_3$ are respectively tangent and the normal at P_2, and therefore perpendicular. $m_3 = \frac{x_{2,3} - x_{2,0}}{x_{1,3} - x_{1,0}}$ where second subscript denotes the point reached.

$$m_4 = -\frac{1}{m_3}$$

$$m_5 = m_3$$

$$m_6 = \frac{x_{2,6} - x_{2,4}}{x_{1,6} - x_{1,4}}$$

$$m_7 = -\frac{1}{m^6}$$

$$m_8 = m_6$$

$$m_9 = \frac{x_{2,9} - x_{2,7}}{x_{1,9} - x_{1,7}} \quad \text{. . and so on.}$$

Continued Partan steps are the same up to P_5 where m_5 is changed to $\frac{x_{2,5} - x_{2,2}}{x_{1,5} - x_{1,2}}$.

$$m_6 = -\frac{1}{m_5}$$

$$m_7 = \frac{x_{2,7} - x_{2,4}}{x_{1,7} - x_{1,4}}$$

$$m_8 = -\frac{1}{m_7}$$

$$m_9 = \frac{x_{2,9} - x_{2,6}}{x_{1,9} - x_{1,6}} \quad \text{. . . and so on.}$$

For iterated scale invariant Partan:

$$m_o = m_4 = m_7 = 0$$

m_2, m_5, m_8 equal $\frac{\partial F/\partial x_2}{\partial F/\partial x_1}$ at P_o, P_4, P_7

m_3, m_6, m_9 are the same as in iterated steepest descent Partan.

5.4 METHOD OF PARALLEL TANGENTS APPLIED TO A CONTROL SYSTEM

The different options of method of parallel tangents are applied to find the optimum value of the performance criterion ITSE in Gibson's System, shown in Figure (10). The method of parallel tangents usually requires large amounts of stored information for iterative calculations and therefore, it was solved on IBM-7040 computer utilizing IBM compiler rather than WATFOR. The computational procedures are exactly similar to those described in 5.3 except that the value of function $F(\omega_1, \omega_2)$ is

obtained by the use of the same subroutine as used in the case of the method of conjugate gradients. Also, partial derivatives are calculated numerically rather than analytically on the same line as discussed in 4.4.

The results of Beale's Function and Gibson's Function are shown in Tables 5.1-24. It can be seen that the Iterated Steepest Descent option of the Partan requires very few iterations to reach the optimum in most cases. The continued option also gives comparable results in some cases. These two variants of Partan show marked reduction in the number of iterations as compared to the methods of steepest descent and conjugate gradients. The performance of Scale Invariant Partan is, however, poorer than the other two in most cases. The overall comparison of different methods will be made graphically in Chapter 6.

TABLE 5.1

ITERATED STEEPEST DESCENT PARTAN

BEALE'S FUNCTION

(Starting Point $x_1 = 5.0$, $x_2 = 0.2$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	5.0	0.2	18.20472
2	3.36170	0.66100	0.30613
3	4.18080	0.68935	0.07830
4	4.16810	0.69011	0.07790
5	2.93939	0.48443	0.00063
6	2.94409	0.48414	0.00059
7	2.94479	0.48416	0.00057

TABLE 5.2

ITERATED STEEPEST DESCENT PARTAN

BEALE'S FUNCTION

(Starting Point $x_1 = 5.0$, $x_2 = 0.8$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	5.0	0.8	0.48672
2	5.00255	0.74298	0.15210
3	4.34720	0.71367	0.09902
4	3.03648	0.54035	0.02546
5	3.04159	0.50171	0.00197
6	2.95968	0.49087	0.00029
7	2.95905	0.49047	0.00028

TABLE 5.3

ITERATED STEEPEST DESCENT PARTAN

BEALE'S FUNCTION

(Starting Point $x_1 = 2.0$, $x_2 = 0.8$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	2.0	0.8	6.27010
2	2.81910	0.10187	0.18480
3	2.40960	0.27402	0.13282
4	2.42223	0.27239	0.13227
5	2.47340	0.27637	0.12672
6	2.43510	0.28129	0.12220
7	2.48620	0.28484	0.11700
8	2.44790	0.29035	0.11248
9	2.49899	0.29390	0.10695
10	2.50529	0.29435	0.10691

TABLE 5.4

ITERATED STEEPEST DESCENT PARTAN

BEALE'S FUNCTION

(Starting Point $x_1 = 2.0$, $x_2 = 0.2$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	2.0	0.2	0.52978
2	2.40949	0.18977	0.26361
3	2.30719	0.24416	0.18940
4	2.51190	0.27235	0.13998
5	2.46080	0.30120	0.10194
6	2.87030	0.48218	0.00825
7	2.89580	0.47478	0.00201
8	2.90850	0.47325	0.00169
9	2.90840	0.47670	0.00149
10	2.91790	0.47640	0.00131
11	2.91860	0.47940	0.00116
12	2.92809	0.47907	0.00103
13	2.92760	0.48154	0.00091
14	2.93389	0.48150	0.00081
15	2.93620	0.48444	0.00071
16	2.94250	0.48384	0.00061
17	2.94360	0.48626	0.00060

TABLE 5.5

CONTINUED PARTAN

BEALE'S FUNCTION

(Starting Point $x_1 = 2.0$, $x_2 = 0.8$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	2.0	0.8	6.27010
2	2.61430	0.27643	0.18480
3	2.40960	0.27402	0.13282
4	2.42230	0.27240	0.13228
5	2.47340	0.27637	0.12672
6	2.45430	0.27636	0.12596
7	2.45400	0.35582	0.09718
8	3.06830	0.51750	0.00072
9	3.06680	0.51755	0.00071

TABLE 5.6

CONTINUED PARTAN

BEALE'S FUNCTION

(Starting Point $x_1 = 2.0$, $x_2 = 0.2$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	2.0	0.2	0.52978
2	2.40950	0.18977	0.26360
3	3.22860	0.51790	0.03991
4	3.02389	0.51260	0.00118
5	3.03660	0.50770	0.00024
6	3.02389	0.50706	0.00012
7	3.02699	0.50645	0.00011

TABLE 5.7

CONTINUED PARTAN

BEALE'S FUNCTION

(Starting Point $x_1 = 5.0$, $x_2 = 0.8$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	5.0	0.8	0.48672
2	5.02550	0.74298	0.15853
3	4.71840	0.74161	0.12342
4	4.75670	0.74241	0.12300
5	4.76620	0.73782	0.12030
6	4.71509	0.73772	0.11836
7	4.71580	0.73421	0.11695
8	4.61350	0.73215	0.11348
9	4.62620	0.72742	0.11066
10	4.54629	0.72656	0.10826
11	4.55260	0.72093	0.10589
12	4.45029	0.71904	0.10164
13	4.45979	0.71389	0.09920
14	4.35750	0.71240	0.09636
15	4.36699	0.70590	0.09240

TABLE 5.8

CONTINUED PARTAN

BEALE'S FUNCTION

(Starting Point $x_1 = 5.0$, $x_2 = 0.2$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	5.0	0.2	18.20472
2	3.36170	0.66020	0.30613
3	4.18080	0.68935	0.07831
4	4.16810	0.69011	0.07788
5	2.93940	0.48443	0.00063
6	2.94890	0.48482	0.00055
7	2.94580	0.48557	0.00050

TABLE 5.9

ITERATED SCALE INVARIANT PARTAN

BEALE'S FUNCTION

(Starting Point $x_1 = 5.0$, $x_2 = 0.2$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	5.0	0.2	18.20470
2	1.72330	0.20000	1.20762
3	2.13279	0.05421	0.52529
4	2.20949	0.05811	0.51296
5	2.18399	0.05810	0.51057
6	2.15850	0.05980	0.50816
7	2.17120	0.05938	0.50786
8	2.18389	0.05940	0.50750
9	2.19340	0.05963	0.50733
10	2.18710	0.05967	0.50720
11	2.18080	0.05956	0.50712
12	2.17609	0.05962	0.50708
13	2.17919	0.05961	0.50704

TABLE 5.10

ITERATED SCALE INVARIANT PARTAN

BEALE'S FUNCTION

(Starting Point $x_1 = 5.0$, $x_2 = 0.8$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	5.0	0.8	0.48672
2	5.81910	0.80000	0.18318
3	5.85740	0.79828	0.18095
4	5.80629	0.79833	0.18053
5	5.82240	0.79833	0.18051

PROGRESS WAS EXTREMELY SLOW

FROM HERE ON

TABLE 5.11

ITERATED SCALE INVARIANT PARTAN

BEALE'S FUNCTION

(Starting Point $x_1 = 2.0$, $x_2 = 0.8$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	2.0	0.8	6.27010
2	5.27670	0.80000	3.13180
3	5.48139	0.77598	0.16594
4	5.37910	0.77668	0.15735
5	5.35359	0.77667	0.15708
6	4.94410	0.76080	0.14897
7	5.04640	0.76454	0.14833
8	5.14869	0.76453	0.14468
9	5.15019	0.76448	0.14457
10	5.12469	0.76449	0.14449
11	5.13099	0.76449	0.14447

TABLE 5.12

ITERATED SCALE INVARIANT PARTAN

BEALE'S FUNCTION

(Starting Point $x_1 = 2.0$, $x_2 = 0.2$)

NUMBER OF ITERATIONS	x_1	x_2	$F(x_1, x_2)$
1	2.0	0.2	0.52978
2	2.40950	0.20000	0.24194
3	2.38400	0.30215	0.12788
4	3.20310	0.52005	0.02520
5	3.10080	0.52005	0.00186
6	3.08810	0.52333	0.00126
7	3.08960	0.52328	0.00125

TABLE 5.13

ITERATED STEEPEST DESCENT PARTAN

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.31$, $\omega_2 = 10.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.31	10.00	0.022460
2	0.29089	10.51047	0.022040
3	0.36760	17.68500	0.018744
4	0.36290	17.68680	0.018733
5	0.36280	17.01969	0.018720
6	0.35970	17.02081	0.018712
7	0.35968	17.02082	0.018710

TABLE 5.14

ITERATED STEEPEST DESCENT PARTAN

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.41$, $\omega_2 = 20.0$)

NUMBER OF ITERATIONS	ω_1	ω_1	$F(\omega_1, \omega_2)$
1	0.41	20.0	0.019695
2	0.37170	20.00380	0.018940
3	0.37160	17.48280	0.018787
4	0.37170	17.81358	0.018770
5	0.36540	17.81840	0.018740
6	0.36530	17.49109	0.018730
7	0.36050	17.24380	0.018717
8	0.36099	16.99185	0.018712

TABLE 5.15

ITERATED STEEPEST DESCENT PARTAN

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.41$, $\omega_2 = 11.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.41	11.0	0.025100
2	0.30769	11.11289	0.020940
3	0.30879	13.60482	0.019886
4	0.30909	13.21873	0.019862
5	0.33460	13.26826	0.019301
6	0.33299	14.78621	0.018984
7	0.34570	14.77852	0.018866
8	0.34599	16.01601	0.018774
9	0.35224	15.77618	0.018745
10	0.35235	16.24918	0.018724
11	0.35234	16.24920	0.018723

TABLE 5.16

ITERATED STEEPEST DESCENT PARTAN

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.31$, $\omega_2 = 20.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.31	20.0	0.026307
2	0.38670	19.97715	0.019010
3	0.38660	19.13796	0.019000
4	0.38510	19.98199	0.018985
5	0.37240	19.92556	0.018924
6	0.37550	18.18123	0.018822
7	0.36599	18.18377	0.018760
8	0.36589	17.24587	0.018735
9	0.35959	17.24911	0.018718
10	0.35949	16.76195	0.018714

TABLE 5.17

CONTINUED PARTAN

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.41$, $\omega_2 = 11.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.41	11.0	0.025100
2	0.30769	11.11275	0.020940
3	0.32039	11.16241	0.020816
4	0.31409	11.21918	0.020762
5	0.33199	11.49548	0.020579
6	0.32049	11.53123	0.020436
7	0.31420	11.64940	0.020369
8	0.32689	11.70720	0.020293
9	0.32059	11.73768	0.020253
10	0.31109	12.97309	0.019805
11	0.33660	12.97902	0.019450
12	0.33030	12.99631	0.019426
13	0.32400	13.27087	0.019379
14	0.33669	13.28418	0.019300
15	0.33599	13.29429	0.019290
16	0.33209	13.30840	0.019280
17	0.35760	14.93219	0.018938
18	0.35730	14.93267	0.018932

TABLE 5.18

CONTINUED PARTAN

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.41$, $\omega_2 = 20.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.41	20.0	0.019695
2	0.37170	20.00380	0.018940
3	0.37639	19.99561	0.018922
4	0.37629	19.99545	0.018921

TABLE 5. 19

CONTINUED PARTAN

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.31$, $\omega_2 = 20.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.31	20.0	0.026307
2	0.38670	19.97715	0.019008
3	0.37399	19.97590	0.018923
4	0.37710	19.94620	0.018913
5	0.37560	19.94450	0.018911

TABLE 5.20

CONTINUED PARTAN

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.31$, $\omega_2 = 10.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.31	10.0	0.022460
2	0.29089	10.51047	0.022040
3	0.31640	10.54909	0.021587
4	0.30370	10.68752	0.021475
5	0.31639	10.74097	0.021320
6	0.30370	10.91876	0.021211
7	0.31640	10.95692	0.021047
8	0.30369	11.26435	0.020880
9	0.32279	11.29986	0.020675
10	0.31650	11.34355	0.020625
11	0.34199	12.08583	0.020165
12	0.32289	12.12182	0.019951
13	0.34839	12.98325	0.019617
14	0.32929	13.00912	0.019423
15	0.33879	13.14385	0.019380
16	0.33249	13.15878	0.019340
17	0.33569	13.23326	0.019319
18	0.33599	13.24582	0.019311

TABLE 5.21

ITERATED SCALE INVARIANT PARTAN

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.31$, $\omega_2 = 10.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.31	10.0	0.022460
2	0.30929	10.00000	0.022451
3	0.29660	11.18796	0.021170
4	0.29630	12.51777	0.020655
5	0.33460	12.51777	0.019713
6	0.32829	12.51777	0.019684
7	0.32199	13.87104	0.019311
8	0.32194	14.40805	0.019300
9	0.34745	14.40805	0.018960
10	0.34734	16.54398	0.018786
11	0.34725	16.12352	0.018776
12	0.35355	16.12352	0.018724
13	0.35350	16.64369	0.018723

TABLE 5.22

ITERATED SCALE INVARIANT PARTAN

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.31$, $\omega_2 = 20.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.31	20.0	0.026307
2	0.38669	20.00000	0.019009
3	0.38529	19.98033	0.018987
4	0.37579	19.98033	0.018920
5	0.37569	18.14513	0.018825
6	0.36619	18.14513	0.018757
7	0.36570	17.26473	0.018733
8	0.36099	17.26473	0.018717
9	0.36109	17.26473	0.018716

TABLE 5.23

ITERATED SCALE INVARIANT PARTAN

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.41$, $\omega_2 = 20.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.41	20.0	0.019695
2	0.37170	20.00000	0.018940
3	0.37170	17.48247	0.018790
4	0.37189	17.81199	0.018780
5	0.36560	17.81199	0.018739
6	0.36410	17.81199	0.018738

TABLE 5.24

ITERATED SCALE INVARIANT PARTAN

GIBSON'S SYSTEM

(Starting Point $\omega_1 = 0.41$, $\omega_2 = 11.0$)

NUMBER OF ITERATIONS	ω_1	ω_2	$F(\omega_1, \omega_2)$
1	0.41	11.0	0.025100
2	0.30769	11.00000	0.021057
3	0.30659	13.49565	0.020009
4	0.30689	13.13361	0.019978
5	0.33239	13.13361	0.019360
6	0.32829	14.51513	0.019102
7	0.32819	14.51513	0.019100

CHAPTER 6

CONCLUSION

Implementation of an adaptive control in situations where plant identification is either inconvenient or undesirable, has two important aspects. One is the figure of merit or performance by which the system measures its performance with a view to optimize it. The other is: of different methods available for adjusting the parameters of a controller to affect this performance optimization, which is the method that adjusts these parameters most quickly and efficiently. In the past few years, several minimization techniques have been developed but very little work has been done to find a method which works satisfactorily in most cases. In this thesis three optimizing techniques, namely:

(i) Method of Steepest Descent

(ii) Method of Conjugate Gradients

(iii) Method of Parallel Tangents and its Variants

have been examined. The last two methods are quite recent and have been claimed to work in most cases. These methods are applied to optimize Beale's function and ITSE performance criterion in a control system. Beale's function represents a response surface, most likely to be encountered in practical situations. The comparison of different methods is shown in Figures (13-21). It can be seen that the Steepest Descent variant of the parallel tangents works better than any in both the cases beginning with different starting points. Continued Partan also shows good performance in most cases.

It has been claimed that the method of conjugate gradients locates the minimum of a linear positive definite quadratic function of n variables in (n) steps and the method of parallel tangents finds the optimum in (2n-1) steps. The functionsto be optimized in practice rarely meet the ideal requirements of being linear and quadratic. Both the functions optimized in this thesis are highly non-quadratic and it can be seen that the minimum is not reached in (n) or (2n-1) steps. From the results obtained in this thesis, it can be seen that whereas the Steepest Descent variant of the method of parallel tangents works fairly well in case of two-parameter optimization, the progress in the method of conjugate gradients is very slow. Surprisingly the method of steepest descent seems to work better in some cases than the method of conjugate gradients although the latter is a modification of the former. Fletcher and Reeves [23] have reported similar experience with the method of conjugate gradients applied to a non-quadratic function. To overcome the slow convergence of the method in non-ideal cases, they reverted to the steepest descent direction in place of conjugate direction whenever the progress was slow. They attributed this to the fact that the successive conjugate directions were so nearly parallel that the points x_i were scarcely separated. Lasdon [31] has formulated different theorems on the method of conjugate gradients and has reported success with positive definite quadratic functions, but at the present very little experience is available with non-quadratic functions. On the basis of the present work, however, the method of conjugate gradients did not work well.

The optimization calculations in the present work have been carried out on an IBM-7040 computer. In case of Beale's function, computations were done using a WATFOR compiler developed by the University of Waterloo Computing Centre. The typical running time was one to two seconds for each iteration as against one minute per iteration by Shah et al [27] for the same function and same starting points. Gibson's System required the solution of a sixth order differential equation and integration of the resulting error function to form the integral performance criterion (ITSE). The digital computer was employed to obtain the approximate numerical solution of the system differential equation using the fourth order Runge-Kutta method [26]. The integration was also done numerically and was required several times for a single iteration and thus involves the inherent inaccuracy present in the numerical integration and large computer time. Faster and more accurate results could have been obtained by using a hybrid computer.

From the results obtained in this work the method of parallel tangents seems to work quite well in the case of practical response surface with two variable parameters. Although this method finds the optimum in less number of iterations, it requires large amounts of stored information and hence a larger computer. From the practical point of view e.g. in space applications and flight control etc., where the weight is one of the main considerations, the method of steepest descent seems attractive because of least stored information requirements, but unfortunately it requires large numbers of iterations to reach the optimum. Thus in practical applications, the choice of optimization technique is a compromise between many conflicting requirements.

Moreover, the method of steepest descent can also be implemented with the help of a small analogue computer, which seems to be the answer for space applications.

Some areas of further work are indicated by this study. It would be interesting to investigate the performance of the method of parallel tangents in the optimization of more than two-parameter cases. The scale invariant option may be developed and investigated further for better results. The method of conjugate gradients may be applied to more non-ideal functions before generalizing anything on its performance in such cases.

FIGURE 13: Beale's Function
Starting Point $x_1 = 2.0$, $x_2 = 0.8$

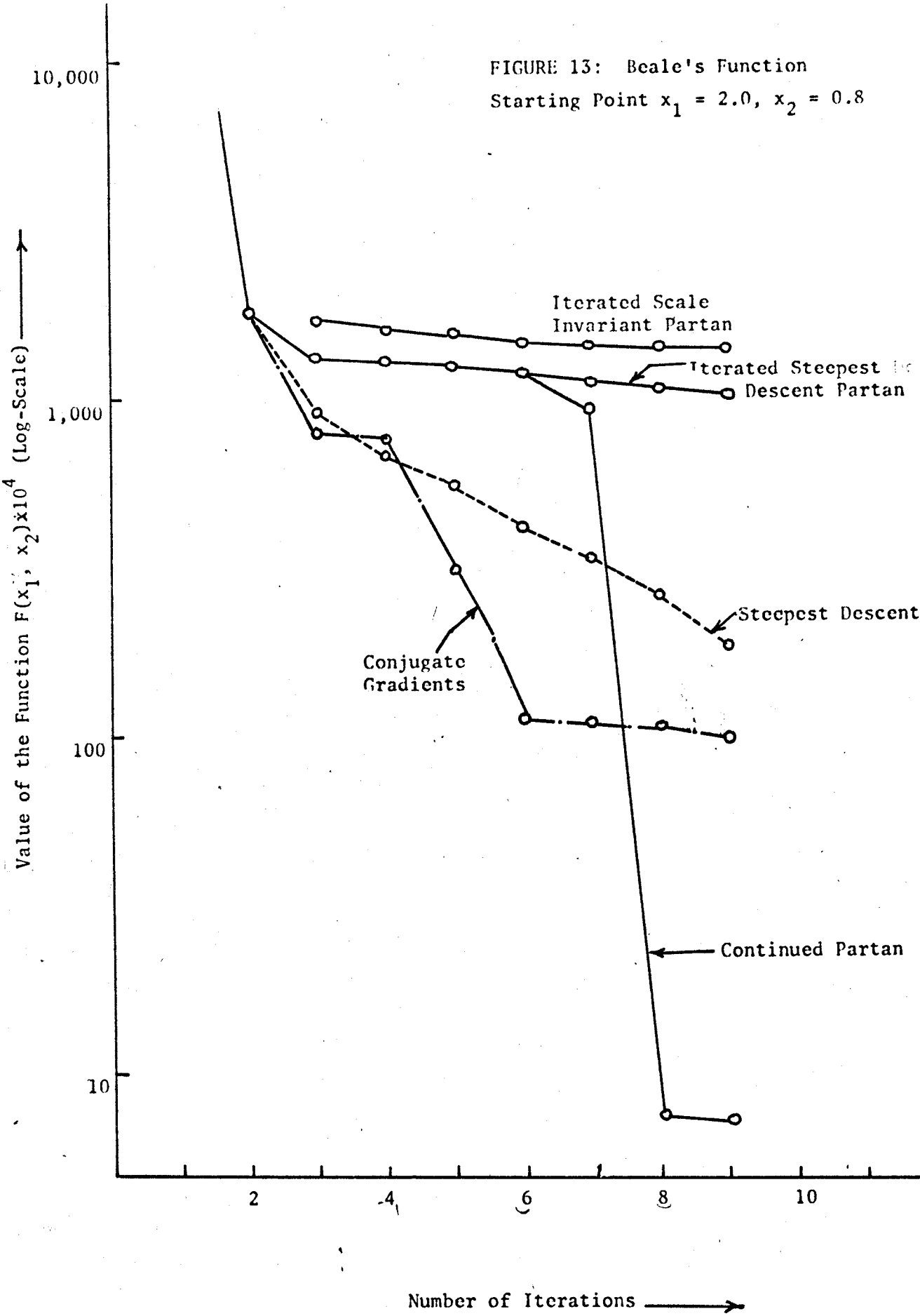
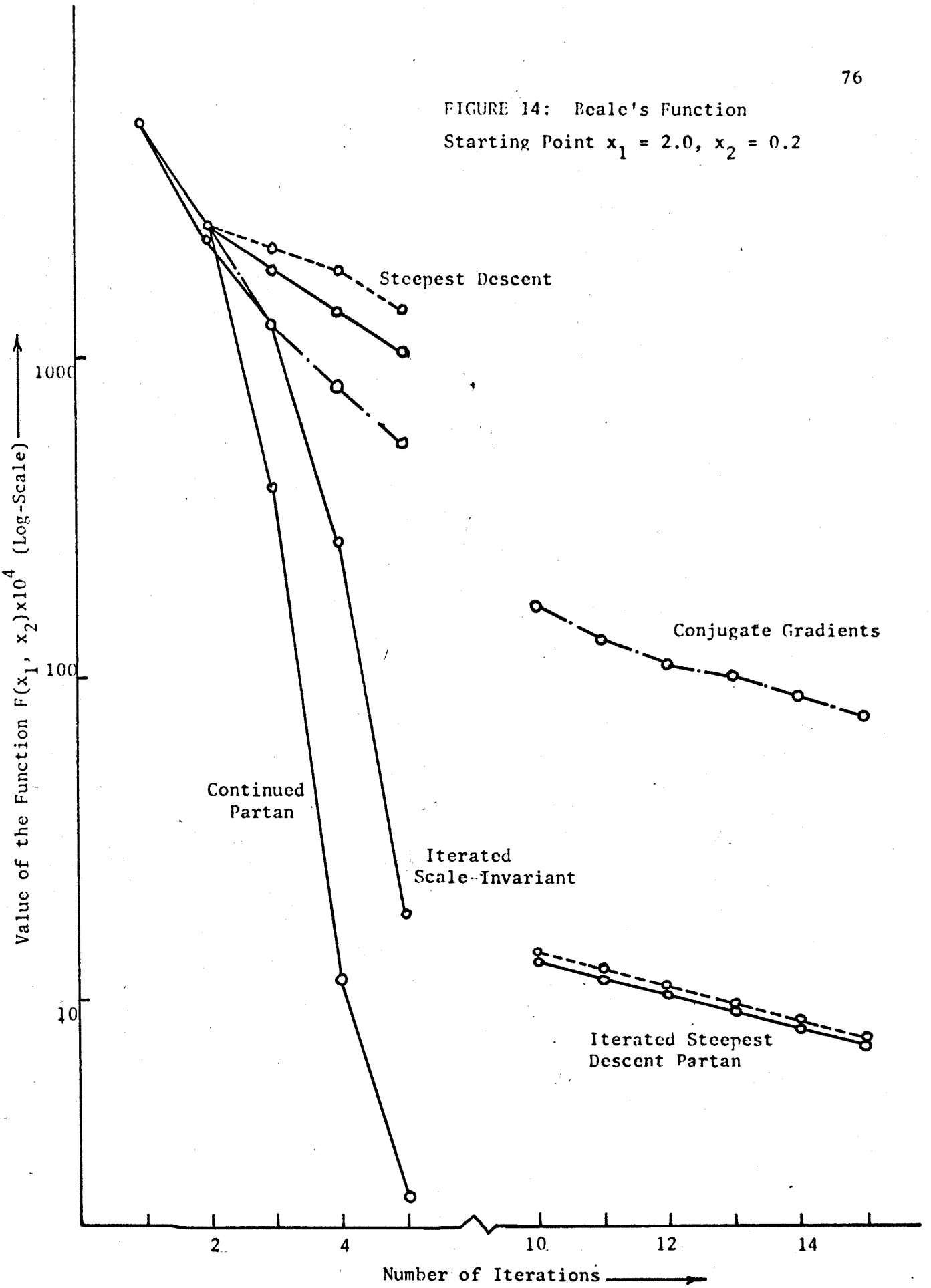


FIGURE 14: Beale's Function
Starting Point $x_1 = 2.0$, $x_2 = 0.2$



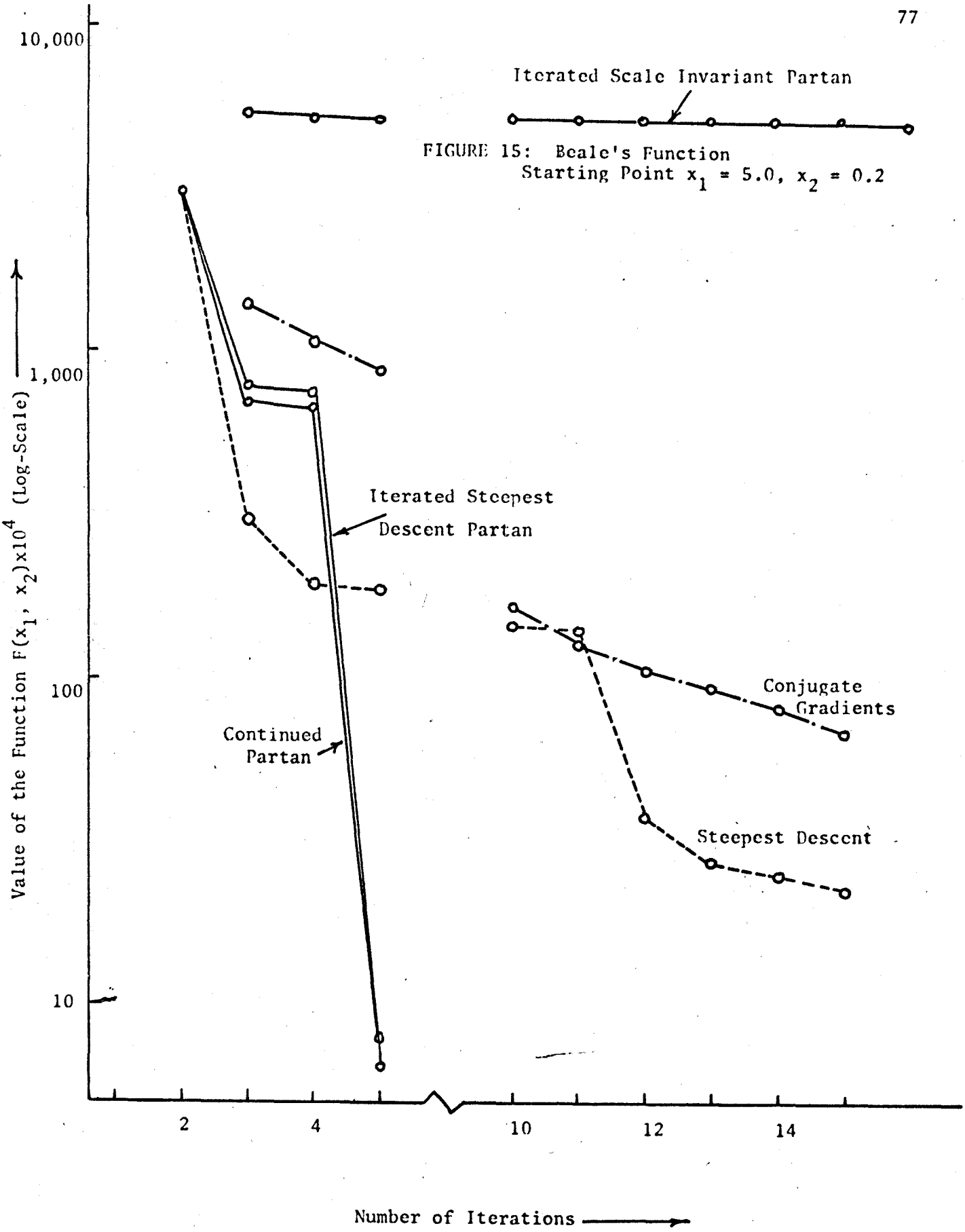
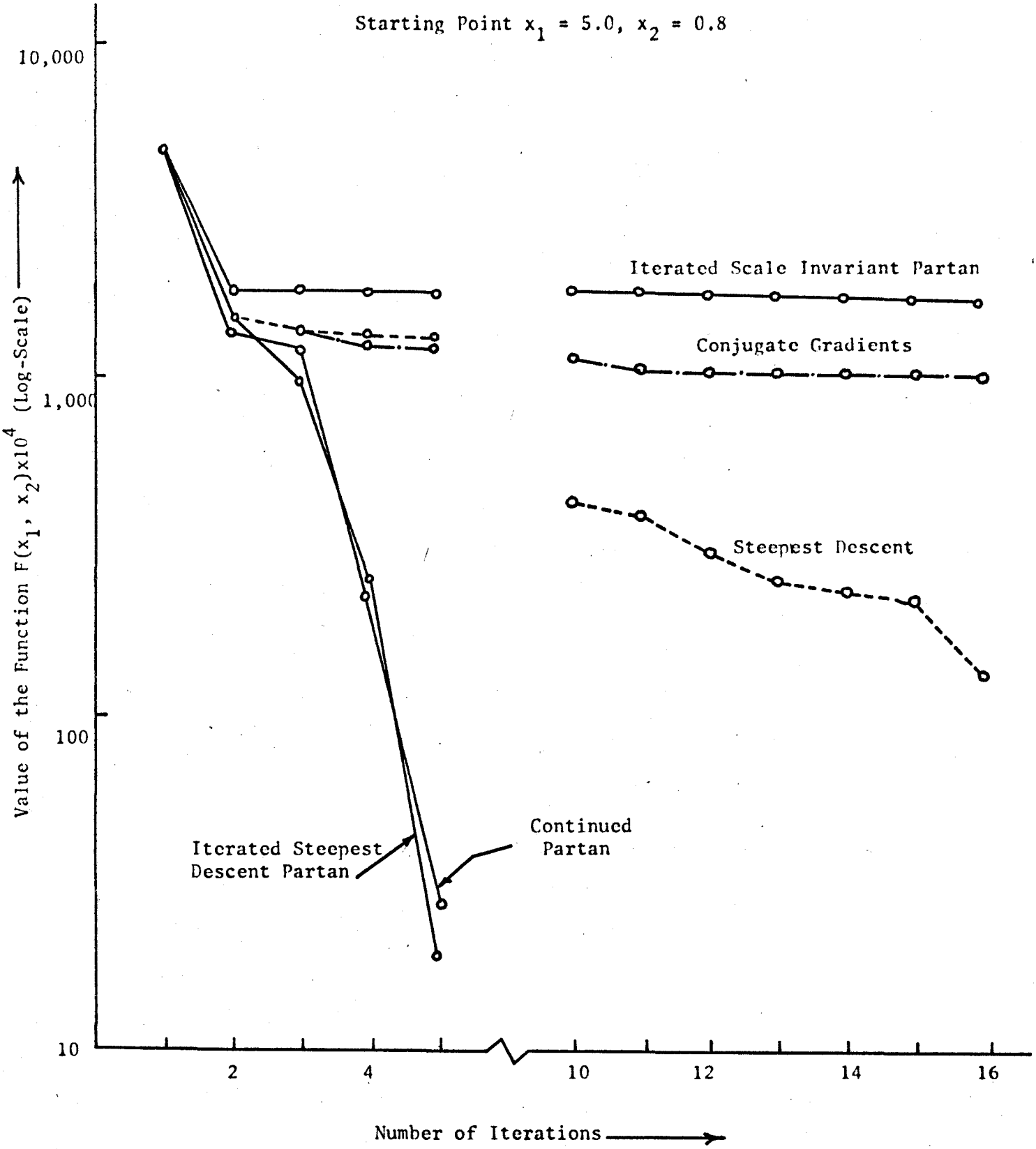


FIGURE 15: Beale's Function
Starting Point $x_1 = 5.0$, $x_2 = 0.2$

FIGURE 16: Beale's Function

Starting Point $x_1 = 5.0$, $x_2 = 0.8$



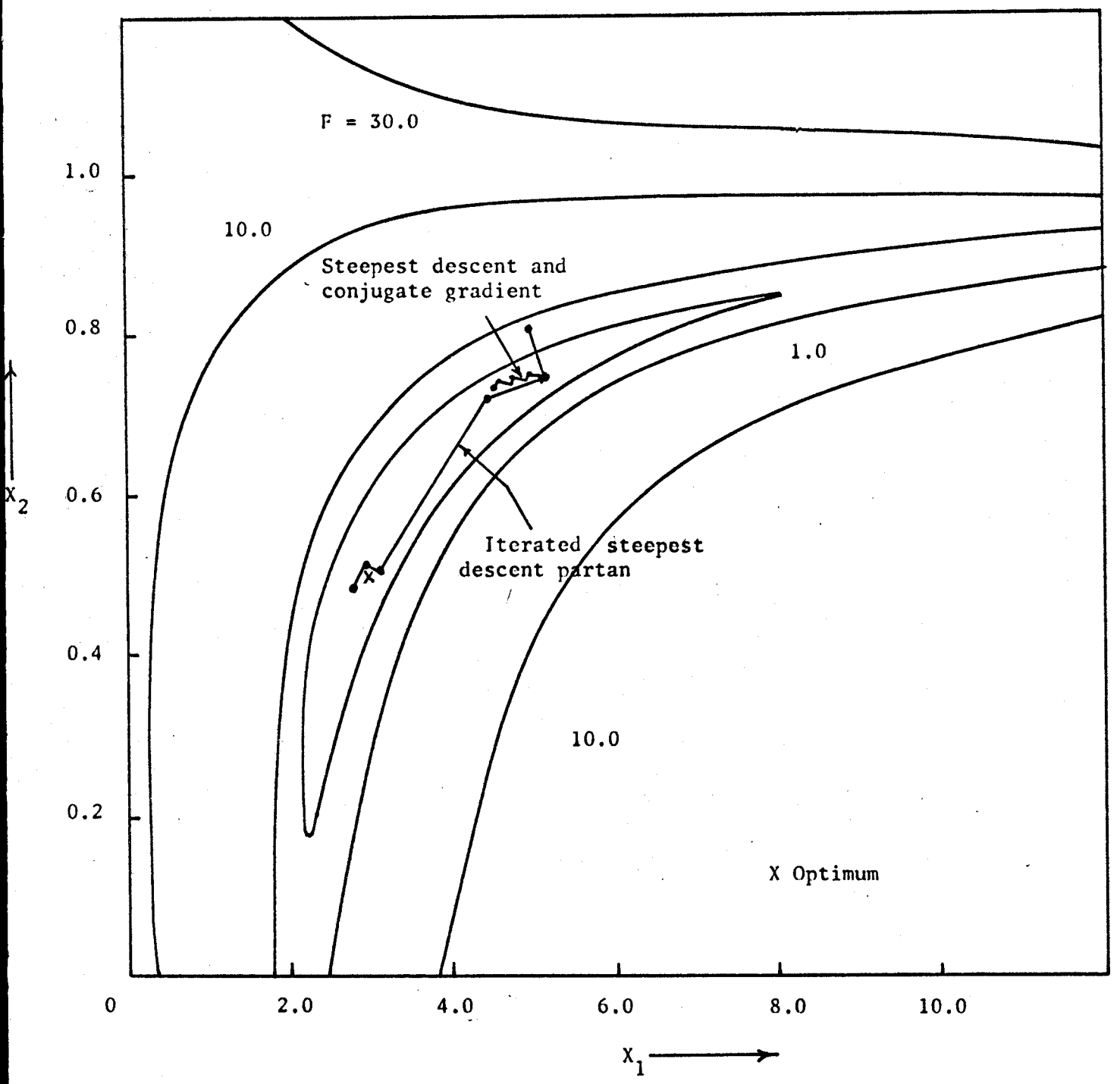


FIGURE 17 : Path traced by different methods for six iterations on the response surface represented by Beale's Function.

FIGURE 18: Gibson's System
(Starting Point $\omega_1 = 0.31$, $\omega_2 = 10.0$)

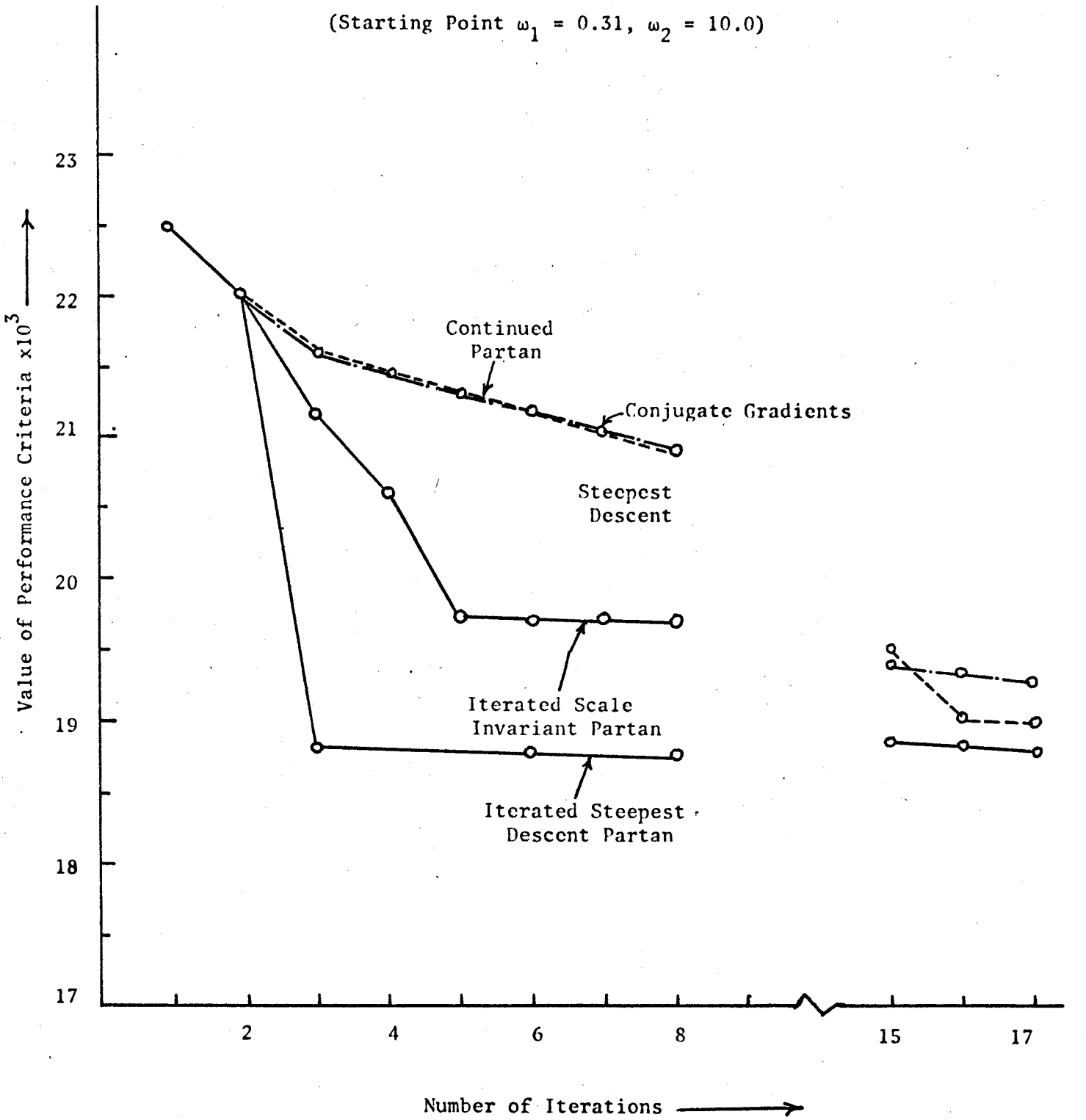


FIGURE 19: Gibson's System

(Starting Point $\omega_1 = 0.41$, $\omega_2 = 20.0$)

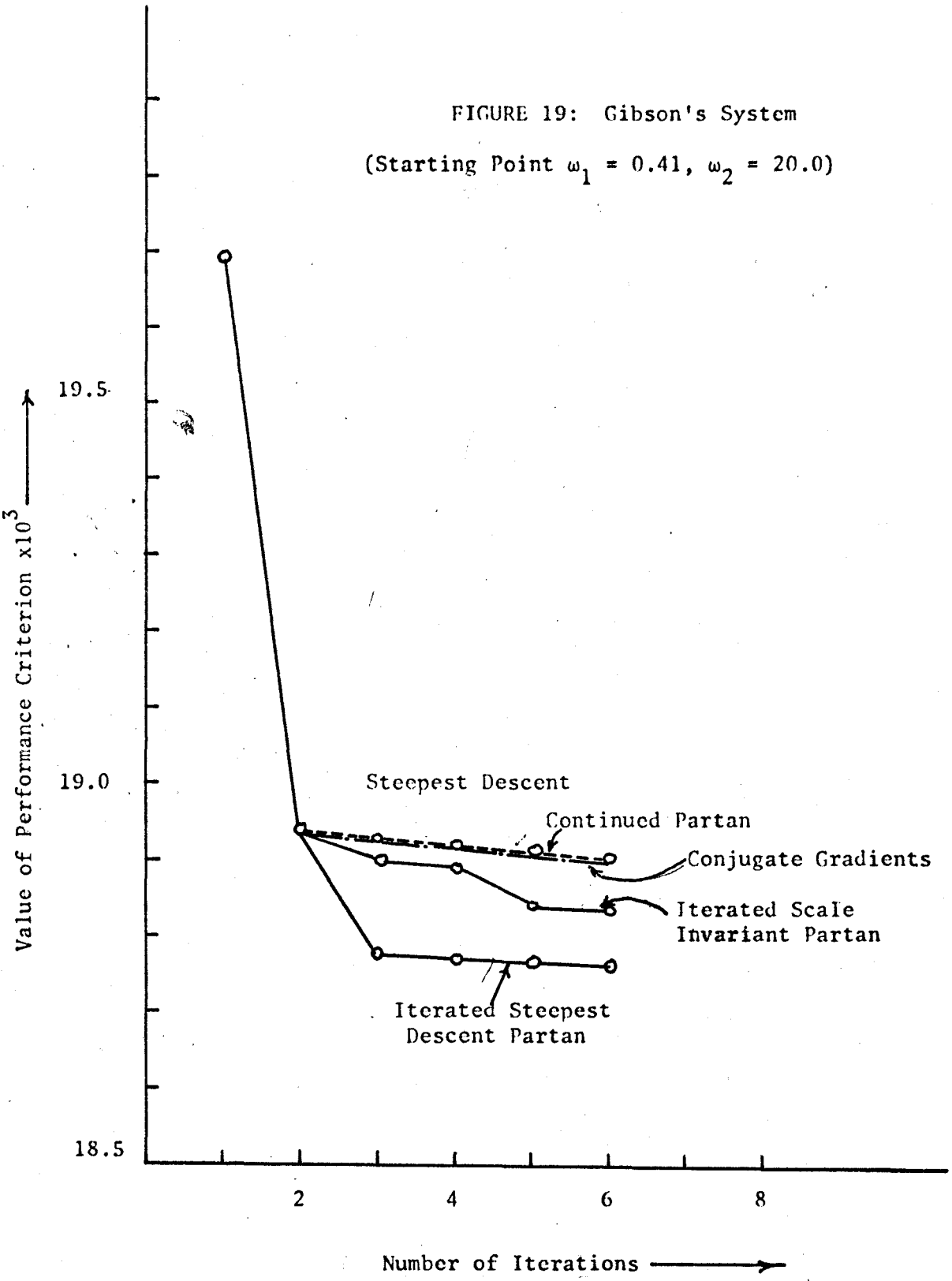
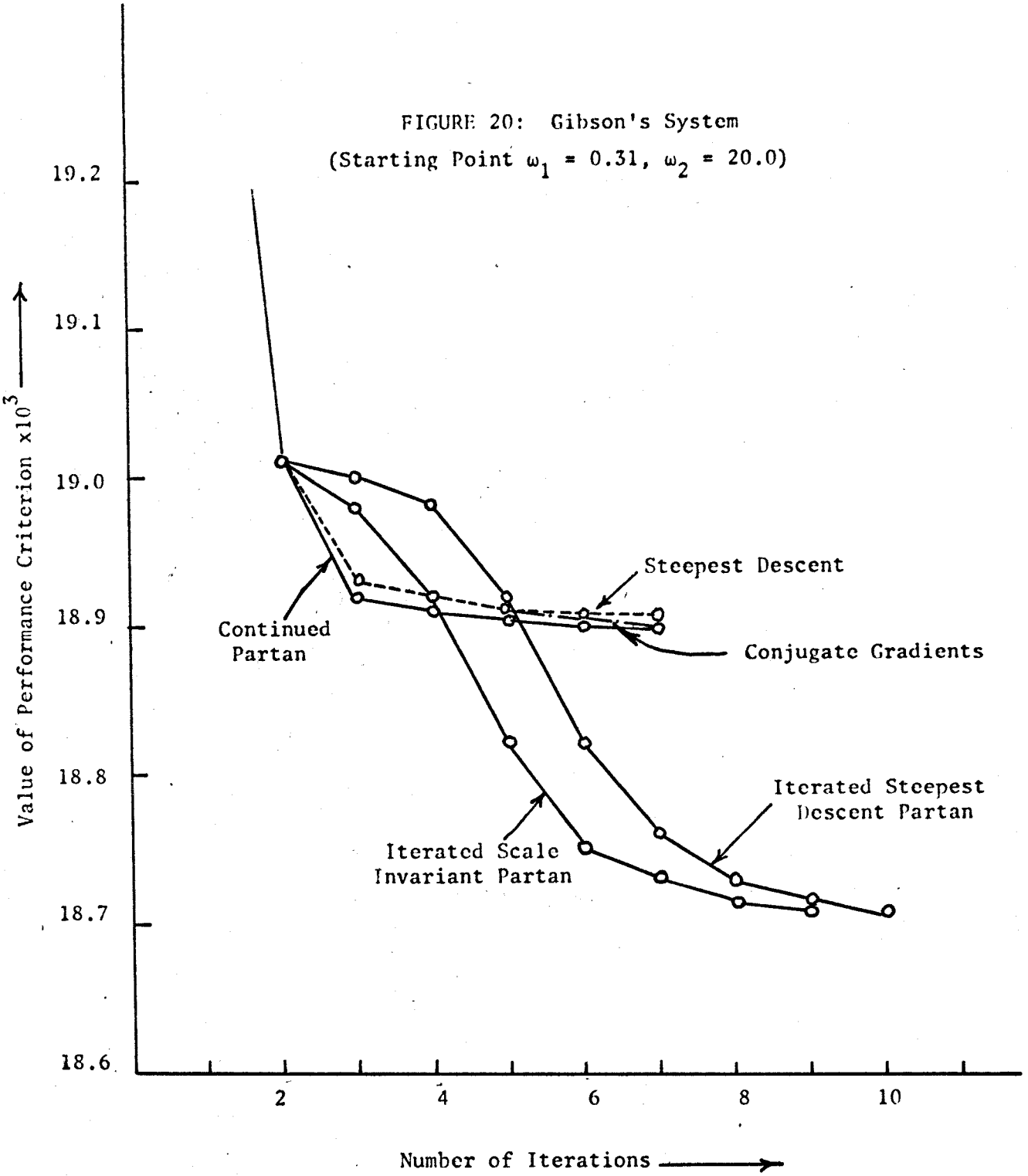


FIGURE 20: Gibson's System
(Starting Point $\omega_1 = 0.31$, $\omega_2 = 20.0$)



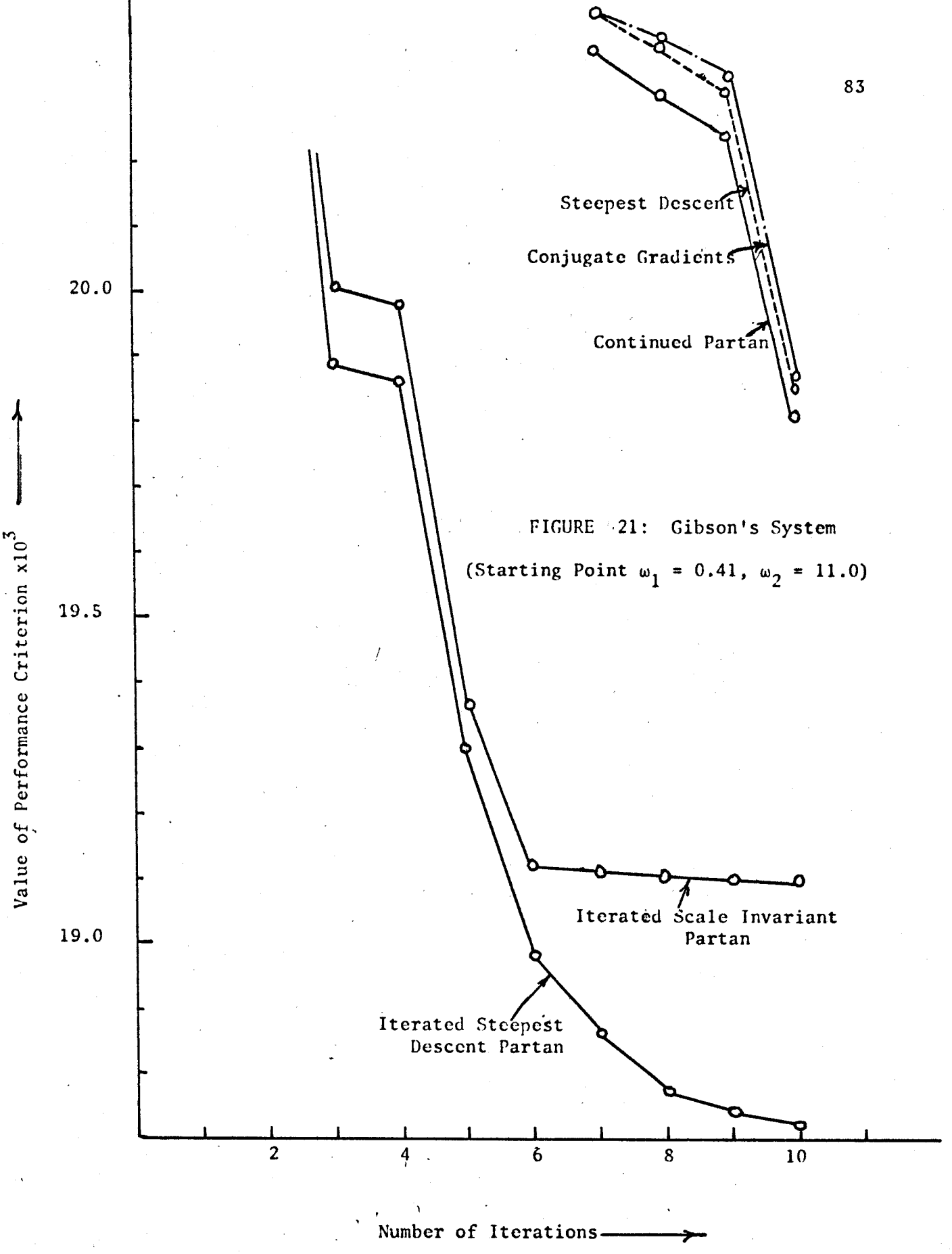


FIGURE 21: Gibson's System

(Starting Point $\omega_1 = 0.41$, $\omega_2 = 11.0$)

REFERENCES

1. Aseltine, J. A., Mancini, A. R. and Sarture, C. W., "A survey of adaptive control systems", IRE Trans. on Automatic Control, December, 1958.

2. Gregory, P. C., (ed.), "Proceedings of the self-adaptive flight control systems symposium", WADC Tech. Rept. 59-49, Wright-Patterson Air Force Base, Ohio, March, 1959.

3. Coales, J. F., Ragazzini, J. R. and Fuller, A. T. "Automation and remote control: Proceedings of the First International Congress of the IFAC, Moscow- 1960", London, Butterworth, 1961.

4. Mishkin, E., and Braun, L. Jr. (eds.)"Adaptive Control Systems", New York, McGraw Hill Book Co., Inc., 1961.

5. Jacobs, O. L. R., "A review of self-adjusting systems in automatic control", J. Electronics and Control, 1961.

6. Mill, P. W., "Review of the literature of Adaptive Control", Trans. Soc. Inst. Technology (SIT, G.B.) June, 1963.

7. Caruthers, F. P., and Levenstein, H., "Adaptive Control Systems: Proceedings of a symposium held in Garden City, Long Island, New York, October, 1960", New York, the MacMillan Co., 1963.

8. Chestnut, H., "Systems Engineering Tools", New York, John Wiley and Sons, Inc., 1965.

9. Wilde, D. J., "Optimum Seeking Methods", New Jersey, Prentice-Hall, Inc., 1964.

10. Rosenbrock, H. H., and Storey, C., "Computational Techniques for Chemical Engineers", New York, Pergamon Press, 1966.

11. Zaborszky, J., and Humphery, W. L., "Control without model or plant identification", IEEE Trans. on Applications and Industry, Nov. 1964.

12. Narendra, K. S., and Streeter, D. N., "An adaptive procedure for controlling undefined linear processes", IEEE Trans. on Automatic Control, Oct., 1964.

13. Sinha, N. K., "Adaptive Control Systems: A General Classification Scheme", Tech. Rept. No. 2, Control Theory Group, The University of Tennessee, October, 1964.

14. Graham, D, and Lathrop, L. C. "The Synthesis of Optimum Transient Response: Criteria and Standard Forms", IEEE Trans. on Applications and Industry, Nov., 1953.

15. Leitman, G., (ed.), "Optimization Techniques", New York, Academic Press, 1962.

16. Tou, J. T., "Modern Control Theory", New York, McGraw-Hill Book Co., 1964.

17. Beckenbach, E. F., "Modern Mathematics for the Engineer", New York, McGraw-Hill Book Co., 1956.

18. Miele, A., (ed.), "Theory of Optimum Aerodynamic Shapes", New York, Academic Press, 1965.

19. Bellman, R., "Dynamic Programming", Princeton, Princeton University Press, 1957.

20. Pontryagin, L. S., Boltyanskii, V. G., Gamkrelidze, R. V., and Mishchenko, E. F., "The Mathematical Theory of Optimal Processes", New York, John Wiley and Sons, Inc., 1962.

21. Booth, A. D., "Numerical Methods", London, Butterworths, 1957.

22. Rosenbrock, H. H., and Storey, C., "Computational Techniques for Chemical Engineers", London, Pergamon Press, 1965.

23. Fletcher, R., and Reeves, C. M., "Function Minimization by Conjugate Gradients", British Computer Journal, July, 1964.

24. Hestenes, M. R., and Steifel, E., "Method of Conjugate Gradients for Solving Linear Systems", J. Res. N. B. S., Vol. 49, 1952.

25. Buehler, R. J., Shah, B. V., and Kempthorne, O., "Methods of Parallel Tangents" Chemical Engineering Progress Symposium Series, Vol. 60, 1964.

26. Ralston, A., and Wilf, H. S. (eds.), "Mathematical Methods for Digital Computers", New York, John Wiley and Sons, Inc., 1960.

27. Buehler, R. J., Shah, B. V. and Kempthorne, O, "Some Algorithms for Minimizing a Function of Several Variables", J. Soc. Indust. Appl. Math., March, 1964.

28. Gibson, J. E., "Self-Optimizing or Adaptive Control Systems" in reference 3.

29. Cress, P., "MIMIC: A Digital-Analog Simulator", Computing Centre, University of Waterloo, Waterloo, Ontario, February, 1966.

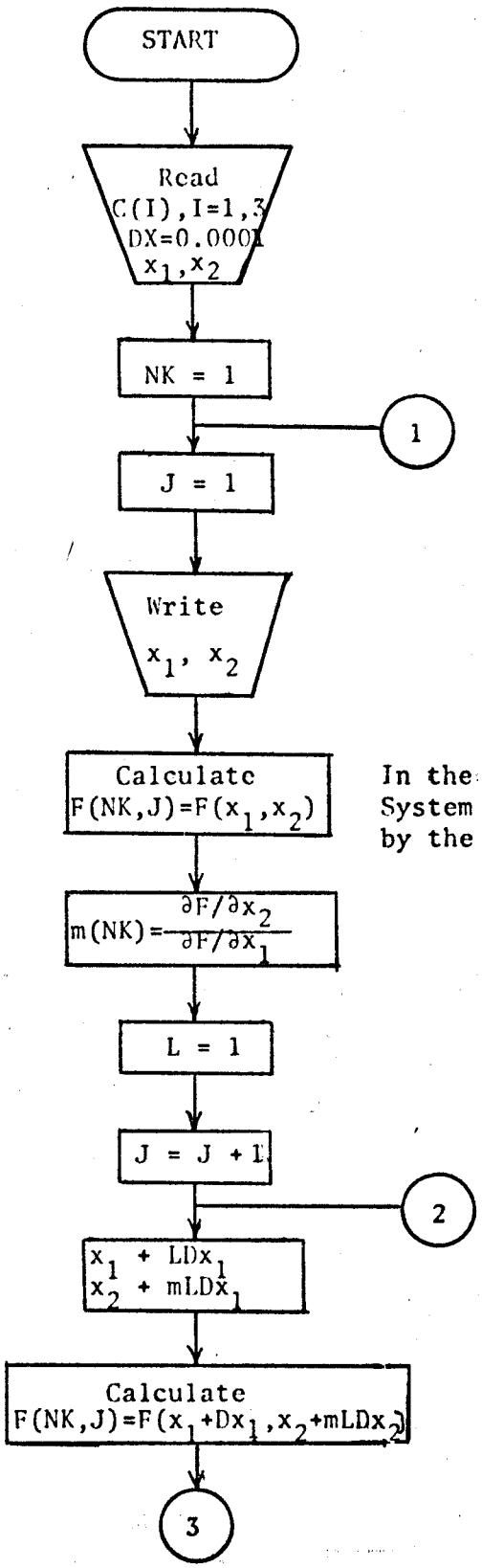
30. Hinchley, E. M., Department of Electrical Engineering, University of Waterloo, Private Communication with Author, September, 1966.

31. Lasdon, L. S., Warren, A. D. and Mitter, S. K., "Conjugate Gradient Method for Optimal Control Problem", to be published in SIAM Journal of Control.

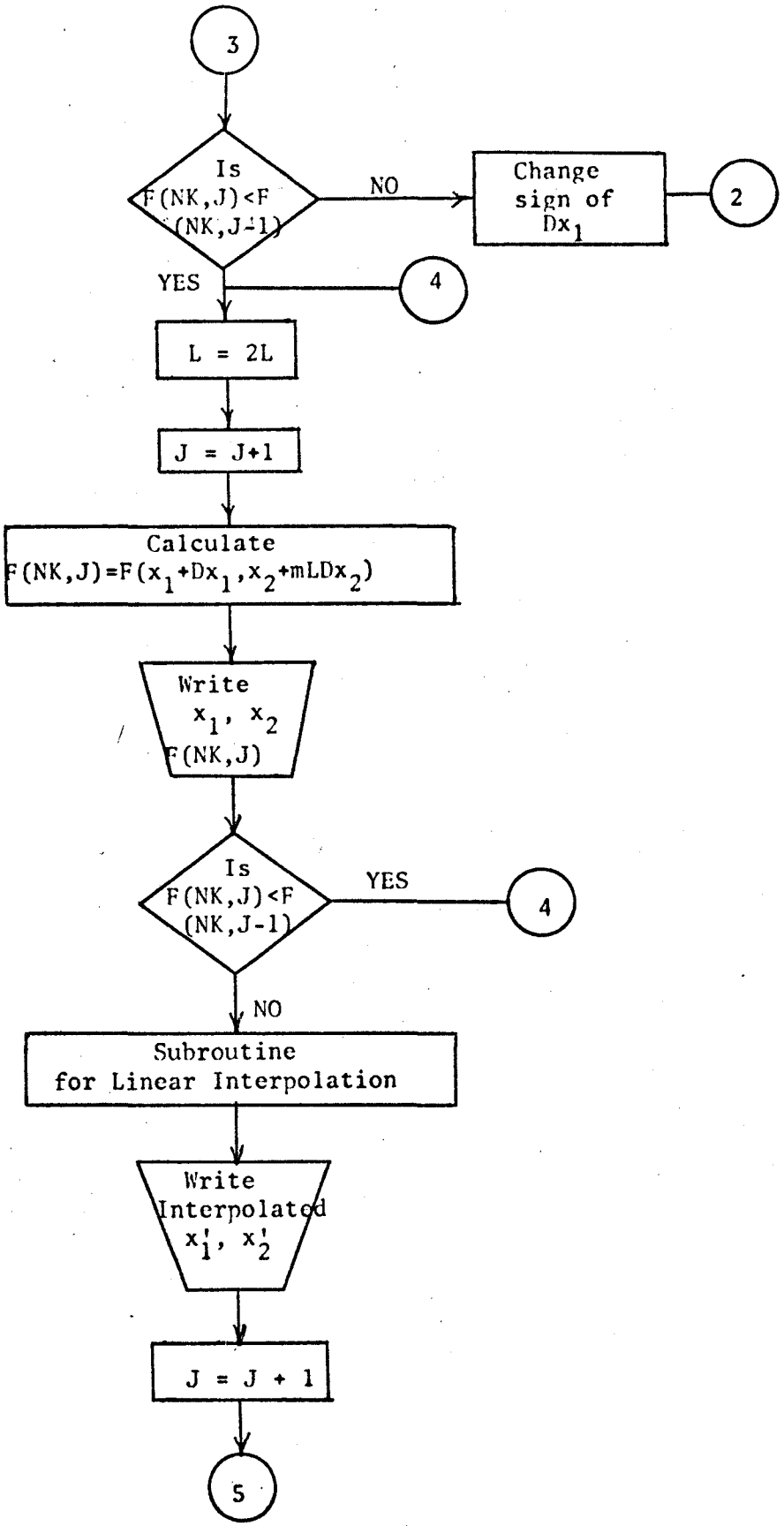
APPENDIX

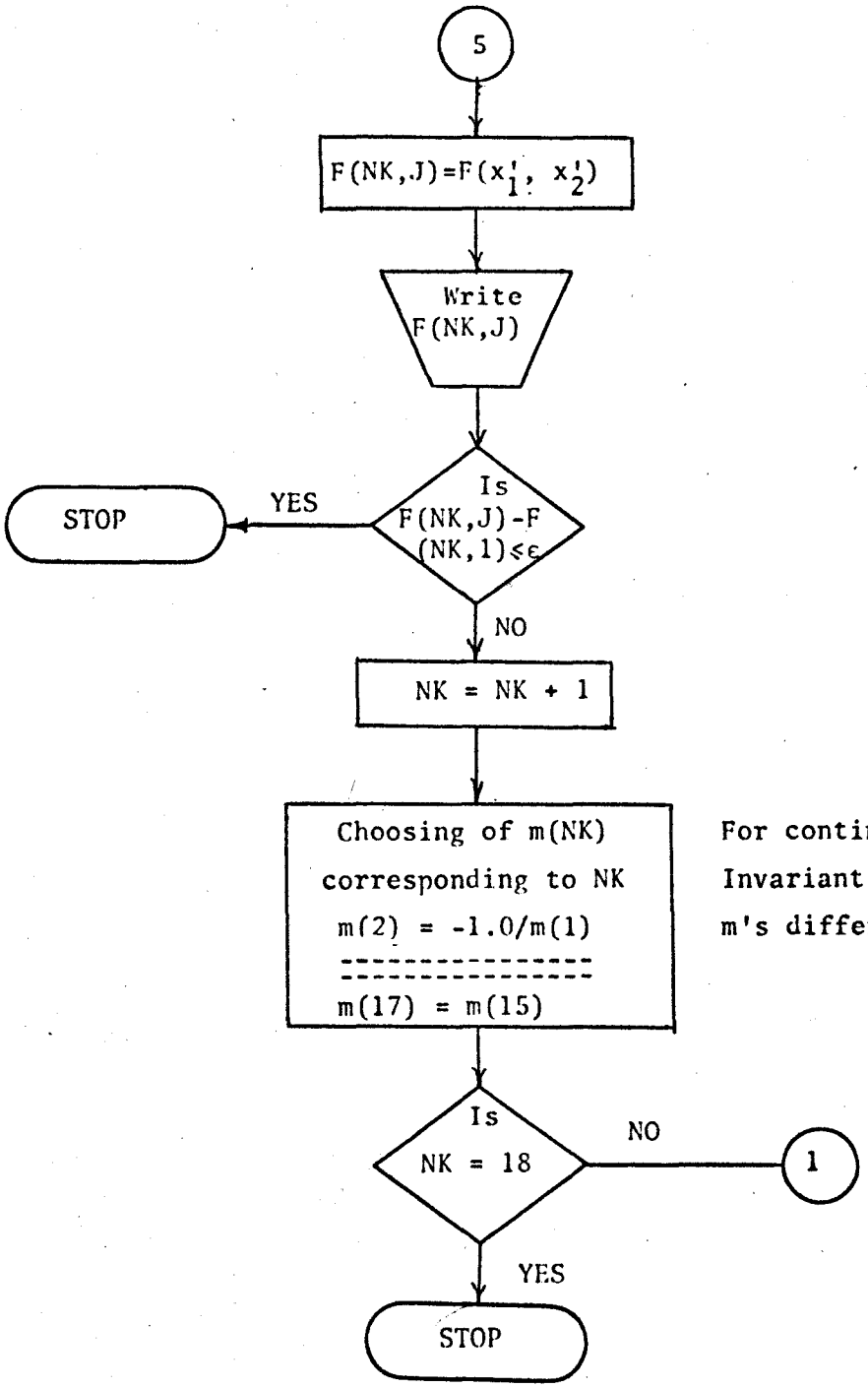
APPENDIX I

Flow Chart for the Method of Parallel Tangents



In the case of Gibson's System F(NK,J) is obtained by the use of Subroutine DEQSET

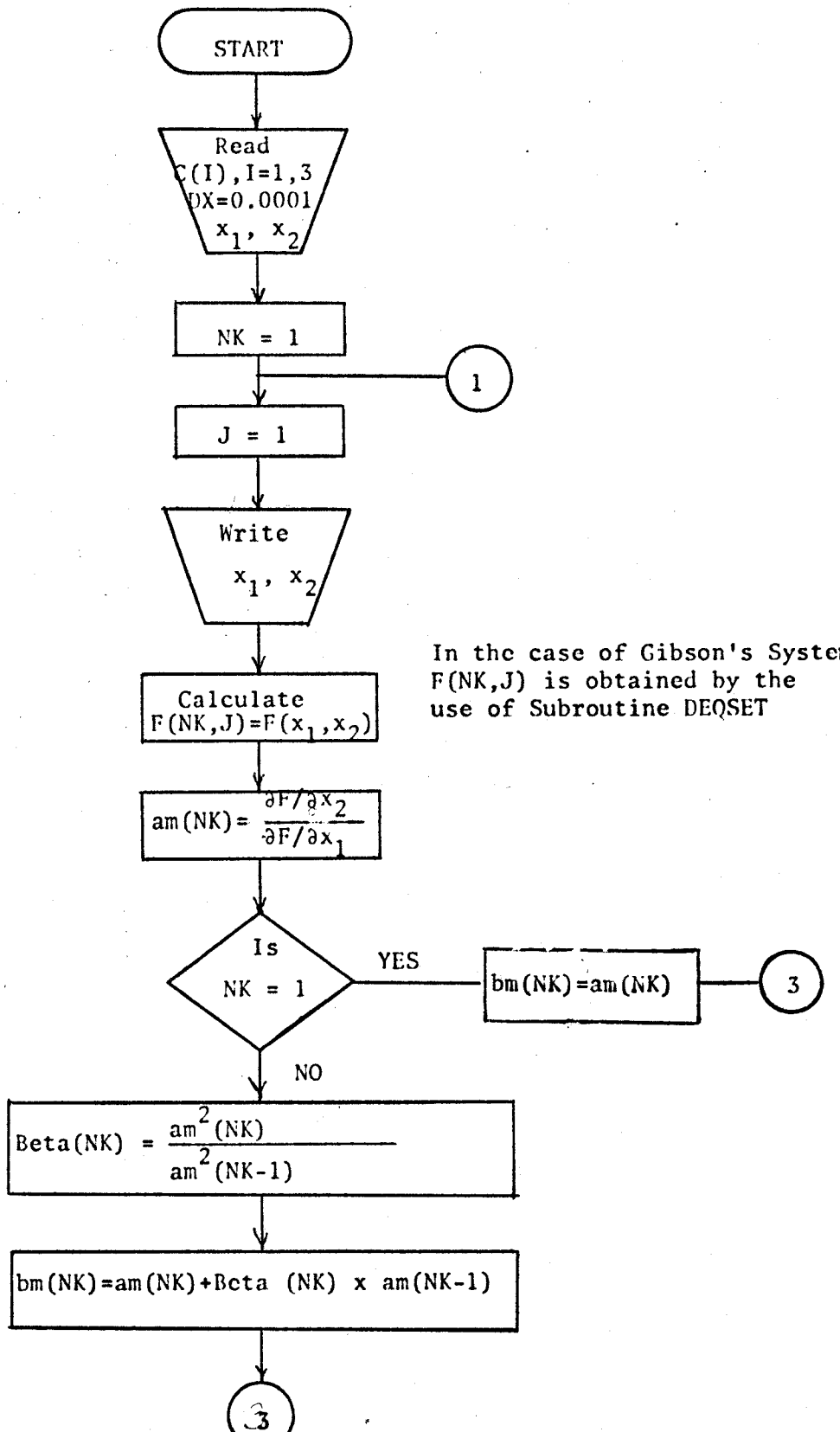

3
Is F(NK,J)<F (NK,J-1)
NO
Change sign of Dx_1
2
YES
4
L = 2L
J = J+1
Calculate F(NK,J)=F(x_1+Dx_1,x_2+mLDx_2)
Write x_1, x_2 F(NK,J)
Is F(NK,J)<F (NK,J-1)
YES
4
NO
Subroutine for Linear Interpolation
Write Interpolated x_1', x_2'
J = J + 1
5

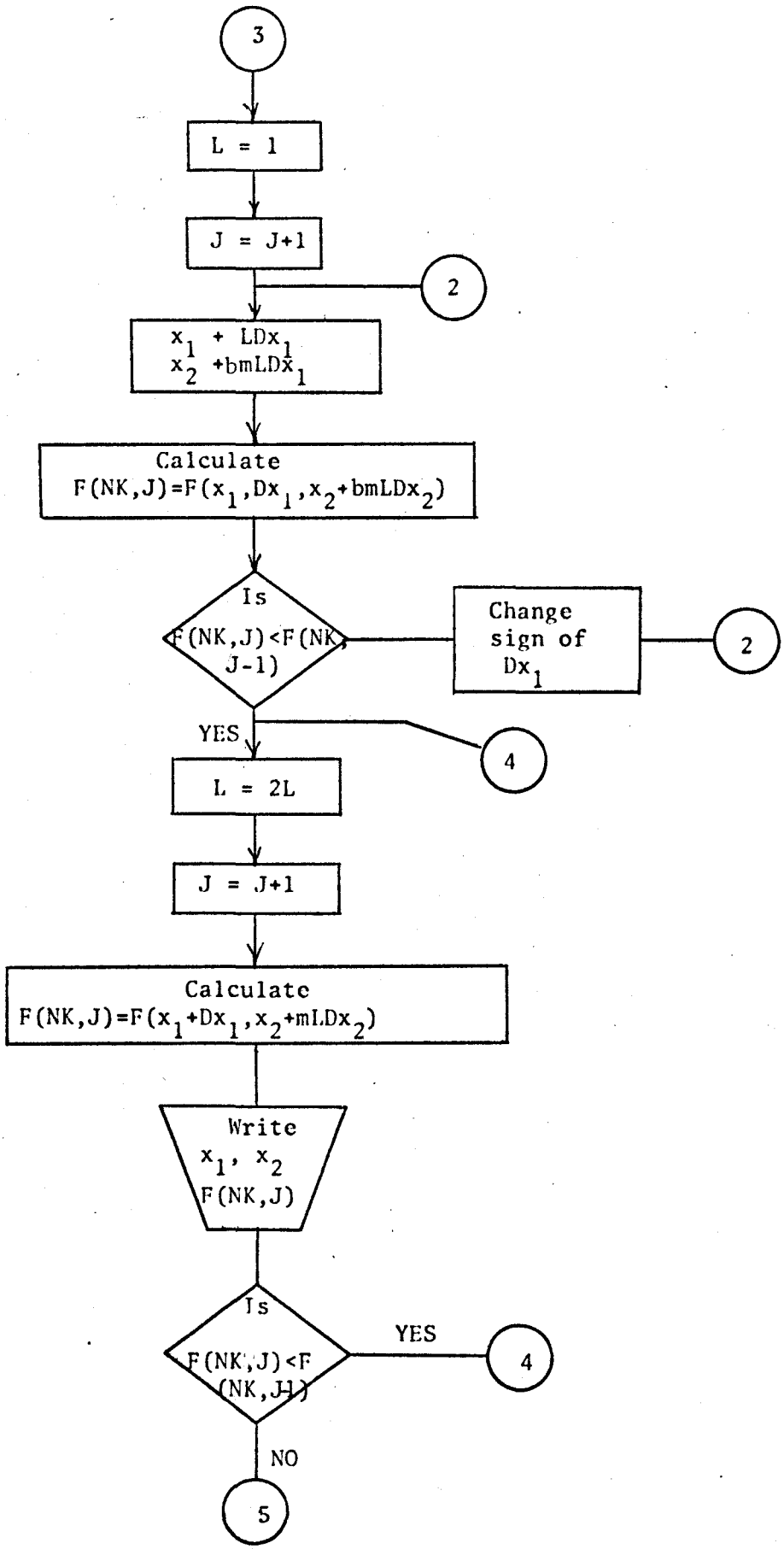


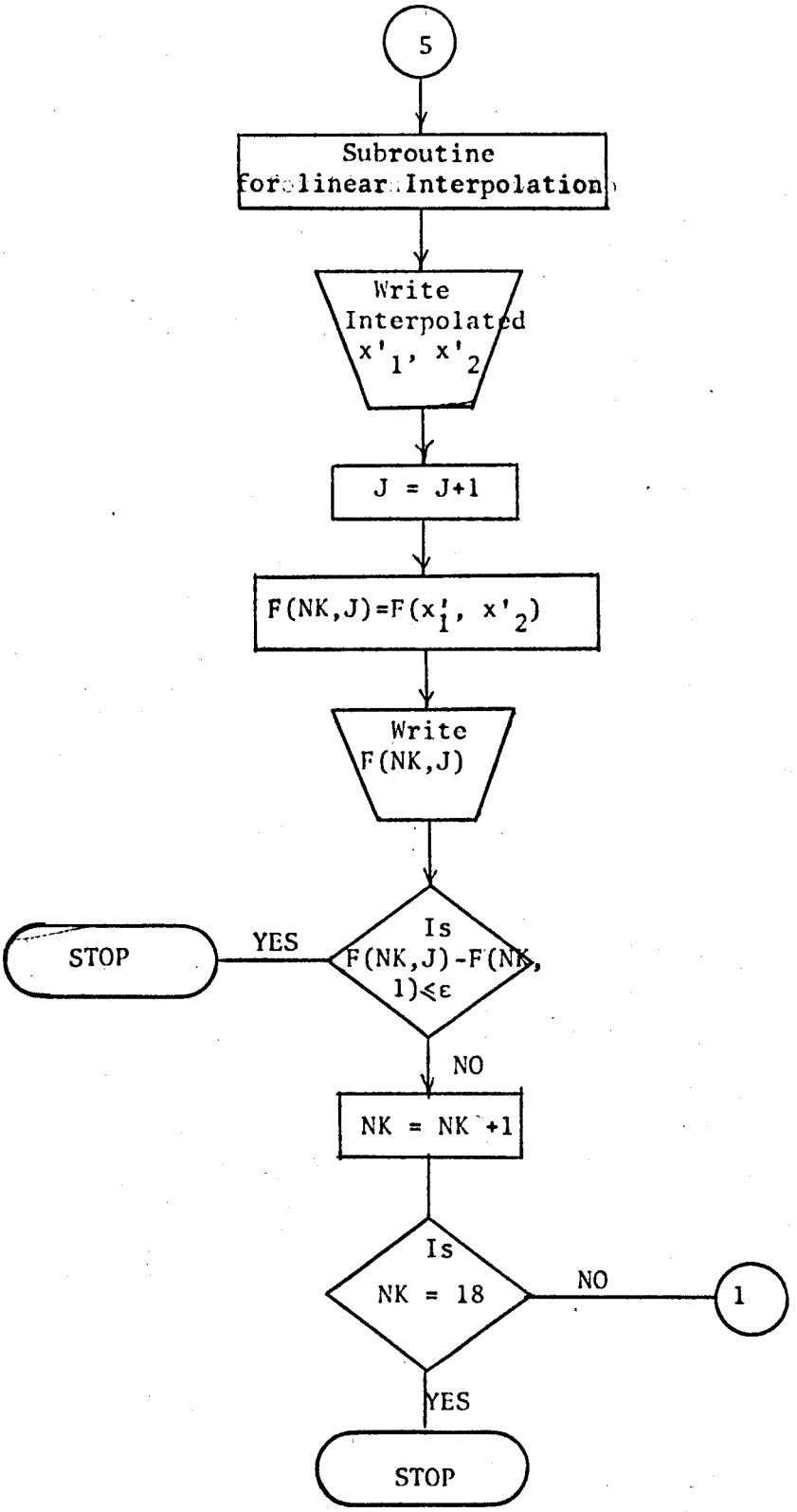
For continued and Scale Invariant Partan the m's differ.

APPENDIX II

Flow Chart for the Method of Conjugate Gradients and Steepest Descent




3
L = 1
J = J+1
2
x_1 + LDx_1
x_2 +bmLDx_1
Calculate
$F(NK,J)=F(x_1,Dx_1,x_2+bmLDx_2)$
Is
$F(NK,J)<F(NK,J-1)$
Change sign of Dx_1
2
YES
4
L = 2L
J = J+1
Calculate
$F(NK,J)=F(x_1+Dx_1,x_2+mLDx_2)$
Write
x_1, x_2
F(NK,J)
Is
$F(NK,J)<F(NK,J1)$
YES
4
NO
5


5
Subroutine for linear Interpolation
Write Interpolated x'_1, x'_2
J = J+1
F(NK,J)=F(x'_1, x'_2)
Write F(NK,J)
Is F(NK,J)-F(NK, 1)≤ε
YES
STOP
NO
NK = NK +1
Is NK = 18
NO
1
YES
STOP

APPENDIX III

```
$JOB    WATFOR  003508 IQLEEM            060    010
$IBJOB          NODECK
$IBFTC
C       METHOD OF CONJUGATE GRADIENTS APPLIED TO 'BEALE,S FUNCTION'
        DIMENSION C(3),U(3),F(20),DF(2,40),XS(2,40,20),X(2,40,20)
        DIMENSION AM(40),DPHI(20),BM(40),BETA(40)
        READ (5,1) (C(I),I=1,3)
     1  FORMAT (3F10.4)
        DX=0.0001
        NK=1
        READ (5,1) (XS(M,NK,1),M=1,2)
    19  WRITE (6,2) XS(1,NK,1),XS(2,NK,1)
     2  FORMAT (1H0,5X,F15.8,5X,F15.8)
        J=1
        DO 3 M=1,2
     3  X(M,NK,J)=XS(M,NK,J)
        U(1)=C(1)-X(1,NK,J)*(1.0-X(2,NK,J))
        U(2)=C(2)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J))
        U(3)=C(3)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J)*X(2,NK,J))
        WRITE (6,4) U(1),U(2),U(3)
     4  FORMAT (1H0,5X,F15.8,5X,F15.8,5X,F15.8)
        F(J)=U(1)*U(1)+U(2)*U(2)+U(3)*U(3)
        WRITE (6,5) F(J)
     5  FORMAT (1H0,40X,F15.8)
        DF(1,NK)=-2.0*(U(1)*(1.0-X(2,NK,J))+U(2)*(1.0-X(2,NK,J)*X(2,NK,J))
       1+U(3)*(1.0-X(2,NK,J)*X(2,NK,J)*X(2,NK,J)))
        DF(2,NK)=2.0*(U(1)*X(1,NK,J)+U(2)*2.0*X(1,NK,J)*X(2,NK,J)
       1+U(3)*3.0*X(1,NK,J)*X(2,NK,J)*X(2,NK,J))
        WRITE (6,6) (DF(I,NK),I=1,2)
     6  FORMAT (1H0,1X,2F15.8)
        AM(NK)=DF(2,NK)/DF(1,NK)
        WRITE (6,7) AM(NK)
     7  FORMAT (1H0,10X,F20.9)
        IF (NK.EQ.1) BM(NK)=AM(NK)
        IF (NK.EQ.1) GO TO 17
        BETA(NK)=(AM(NK)*AM(NK))/(AM(NK-1)*AM(NK-1))
        WRITE (6,18) BETA(NK)
    18  FORMAT (1H0,20X,F20.9)
        BM(NK)=AM(NK)+BETA(NK)*AM(NK-1)
        WRITE (6,7) BM(NK)
    17  FACTR=1.0
        J=J+1
        L=1
        IC=J
        X(1,NK,J)=XS(1,NK,1)+FACTR*FLOAT(L)*DX
        X(2,NK,J)=XS(2,NK,1)+FACTR*FLOAT(L)*BM(NK)*DX*0.01
        U(1)=C(1)-X(1,NK,J)*(1.0-X(2,NK,J))
        U(2)=C(2)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J))
        U(3)=C(3)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J)*X(2,NK,J))
        F(J)=U(1)*U(1)+U(2)*U(2)+U(3)*U(3)
        IF ((F(J)-F(J-1)).GT.0.0) FACTR=-1.0
```

```
      IF ((F(J)-F(J-1)).LT.0.0) FACTR=1.0
      X(1,NK,J)=XS(1,NK,1)+FACTR*FLOAT(L)*DX
      X(2,NK,J)=XS(2,NK,1)+FACTR*FLOAT(L)*BM(NK)*DX*0.01
   15 WRITE (6,8) (X(M,NK,J),M=1,2),IC
    8 FORMAT (1H0,5X,F15.8,5X,F15.8,I5)
      U(1)=C(1)-X(1,NK,J)*(1.0-X(2,NK,J))
      U(2)=C(2)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J))
      U(3)=C(3)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J)*X(2,NK,J))
      F(J)=U(1)*U(1)+U(2)*U(2)+U(3)*U(3)
      WRITE (6,5) F(J)
      DPHI(J)=F(J)-F(J-1)
      IF (DPHI(J).GT.0.0) GO TO 9
      L=2*L
      J=J+1
      IC=IC+1
      X(1,NK,J)=X(1,NK,J-1)+FACTR*FLOAT(L)*DX
      X(2,NK,J)=X(2,NK,J-1)+FACTR*FLOAT(L)*BM(NK)*DX*0.01
      GO TO 15
    9 IF (IC.GE.3) GO TO 11
      IF (IC.EQ.2) GO TO 12
   11 J=J+1
      NK=NK+1
      DX=0.0001
      X(1,NK,J)=(X(1,NK-1,J-1)+X(1,NK-1,J-2))/2.0
      X(2,NK,J)=(X(2,NK-1,J-1)+X(2,NK-1,J-2))/2.0
      U(1)=C(1)-X(1,NK,J)*(1.0-X(2,NK,J))
      U(2)=C(2)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J))
      U(3)=C(3)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J)*X(2,NK,J))
      F(J)=U(1)*U(1)+U(2)*U(2)+U(3)*U(3)
      IF (F(J).LT.F(J-2)) GO TO 20
   21 X(1,NK,J)=X(1,NK-1,J-2)
      X(2,NK,J)=X(2,NK-1,J-2)
      GO TO 20
   12 J=J+1
      NK=NK+1
      DX=0.00001
      X(1,NK,J)=X(1,NK-1,J-2)
      X(2,NK,J)=X(2,NK-1,J-2)
   20 IF (NK.EQ.40) GO TO 16
      IC=1
      XS(1,NK,1)=X(1,NK,J)
      XS(2,NK,1)=X(2,NK,J)
      J=1
      GO TO 19
   16 STOP
      END
$ENTRY
  1.5        2.25       2.625
```

```
5.0          0.2
$IBSYS
```

The method of Steepest Descent is obtained by simply putting BETA = 0.0 in this program.

APPENDIX IV

```
$JOB   WATFOR  003508 IQLEEM            060   010
$IBJOB         NODECK
$IBFTC
C      METHOD OF PARALLEL TANGENTS APPLIED TO 'BEALE'S FUNCTION '
C      ITERATED STEEPEST DESCENT PARTAN
       DIMENSION C(3),U(3),F(30),DF(2,30),XS(2,20,30),X(2,20,30)
       DIMENSION AM(20),DPHI(30)
       READ (5,1) (C(I),I=1,3)
     1 FORMAT (3F10.4)
       DX=0.0001
       NK=1
       READ (5,1) (XS(M,NK,1),M=1,2)
       WRITE (6,2) XS(1,NK,1),XS(2,NK,1)
     2 FORMAT (1H0,5X,F15.8,5X,F15.8)
       J=1
       DO 3 M=1,2
     3 X(M,NK,J)=XS(M,NK,J)
    17 U(1)=C(1)-X(1,NK,J)*(1.0-X(2,NK,J))
       U(2)=C(2)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J))
       U(3)=C(3)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J)*X(2,NK,J))
       WRITE (6,4) U(1),U(2),U(3)
     4 FORMAT (1H0,5X,F15.8,5X,F15.8,5X,F15.8)
       F(J)=U(1)*U(1)+U(2)*U(2)+U(3)*U(3)
       WRITE (6,5) F(J)
     5 FORMAT (1H0,40X,F15.8)
       DF(1,NK)=-2.0*(U(1)*(1.0-X(2,NK,J))+U(2)*(1.0-X(2,NK,J)*X(2,NK,J))
      1+U(3)*(1.0-X(2,NK,J)*X(2,NK,J)*X(2,NK,J)))
       DF(2,NK)=2.0*(U(1)*X(1,NK,J)+U(2)*2.0*X(1,NK,J)*X(2,NK,J)
      1+U(3)*3.0*X(1,NK,J)*X(2,NK,J)*X(2,NK,J))
       WRITE (6,6) (DF(I,NK),I=1,2)
     6 FORMAT (1H0,1X,2F15.8)
       AM(NK)=DF(2,NK)/DF(1,NK)
    18 WRITE (6,7) AM(NK)
     7 FORMAT (1H0,10X,F20.9)
       FACTR=1.0
       J=J+1
       L=1
       IC=J
       X(1,NK,J)=XS(1,NK,1)+FACTR*FLOAT(L)*DX
       X(2,NK,J)=XS(2,NK,1)+FACTR*FLOAT(L)*AM(NK)*DX*0.1
       U(1)=C(1)-X(1,NK,J)*(1.0-X(2,NK,J))
       U(2)=C(2)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J))
       U(3)=C(3)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J)*X(2,NK,J))
       F(J)=U(1)*U(1)+U(2)*U(2)+U(3)*U(3)
       IF ((F(J)-F(J-1)).GT.0.0) FACTR=-1.0
       IF ((F(J)-F(J-1)).LT.0.0) FACTR=1.0
       X(1,NK,J)=XS(1,NK,1)+FACTR*FLOAT(L)*DX
       X(2,NK,J)=XS(2,NK,1)+FACTR*FLOAT(L)*AM(NK)*DX*0.1
    15 WRITE (6,8) (X(M,NK,J),M=1,2),IC
     8 FORMAT (1H0,5X,F15.8,5X,F15.8,I5)
       U(1)=C(1)-X(1,NK,J)*(1.0-X(2,NK,J))
```

```
      U(2)=C(2)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J))
      U(3)=C(3)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J)*X(2,NK,J))
      F(J)=U(1)*U(1)+U(2)*U(2)+U(3)*U(3)
      WRITE (6,5) F(J)
      DPHI(J)=F(J)-F(J-1)
      IF (DPHI(J).GT.0.0) GO TO 9
      L=2*L
      J=J+1
      IC=IC+1
       X(1,NK,J)=X(1,NK,J-1)+FACTR*FLOAT(L)*DX
      X(2,NK,J)=X(2,NK,J-1)+FACTR*FLOAT(L)*AM(NK)*DX*0.1
      GO TO 15
    9 IF (IC.GE.3) GO TO 11
      IF (IC.EQ.2) GO TO 12
   11 J=J+1
      NK=NK+1
      DX=0.0001
      X(1,NK,J)=(X(1,NK-1,J-1)+X(1,NK-1,J-2))/2.0
      X(2,NK,J)=(X(2,NK-1,J-1)+X(2,NK-1,J-2))/2.0
      U(1)=C(1)-X(1,NK,J)*(1.0-X(2,NK,J))
      U(2)=C(2)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J))
      U(3)=C(3)-X(1,NK,J)*(1.0-X(2,NK,J)*X(2,NK,J)*X(2,NK,J))
      F(J)=U(1)*U(1)+U(2)*U(2)+U(3)*U(3)
      IF (F(J).LT.F(J-2)) GO TO 19
      X(1,NK,J)=X(1,NK-1,J-2)
      X(2,NK,J)=X(2,NK-1,J-2)
      GO TO 19
   12 J=J+1
      NK=NK+1
      DX=0.00001
      X(1,NK,J)=X(1,NK-1,J-2)
      X(2,NK,J)=X(2,NK-1,J-2)
   19 IC=1
      IF (NK.EQ.2) AM(2)=-1.0/AM(1)
      IF (NK.EQ.3) AM(3)=    ((X(2,3,J)-X(2,1,1))/(X(1,3,J)-X(1,1,1)))
      IF (NK.EQ.4) AM(4)=-1.0/AM(3)
      IF (NK.EQ.5) AM(5)=AM(3)
      IF (NK.EQ.6) AM(6)=((X(2,6,J)-X(2,4,1))/(X(1,6,J)-X(1,4,1)))
      IF (NK.EQ.7) AM(7)=-1.0/AM(6)
      IF (NK.EQ.8) AM(8)=AM(6)
      IF (NK.EQ.9) AM(9)=((X(2,9,J)-X(2,6,1))/(X(1,9,J)-X(1,6,1)))
      IF (NK.EQ.10) AM(10)=-1.0/AM(9)
      IF (NK.EQ.11) AM(11)=AM(9)
      IF (NK.EQ.12) AM(12)=((X(2,12,J)-X(2,10,1))/(X(1,12,J)-X(1,10,1)))
      IF (NK.EQ.13) AM(13)=-1.0/AM(12)
      IF (NK.EQ.14) AM(14)=AM(12)
      IF (NK.EQ.15) AM(15)=((X(2,15,J)-X(2,12,1))/(X(1,15,J)-X(1,12,1)))
      IF (NK.EQ.16) AM(16)=-1.0/AM(15)
      IF (NK.EQ.17) AM(17)=AM(15)
      IF (NK.EQ.18) GO TO 16
      XS(1,NK,1)=X(1,NK,J)
      XS(2,NK,1)=X(2,NK,J)
```

```
      WRITE (6,2) XS(1,NK,1),XS(2,NK,1)
      J=1
      X(1,NK,J)=XS(1,NK,1)
      X(2,NK,J)=XS(2,NK,1)
      GO TO 17
   16 STOP
      END
$ENTRY
  1.5         2.25        2.625
  2.0         0.2
$IBSYS
```

Continued and iterated scale invariant partan programs were made on a similar basis.

```
$JOB            003508 IQLEEM            060    010
$IBJOB
$IBFTC   FITS
C      ADAPTIVE CONTROL WITHOUT IDENTIFICATION
C      METHOD OF CONJUGATE GRADIENTS
C      PLANT PARAMETERS FIXED-ADJUSTABLE CONTROLLER WITH TWO DEGREE
C      OF FREEDOM
       DIMENSION WS(2,50,15),WI(2,50,15),DK(2),A(6),YI(7)
       DIMENSION DF(2,15),S(4,15,100),YSOLN(7,100),DPHI(15),F(7,15)
       DIMENSION AM(50),BETA(50),BM(50)
       EXTERNAL DEQSET
       COMMON A
       DW=0.0001
       NK=1
       READ (5,1) (WS(I,NK,1),I=1,2)
     1 FORMAT (2F10.4)
       READ (5,1) (DK(I),I=1,2)
   260 WRITE (6,8) WS(1,NK,1),WS(2,NK,1)
     8 FORMAT (1H0,5X,F15.8,5X,F15.8)
       J=1
       DO 110 M=1,2
   110 WI(M,NK,J)=WS(M,NK,J)
       DO 120 N=1,3
       A(1)=8.55+WI(2,NK,J)
       A(2)=404.46+8.55*WI(2,NK,J)
       A(3)=220.48+404.46*WI(2,NK,J)
       A(4)=24.0+220.48*WI(2,NK,J)+900.0*WI(2,NK,J)/WI(1,NK,J)
       A(5)=924.0*WI(2,NK,J)+720.0*WI(2,NK,J)/WI(1,NK,J)
       A(6)=720.0*WI(2,NK,J)
       WRITE (6,2) (A(I),I=1,6)
     2 FORMAT (1X,6E12.5/)
C      THE  5TH  ORDER CONTROL SYSTEM IS REPRESENTED BY A SET OF FIVE
C      SIMULTANEOUS FIRST ORDER DIFFERTIAL EQUATIONS AND THE SOLUTION IS
C      OBTAINED BY USING A SUBROUTINE
C      INITIAL CONDITIONS
       YI(1)=0.0
       YI(2)=0.0
       YI(3)=0.0
       YI(4)=0.0
       YI(5)=0.0
       YI(6)=900.0*WI(2,NK,J)/WI(1,NK,J)
       YI(7)=900.0*(WI(2,NK,J)/WI(1,NK,J))*(WI(1,NK,J)-WI(2,NK,J)-7.75)
       CALL RUNGE(DEQSET,7,100,0.024,YI,YSOLN)
       DO 130 KK=1,100
   130 S(N,J,KK)=YSOLN(1,KK)*(1.0-YSOLN(2,KK))*(1.0-YSOLN(2,KK))
       F(N,J)=0.0
       DELT=0.024
       DO 140 KK=1,99
       MM=KK+1
   140 F(N,J)=F(N,J)+S(N,J,KK)*DELT+0.5*DELT*(S(N,J,MM)-S(N,J,KK))
C      EVALUATION OF PARTIAL DERIVATIVES BY THE LOCAL EXAMINATION OF
```

```
C       THE RESPONSE SURFACE
        IF (N.GE.2) GO TO 210
        WI(1,NK,J)=WI(1,NK,J)+DK(1)
        GO TO 120
  210   WI(1,NK,J)=WS(1,NK,1)
        IF (N.GE.3) GO TO 220
        WI(2,NK,J)=WI(2,NK,J)+DK(2)
        GO TO 120
  220   WI(2,NK,J)=WS(2,NK,1)
        DO 150 NN=2,3
        M=NN-1
        DF(M,J)=(F(NN,J)-F(1,J))/DK(M)
  150   CONTINUE
        AM(NK)=DF(2,J)/DF(1,J)
  120   CONTINUE
        WRITE (6,3) F(1,J),F(2,J),F(3,J)
    3   FORMAT (1H0,5X,F15.8,5X,F15.8,5X,F15.8)
        WRITE (6,4) (DF(M,J),M=1,2)
    4   FORMAT (1H0,1X,2F15.8)
        WRITE (6,5) AM(NK)
    5   FORMAT (1H0,10X,F20.9)
        IF (NK.EQ.1) BM(NK)=AM(NK)
        IF (NK.EQ.1) GO TO 270
        BETA(NK)=AM(NK)*AM(NK)/AM(NK-1)*AM(NK-1)
        WRITE (6,6) BETA(NK)
    6   FORMAT (1H0,15X,F20.9)
        BM(NK)=AM(NK)+BETA(NK)*AM(NK-1)
        WRITE (6,5) BM(NK)
  270   F(4,1)=F(1,1)
        FACTR=1.0
        J=J+1
        N=4
        L=1
        IC=1
        WI(1,NK,J)=WS(1,NK,1)+FACTR*FLOAT(L)*DW
        WI(2,NK,J)=WS(2,NK,1)+FACTR*BM(NK)*FLOAT(L)*DW*50.0
        A(1)=8.55+WI(2,NK,J)
        A(2)=404.46+8.55*WI(2,NK,J)
        A(3)=220.48+404.46*WI(2,NK,J)
        A(4)=24.0+220.48*WI(2,NK,J)+900.0*WI(2,NK,J)/WI(1,NK,J)
        A(5)=924.0*WI(2,NK,J)+720.0*WI(2,NK,J)/WI(1,NK,J)
        A(6)=720.0*WI(2,NK,J)
C       SOLUTION OF SYSTEM DIFFERENTIAL EQUATION IS OBTAINED AS BEFORE
C       INITIAL CONDITIONS
        YI(1)=0.0
        YI(2)=0.0
        YI(3)=0.0
        YI(4)=0.0
        YI(5)=0.0
        YI(6)=900.0*WI(2,NK,J)/WI(1,NK,J)
        YI(7)=900.0*(WI(2,NK,J)/WI(1,NK,J))*(WI(1,NK,J)-WI(2,NK,J)-7.75)
        CALL RUNGE(DEQSET,7,100,0.024,YI,YSOLN)
```

```
      DO 160 KK=1,100
  160 S(N,J,KK)=YSOLN(1,KK)*(1.0-YSOLN(2,KK))*(1.0-YSOLN(2,KK))
      F(N,J)=0.0
      DELT=0.024
      DO 170 KK=1,99
      LL=KK+1
  170 F(N,J)=F(N,J)+S(N,J,KK)*DELT+0.5*DELT*(S(N,J,LL)-S(N,J,KK))
      IF ((F(4,J)-F(4,J-1)).GT.0.0) FACTR=-1.0
      IF ((F(4,J)-F(4,J-1)).LT.0.0) FACTR=1.0
      WI(1,NK,J)=WS(1,NK,1)+FACTR*FLOAT(L)*DW
      WI(2,NK,J)=WS(2,NK,1)+FACTR*BM(NK)*FLOAT(L)*DW*50.0
  240 WRITE (6,7) (WI(M,NK,J),M=1,2),IC
    7 FORMAT (1H0,5X,F15.8,5X,F15.8,I5)
      A(1)=8.55+WI(2,NK,J)
      A(2)=404.46+8.55*WI(2,NK,J)
      A(3)=220.48+404.46*WI(2,NK,J)
      A(4)=24.0+220.48*WI(2,NK,J)+900.0*WI(2,NK,J)/WI(1,NK,J)
      A(5)=924.0*WI(2,NK,J)+720.0*WI(2,NK,J)/WI(1,NK,J)
      A(6)=720.0*WI(2,NK,J)
C     SOLUTION OF SYSTEM DIFFERENTIAL EQUATION IS OBTAINED AS BEFORE
C     INITIAL CONDITIONS
      YI(1)=0.0
      YI(2)=0.0
      YI(3)=0.0
      YI(4)=0.0
      YI(5)=0.0
      YI(6)=900.0*WI(2,NK,J)/WI(1,NK,J)
      YI(7)=900.0*(WI(2,NK,J)/WI(1,NK,J))*(WI(1,NK,J)-WI(2,NK,J)-7.75)
      CALL RUNGE(DEQSET,7,100,0.024,YI,YSOLN)
      DO 180 KK=1,100
  180 S(N,J,KK)=YSOLN(1,KK)*(1.0-YSOLN(2,KK))*(1.0-YSOLN(2,KK))
      F(N,J)=0.0
      DELT=0.024
      DO 190 KK=1,99
      LL=KK+1
  190 F(N,J)=F(N,J)+S(N,J,KK)*DELT+0.5*DELT*(S(N,J,LL)-S(N,J,KK))
      WRITE (6,9) F(N,J)
    9 FORMAT (1H0,40X,F15.8)
      DPHI(J)=F(4,J)-F(4,J-1)
      IF (DPHI(J).GT.0.0) GO TO 230
      J=J+1
      L=2*L
      IC=IC+1
      WI(1,NK,J)=WI(1,NK,J-1)+FACTR*FLOAT(L)*DW
      WI(2,NK,J)=WI(2,NK,J-1)+FACTR*BM(NK)*FLOAT(L)*DW*50.0
      GO TO 240
  230 J=J+1
      NK=NK+1
      DW=0.0001
      WI(1,NK,J)=(WI(1,NK-1,J-1)+WI(1,NK-1,J-2))/2.0
      WI(2,NK,J)=(WI(2,NK-1,J-1)+WI(2,NK-1,J-2))/2.0
      A(1)=8.55+WI(2,NK,J)
```

```
      A(2)=404.46+8.55*WI(2,NK,J)
      A(3)=220.48+404.46*WI(2,NK,J)
      A(4)=24.0+220.48*WI(2,NK,J)+900.0*WI(2,NK,J)/WI(1,NK,J)
      A(5)=924.0*WI(2,NK,J)+720.0*WI(2,NK,J)/WI(1,NK,J)
      A(6)=720.0*WI(2,NK,J)
C     SOLUTION OF SYSTEM DIFFERENTIAL EQUATION IS OBTAINED AS BEFORE
C     INITIAL CONDITIONS
      YI(1)=0.0
      YI(2)=0.0
      YI(3)=0.0
      YI(4)=0.0
      YI(5)=0.0
      YI(6)=900.0*WI(2,NK,J)/WI(1,NK,J)
      YI(7)=900.0*(WI(2,NK,J)/WI(1,NK,J))*(WI(1,NK,J)-WI(2,NK,J)-7.75)
      CALL RUNGE(DEQSET,7,100,0.024,YI,YSOLN)
      DO 300 KK=1,100
  300 S(N,J,KK)=YSOLN(1,KK)*(1.0-YSOLN(2,KK))*(1.0-YSOLN(2,KK))
      F(N,J)=0.0
      DELT=0.024
      DO 310 KK=1,99
      MM=KK+1
  310 F(N,J)=F(N,J)+S(N,J,KK)*DELT+0.5*DELT*(S(N,J,MM)-S(N,J,KK))
      IF (F(4,J).LT.F(4,J-2)) GO TO 400
      WI(1,NK,J)=WI(1,NK-1,J-2)
      WI(2,NK,J)=WI(2,NK-1,J-2)
      DW=0.00001
  400 IF (NK.EQ.30) GO TO 250
      IC=1
      WS(1,NK,1)=WI(1,NK,J)
      WS(2,NK,1)=WI(2,NK,J)
      GO TO 260
  250 CALL EXIT
      END
$IBFTC QRUNGE
      SUBROUTINE  RUNGE(DEQSET,NDEQ,NVAL,H,YI,YSOLN)
C
C  RUNGE-KUTTA SOLUTION TO A SYSTEM OF SIMULTANEOUS 1ST-ORDER DIFF-EQNS.
C  NDEQ, THE NUMBER OF EQUATIONS, MUST NOT EXCEED 10
C  NVAL, THE NUMBER OF POINTS AT WHICH SOLUTION IS DESIRED, INCLUDES X0.
C  H IS THE STEP LENGTH FOR THE INDEPENDENT VARIABLE.
C  YI IS THE ARRAY OF INITIAL VALUES OF THE FUNCTIONS.
C  YSOLN, DIMENSIONED (NDEQ,NVAL), CONTAINS THE SOLUTIONS IN THE ORDER
C        IN WHICH THE DEPENDENT VARIABLES ARE DEFINED IN SUBRTN-'DEQSET'
C
      DIMENSION  YI(7),YSOLN(7,100),A(6)
      DIMENSION  YY(10), YDOT(10), YDEL(10)
      COMMON A
      DO 1 I=1,NDEQ
    1 YSOLN(I,1) = YI(I)
      DO 2 J=2,NVAL
      DO 3 I=1,NDEQ
    3 YY(I) = YSOLN(I,J-1)
```

```
      CALL  DEQSET(YDOT,YY)
      DO 4 I=1,NDEQ
    4 YDEL(I) = YDOT(I)
      DO 5 KTIMES=1,2
      DO 6 I=1,NDEQ
    6 YY(I) = YSOLN(I,J-1) + YDOT(I)*H/2.
      CALL  DEQSET(YDOT,YY)
      DO 7 I=1,NDEQ
    7 YDEL(I) = YDEL(I) + 2.*YDOT(I)
  5   CONTINUE
      DO 8 I=1,NDEQ
    8 YY(I) = YSOLN(I,J-1) + YDOT(I)*H
      CALL  DEQSET(YDOT,YY)
      DO 9 I=1,NDEQ
    9 YSOLN(I,J) = YSOLN(I,J-1) + (YDEL(I)+YDOT(I))*H/6.
    2 CONTINUE
      RETURN
      END
$IBFTC  FUNC1
      SUBROUTINE DEQSET(YDOT,YY)
      DIMENSION YY(7),YDOT(7),A(6)
      COMMON A
C     EQUATION FOR THE INDEPENDENT VARIABLE, INITIALLY ZERO
      YDOT(1)=1.0
C     SYSTEM EQUATIONS
      YDOT(2)=YY(3)
      YDOT(3)=YY(4)
      YDOT(4)=YY(5)
      YDOT(5)=YY(6)
      YDOT(6)=YY(7)
      YDOT(7)=A(6)-A(6)*YY(2)-A(5)*YY(3)-A(4)*YY(4)-A(3)*YY(5)-
     1A(2)*YY(6)-A(1)*YY(7)
      RETURN
      END
$ENTRY
  0.31       20.0
  0.0033     0.18
$IBSYS
```

The steepest descent program is obtained by putting

```
BETA  = 0.0
```

APPENDIX VI

```
$JOB            003508 IQLEEM            060    010
$IBJOB
$IBFTC   FITS
C       ADAPTIVE CONTROL WITHOUT IDENTIFICATION
C       METHOD OF PARALLEL TANGENTS (PARTAN)
C       ITERATED STEEPEST DESCENT PARTAN
C       PLANT PARAMETERS FIXED-ADJUSTABLE CONTROLLER WITH TWO DEGREE
C       OF FREEDOM
        DIMENSION WS(2,20,20),WI(2,20,20),DK(2),A(6),YI(7),AM(20)
        DIMENSION DF(2,20),S(4,18,100),YSOLN(7,100),DPHI(20),F(7,20)
        EXTERNAL DEQSET
        COMMON A
        WRITE (6,700)
  700   FORMAT (10X,32HITERATED STEEPEST DESCENT PARTAN//)
        DW=0.0001
        NK=1
        READ (5,1) (WS(I,NK,1),I=1,2)
    1   FORMAT (2F10.4)
        READ (5,1) (DK(I),I=1,2)
  260   WRITE (6,8) WS(1,NK,1),WS(2,NK,1)
    8   FORMAT (1H0,5X,F15.8,5X,F15.8)
        J=1
        DO 110 M=1,2
  110   WI(M,NK,J)=WS(M,NK,J)
        DO 120 N=1,3
        A(1)=8.55+WI(2,NK,J)
        A(2)=404.46+8.55*WI(2,NK,J)
        A(3)=220.48+404.46*WI(2,NK,J)
        A(4)=24.0+220.48*WI(2,NK,J)+900.0*WI(2,NK,J)/WI(1,NK,J)
        A(5)=924.0*WI(2,NK,J)+720.0*WI(2,NK,J)/WI(1,NK,J)
        A(6)=720.0*WI(2,NK,J)
        WRITE (6,2) (A(I),I=1,6)
    2   FORMAT (1X,6E12.5/)
C       THE  5TH  ORDER CONTROL SYSTEM IS REPRESENTED BY A SET OF FIVE
C       SIMULTANEOUS FIRST ORDER DIFFERTIAL EQUATIONS AND THE SOLUTION IS
C       OBTAINED BY USING A SUBROUTINE
C       INITIAL CONDITIONS
        YI(1)=0.0
        YI(2)=0.0
        YI(3)=0.0
        YI(4)=0.0
        YI(5)=0.0
        YI(6)=900.0*WI(2,NK,J)/WI(1,NK,J)
        YI(7)=900.0*(WI(2,NK,J)/WI(1,NK,J))*(WI(1,NK,J)-WI(2,NK,J)-7.75)
        CALL RUNGE(DEQSET,7,100,0.024,YI,YSOLN)
        DO 130 KK=1,100
  130   S(N,J,KK)=YSOLN(1,KK)*(1.0-YSOLN(2,KK))*(1.0-YSOLN(2,KK))
        F(N,J)=0.0
        DELT=0.024
        DO 140 KK=1,99
        MM=KK+1
```

```
  140 F(N,J)=F(N,J)+S(N,J,KK)*DELT+0.5*DELT*(S(N,J,MM)-S(N,J,KK))
      IF (NK.GE.2) GO TO 270
C     EVALUATION OF PARTIAL DERIVATIVES BY THE LOCAL EXAMINATION OF
C     THE RESPONSE SURFACE
C     THE PARTIAL DERIVATIVES IN THE METHOD OF PARTAN ARE DETERMINED
C     ONLY AT THE STARTING POINT AND SLOPE CALCULATED
      IF (N.GE.2) GO TO 210
      WI(1,NK,J)=WI(1,NK,J)+DK(1)
      GO TO 120
  210 WI(1,NK,J)=WS(1,NK,J)
      IF (N.GE.3) GO TO 220
      WI(2,NK,J)=WI(2,NK,J)+DK(2)
      GO TO 120
  220 WI(2,NK,J)=WS(2,NK,J)
      DO 150 NN=2,3
      M=NN-1
      DF(M,J)=(F(NN,J)-F(1,J))/DK(M)
  150 CONTINUE
      AM(NK)=DF(2,J)/DF(1,J)
  120 CONTINUE
      WRITE (6,4) (DF(M,J),M=1,2)
    4 FORMAT (1H0,1X,2F15.8)
  270 WRITE (6,3) F(1,J),F(2,J),F(3,J)
    3 FORMAT (1H0,5X,F15.8,5X,F15.8,5X,F15.8)
      WRITE (6,5) AM(NK)
    5 FORMAT (1H0,10X,F20.9)
      F(4,1)=F(1,1)
      FACTR=1.0
      J=J+1
      N=4
      L=1
      IC=1
      WI(1,NK,J)=WS(1,NK,1)+FACTR*FLOAT(L)*DW
      WI(2,NK,J)=WS(2,NK,1)+FACTR*AM(NK)*FLOAT(L)*DW*50.0
      A(1)=8.55+WI(2,NK,J)
      A(2)=404.46+8.55*WI(2,NK,J)
      A(3)=220.48+404.46*WI(2,NK,J)
      A(4)=24.0+220.48*WI(2,NK,J)+900.0*WI(2,NK,J)/WI(1,NK,J)
      A(5)=924.0*WI(2,NK,J)+720.0*WI(2,NK,J)/WI(1,NK,J)
      A(6)=720.0*WI(2,NK,J)
C     SOLUTION OF SYSTEM DIFFERENTIAL EQUATION IS OBTAINED AS BEFORE
C     INITIAL CONDITIONS
      YI(1)=0.0
      YI(2)=0.0
      YI(3)=0.0
      YI(4)=0.0
      YI(5)=0.0
      YI(6)=900.0*WI(2,NK,J)/WI(1,NK,J)
      YI(7)=900.0*(WI(2,NK,J)/WI(1,NK,J))*(WI(1,NK,J)-WI(2,NK,J)-7.75)
      CALL RUNGE(DEQSET,7,100,0.024,YI,YSOLN)
      DO 160 KK=1,100
  160 S(N,J,KK)=YSOLN(1,KK)*(1.0-YSOLN(2,KK))*(1.0-YSOLN(2,KK))
```

```
      F(N,J)=0.0
      DELT=0.024
      DO 170 KK=1,99
      LL=KK+1
  170 F(N,J)=F(N,J)+S(N,J,KK)*DELT+0.5*DELT*(S(N,J,LL)-S(N,J,KK))
      IF ((F(4,J)-F(4,J-1)).GT.0.0) FACTR=-1.0
      IF ((F(4,J)-F(4,J-1)).LT.0.0) FACTR=1.0
      WI(1,NK,J)=WS(1,NK,1)+FACTR*FLOAT(L)*DW
      WI(2,NK,J)=WS(2,NK,1)+FACTR*AM(NK)*FLOAT(L)*DW*50.0
  240 WRITE (6,7) (WI(M,NK,J),M=1,2),IC
    7 FORMAT (1H0,5X,F15.8,5X,F15.8,I5)
      A(1)=8.55+WI(2,NK,J)
      A(2)=404.46+8.55*WI(2,NK,J)
      A(3)=220.48+404.46*WI(2,NK,J)
      A(4)=24.0+220.48*WI(2,NK,J)+900.0*WI(2,NK,J)/WI(1,NK,J)
      A(5)=924.0*WI(2,NK,J)+720.0*WI(2,NK,J)/WI(1,NK,J)
      A(6)=720.0*WI(2,NK,J)
C     SOLUTION OF SYSTEM DIFFERENTIAL EQUATION IS OBTAINED AS BEFORE
C     INITIAL CONDITIONS
      YI(1)=0.0
      YI(2)=0.0
      YI(3)=0.0
      YI(4)=0.0
      YI(5)=0.0
      YI(6)=900.0*WI(2,NK,J)/WI(1,NK,J)
      YI(7)=900.0*(WI(2,NK,J)/WI(1,NK,J))*(WI(1,NK,J)-WI(2,NK,J)-7.75)
      CALL RUNGE(DEQSET,7,100,0.024,YI,YSOLN)
      DO 180 KK=1,100
  180 S(N,J,KK)=YSOLN(1,KK)*(1.0-YSOLN(2,KK))*(1.0-YSOLN(2,KK))
      F(N,J)=0.0
      DELT=0.024
      DO 190 KK=1,99
      LL=KK+1
  190 F(N,J)=F(N,J)+S(N,J,KK)*DELT+0.5*DELT*(S(N,J,LL)-S(N,J,KK))
      WRITE (6,9) F(N,J)
    9 FORMAT (1H0,40X,F15.8)
      DPHI(J)=F(4,J)-F(4,J-1)
      IF (DPHI(J).GT.0.0) GO TO 230
      J=J+1
      L=2*L
      IC=IC+1
      WI(1,NK,J)=WI(1,NK,J-1)+FACTR*FLOAT(L)*DW
      WI(2,NK,J)=WI(2,NK,J-1)+FACTR*AM(NK)*FLOAT(L)*DW*50.0
      GO TO 240
  230 J=J+1
      NK=NK+1
      DW=0.0001
      WI(1,NK,J)=(WI(1,NK-1,J-1)+WI(1,NK-1,J-2))/2.0
      WI(2,NK,J)=(WI(2,NK-1,J-1)+WI(2,NK-1,J-2))/2.0
      A(1)=8.55+WI(2,NK,J)
      A(2)=404.46+8.55*WI(2,NK,J)
      A(3)=220.48+404.46*WI(2,NK,J)
```

```
      A(4)=24.0+220.48*WI(2,NK,J)+900.0*WI(2,NK,J)/WI(1,NK,J)
      A(5)=924.0*WI(2,NK,J)+720.0*WI(2,NK,J)/WI(1,NK,J)
      A(6)=720.0*WI(2,NK,J)
C     INITIAL CONDITIONS
      YI(1)=0.0
      YI(2)=0.0
      YI(3)=0.0
      YI(4)=0.0
      YI(5)=0.0
      YI(6)=900.0*WI(2,NK,J)/WI(1,NK,J)
      YI(7)=900.0*(WI(2,NK,J)/WI(1,NK,J))*(WI(1,NK,J)-WI(2,NK,J)-7.75)
      CALL RUNGE(DEQSET,7,100,0.024,YI,YSOLN)
      DO 300 KK=1,100
  300 S(N,J,KK)=YSOLN(1,KK)*(1.0-YSOLN(2,KK))*(1.0-YSOLN(2,KK))
      F(N,J)=0.0
      DELT=0.024
      DO 310 KK=1,99
      MM=KK+1
  310 F(N,J)=F(N,J)+S(N,J,KK)*DELT+0.5*DELT*(S(N,J,MM)-S(N,J,KK))
      IF (F(4,J).LT.F(4,J-2)) GO TO 400
      WI(1,NK,J)=WI(1,NK-1,J-2)
      WI(2,NK,J)=WI(2,NK-1,J-2)
  400 IC=1
      IF (NK.EQ.2) AM(2)=-1.0/AM(1)
      IF (NK.EQ.3) AM(3)=((WI(2,3,J)-WI(2,1,1))/(WI(1,3,J)-WI(1,1,1)))
      IF (NK.EQ.4) AM(4)=-1.0/AM(3)
      IF (NK.EQ.5) AM(5)=AM(3)
      IF (NK.EQ.6) AM(6)=((WI(2,6,J)-WI(2,4,1))/(WI(1,6,J)-WI(1,4,1)))
      IF (NK.EQ.7) AM(7)=-1.0/AM(6)
      IF (NK.EQ.8) AM(8)=AM(6)
      IF (NK.EQ.9) AM(9)=((WI(2,9,J)-WI(2,7,1))/(WI(1,9,J)-WI(1,7,1)))
      IF (NK.EQ.10) AM(10)=-1.0/AM(9)
      IF (NK.EQ.11) AM(11)=AM(9)
      IF (NK.EQ.12) AM(12)=((WI(2,12,J)-WI(2,10,1))/(WI(1,12,J)-WI(1,10
     1,1)))
      IF (NK.EQ.13) AM(13)=-1.0/AM(12)
      IF (NK.EQ.14) AM(14)=AM(12)
      IF (NK.EQ.15) AM(15)=((WI(2,15,J)-WI(2,13,1))/(WI(1,15,J)-WI(1,13
     1,1)))
      IF (NK.EQ.16) AM(16)=-1.0/AM(15)
      IF (NK.EQ.17) AM(17)=AM(15)
      IF (NK.EQ.18) GO TO 250
      WS(1,NK,1)=WI(1,NK,J)
      WS(2,NK,1)=WI(2,NK,J)
      GO TO 260
  250 CALL EXIT
      END
$IBFTC QRUNGE
      SUBROUTINE  RUNGE(DEQSET,NDEQ,NVAL,H,YI,YSOLN)
C
C  RUNGE-KUTTA SOLUTION TO A SYSTEM OF SIMULTANEOUS 1ST-ORDER DIFF-EQNS.
C  NDEQ, THE NUMBER OF EQUATIONS, MUST NOT EXCEED 10
```

```
C  NVAL, THE NUMBER OF POINTS AT WHICH SOLUTION IS DESIRED, INCLUDES X0.
C  H IS THE STEP LENGTH FOR THE INDEPENDENT VARIABLE.
C  YI IS THE ARRAY OF INITIAL VALUES OF THE FUNCTIONS.
C  YSOLN, DIMENSIONED (NDEQ,NVAL), CONTAINS THE SOLUTIONS IN THE ORDER
C        IN WHICH THE DEPENDENT VARIABLES ARE DEFINED IN SUBRTN-'DEQSET'.
C
      DIMENSION  YI(7),YSOLN(7,100),A(6)
      DIMENSION  YY(10), YDOT(10), YDEL(10)
      COMMON A
      DO 1 I=1,NDEQ
    1 YSOLN(I,1) = YI(I)
      DO 2 J=2,NVAL
      DO 3 I=1,NDEQ
    3 YY(I) = YSOLN(I,J-1)
      CALL  DEQSET(YDOT,YY)
      DO 4 I=1,NDEQ
    4 YDEL(I) = YDOT(I)
      DO 5 KTIMES=1,2
      DO 6 I=1,NDEQ
    6 YY(I) = YSOLN(I,J-1) + YDOT(I)*H/2.
      CALL  DEQSET(YDOT,YY)
      DO 7 I=1,NDEQ
    7 YDEL(I) = YDEL(I) + 2.*YDOT(I)
  5   CONTINUE
      DO 8 I=1,NDEQ
    8 YY(I) = YSOLN(I,J-1) + YDOT(I)*H
      CALL  DEQSET(YDOT,YY)
    2 CONTINUE
      DO 9 I=1,NDEQ
    9 YSOLN(I,J) = YSOLN(I,J-1) + (YDEL(I)+YDOT(I))*H/6.
      RETURN
      END
$IBFTC  FUNC1
      SUBROUTINE DEQSET(YDOT,YY)
      DIMENSION YY(7),YDOT(7),A(6)
      COMMON A
C     EQUATION FOR THE INDEPENDENT VARIABLE, INITIALLY ZERO
      YDOT(1)=1.0
C     SYSTEM EQUATIONS
      YDOT(2)=YY(3)
      YDOT(3)=YY(4)
      YDOT(4)=YY(5)
      YDOT(5)=YY(6)
      YDOT(6)=YY(7)
      YDOT(7)=A(6)-A(6)*YY(2)-A(5)*YY(3)-A(4)*YY(4)-A(3)*YY(5)-
     1A(2)*YY(6)-A(1)*YY(7)
      RETURN
      END
$IBSYS
```

Continued and scale invariant partan programs are slightly different to one given here. The difference lies in the different value of slopes taken in each case.