THE UTILITY OF NONLINEAR PROGRAMMING ALGORITHMS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Eric Sandgren

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 1977

Grad. School
Form No. 9
Revised

PURDUE UNIVERSITY

Graduate School

This is to certify that the thesis prepared

By ERIC SANDGREN

Entitled THE UTILITY OF NONLINEAR PROGRAMMING ALGORITHMS

Complies with the University regulations and that it meets the accepted standards of the Graduate School with respect to originality and quality

For the degree of:

Doctor of Philosophy

Signed by the final examining committee:

[illegible], chairman

A. Ravindran

[illegible]

[illegible]

Approved by the head of school or department:

September 7 1977 [illegible]

To the librarian:

~~is~~

This thesis is not to be regarded as confidential

[illegible]

Professor in charge of the thesis

Respectfully Dedicated

to my Father

Find Sandgren

ACKNOWLEDGMENTS

The author would like to express his deepest gratitude to Dr. K. M. Ragsdell for his guidance and council throughout this research. The author would also like to thank the other members of his committee, Drs. A. S. Hall Jr, G. V. Reklaitis and A. Ravindrin for their helpful assistance. Also my thanks go to the School of Mechanical Engineering of Purdue University and to the National Science Foundation for their support of the research.

A special vote of appreciation is due to the following people and corporations for contributing their algorithms and problems to the study, J. N. Siddall, L. J. LaFrance, Honeywell Corporation and S. B. Schuldt, J. Abadie, L. S. Lasdon, Gulf Oil Corporation, R. W. Mayne, Research Analysis Corporation, Shirley Lill, D. M. Himmelblau, R. S. Dembo, Whirlpool Corporation, R. Philipson, and F. A. Lootsma. Without contributions from these people and corporations this comparative study could not have taken place. Finally a special thank you is in order to my optimal design cohorts Gary A. Gabriele, Ronald R. Root and Behrooz Fallahi for their friendship and support.

TABLE OF CONTENTS

TABLE OF CONTENTS, cont.

LIST OF TABLES

LIST OF TABLES, cont.

LIST OF TABLES, cont.

LIST OF FIGURES

LIST OF FIGURES, cont.

ABSTRACT

Sandgren, Eric. Ph.D., Purdue University, December 1977. The Utility of Nonlinear Programming Algorithms. Major Professor: K. M. Ragsdell.

A comprehensive comparative study of nonlinear programming algorithms as applied to engineering design is presented. Linear approximation methods, interior penalty function methods and exterior penalty function methods were tested on a set of thirty problems and were rated on their ability to solve problems within a reasonable amount of computational time. The effect of the problem parameters on the solution time for the various classifications of algorithms was studied. The variable parameters included the number of design variables, the number of inequality constraints, the number of equality constraints and the degree of nonlinearity of the objective function and constraints. Also the combination of penalty function algorithms and linear approximation algorithms was investigated.

CHAPTER 1 INTRODUCTION

1.1 Introduction

The field of design engineering has been in a state of rapid evolution over the past two decades. This evolution has been greatly influenced by the introduction of computer aided design. Design procedures previously carried out by hand are now being programmed for a computational solution. This reliance on the computer originated at the dawn of the space age and has continued to an ever increasing degree.

Previously the goal of an engineering designer was to devise an adequate component. The fact that the proposed design was not very compact or slightly overdesigned and overweight was far overshadowed by the adequate performance of the component. However, as the space age progressed, components had to be designed which were strong, compact, extremely light in weight and above all dependable. These often competing objectives greatly increased the complexity of engineering design and introduced the concept of the "best" or optimal design to everyday language. Now the designer had to select from a wide range of feasible designs, the one that met the design criteria in the best fashion. This is where the computer became extremely helpful, for the designer was now required to attack problems which were of such a complex nature that once the design variables were selected it was not at all obvious as to how to proceed to meet the desired objectives. This is why design procedures formerly carried out by hand were converted for a computational solution. This conversion enabled the linking of the engineer with his knowledge of the problem and the computer which could quickly analyze a proposed design. With the current energy crisis and the ever increasing cost of material and labor, this linking of designer and computer to reach a better solution is rapidly becoming a way of life in mechanical engineering. A computational procedure to aid the designer in evaluating a series of choices of the design variables so as to best achieve the objective without violating any design constraints is then very desirable. A powerful computational tool which provides this aid is the field of nonlinear programming.

Based upon his knowledge of the problem the design engineer is able to define a set of design variables, a set of constraints which define the feasible region, and an overall objective of the proposed design. The mathematical formulation of this problem is to:

$$\text{Minimize} \quad f(\overline{x}); \quad \overline{x} = [x_1, x_2, x_3, \ldots, x_N] \; \varepsilon R^N \tag{1.1}$$

subject to

$$g_k(\overline{x}) \geq 0; \; k = 1, 2, 3, \ldots, K \tag{1.2}$$

and

$$h_\ell \equiv 0; \quad \ell = 1,2,3, \ldots, L \tag{1.3}$$

In this formulation $\bar{x}$ represents a column vector containing N design variables, $f(\bar{x})$ represents the objective function which gives an indication of the quality of a given design, and $g(\bar{x})$ and $h(\bar{x})$ represent the sets of inequality and equality constraints respectively which serve to limit the design space to some feasible region.

This general formulation has been applied to a wide range of problems over the past fifteen years, encompassing the fields of engineering, economics and the physical sciences. If the objective function and constraints are linear, the problem falls into the classification of linear programming, a highly developed branch of mathematical programming. Solution of a linear programming problem is possible even when the problem involves several hundred variables and constraints. Unfortunately, the objective function and constraints are generally nonlinear in nature for the majority of problems a mechanical engineer faces. This type of problem where either the objective function, the constraints or both may be nonlinear falls into the field of nonlinear mathematical programming and although the subject of much current research, there does not presently exist a general method to solve the nonlinear programming problem. So while the simplex algorithm can be successfully applied to practically any linear programming problem, the engineering designer must choose between a large number of available algorithms to attempt to solve a nonlinear programming problem with no guarantee that any selected method can solve the problem. Since each available algorithm developed to handle the nonlinear programming problem has a limited capacity to solve any given problem it becomes necessary to investigate the ability of each of them to handle the engineering design problem.

Several attempts at conducting comparative studies of nonlinear programming algorithms have been made to date but none of these studies have produced any conclusive results. In fact many of the results of these studies are contradictory and the designer still has no real guide in the selection of a nonlinear programming algorithm. It is the major goal of this work to provide this information.

1.2 A Brief Review of Constrained Nonlinear Programming Techniques

The vast majority of the algorithms available today which handle the constrained nonlinear programming problem as represented by equations (1.1) through (1.3) are of two basic types. The first approach known as the penalty function or transformation approach seeks to transform the constrained problem to a sequence of unconstrained problems. This sequence of unconstrained problems may then be solved by any of a large number of unconstrained search techniques. The second approach known as the linearization type approach handles the problem directly as a constrained problem by the linearization of the objective function and/or the constraints. Each of these basic methods will now be discussed in more detail.

1.2.1 Penalty Function Methods

The penalty type approach reformulates the constrained nonlinear programming problem to the following form:

$$\text{Minimize } P(\overline{x},R) = f(\overline{x}) + \Omega[R,g(\overline{x}),h(\overline{x})] \qquad (1.4)$$

Here $f(\overline{x})$ represents the original objective function and Ω represents the penalty term which is a function of the penalty parameters R, the inequality constraints and the equality constraints. The exact way in which the penalty term is formed defines the particular method. The basic idea behind the penalty function methods is to penalize any design which violates one or more of the constraints. However, it must be noted that this is accomplished by drastically distorting the contours of the original objective function which can make the unconstrained search very difficult. To circumvent this difficulty the original constrained problem is replaced by a sequence of unconstrained problems. At the initial stage the penalty term is designed so that the original function contours are not altered drastically. Then at each successive unconstrained problem the contours are altered to a greater extent making each unconstrained minimization more difficult. However, each successive unconstrained search is started from the solution of the preceding stage and the distance traveled from one stage to the next decreases as the number of stages increases. Ideally the increase in difficulty from stage to stage is offset by the smaller distance traveled so that each stage requires approximately the same computational effort. Unfortunately, this is not always the case.

The penalty function approach may be subdivided into two classes. The interior penalty function acts as a barrier to keep the successive points from leaving the feasible region with respect to the inequality constraints. Typical examples of an interior type penalty function would be

$$P(\bar{x},R) = f(\bar{x}) - R\sum_{k=1}^{K} \ln\{g_k(\bar{x})\} \tag{1.5}$$

or

$$P(\bar{x},R) = f(\bar{x}) + R\sum_{k=1}^{K} \frac{1}{g_k(\bar{x})} \tag{1.6}$$

The penalty function given by equation (1.5) has been implemented in one of the more widely used penalty function algorithms SUMT [1], and the penalty function given by equation (1.6), developed by Carroll [2], has been used in many algorithms including an earlier version of SUMT. Starting from a feasible point for a given value of R none of the constraints may be theoretically violated throughout the solution procedure. This is because in order to pass from the feasible region to the infeasible region one or more of the constraints would have to change from a positive value to a negative value. As any $g_k(\overline{x})$ approaches zero, however, the penalty term would add penalty value which approaches infinity thus creating a supposedly insurmountable barrier. So by starting with a given value of R (say between 1 and 10) and successively performing unconstrained searches where at the end of each unconstrained minimization R is divided by a factor (say 10), the successive stages should approach some constrained solution. Under certain conditions including a continuous objective function and constraints and the existence of a nonempty compact set the successive penalty stages can be shown to converge to the minimum of the constrained problem [3]. Realistically, however, the penalty type approach will handle a large number of problems which do not satisfy all of the required conditions for convergence to the constrained minimum.

Since equality constraints cannot continuously be satisfied by an interior penalty type method, this kind of constraint must be handled by an exterior penalty term. Thus an interior type penalty function which includes equality constraints is actually a mixed interior-exterior penalty function. The added term is usually a function of the square of the equality constraints. With this additional term equations (1.5) and (1.6) become:

$$P(\bar{x},R) = f(\bar{x}) - R\sum_{k=1}^{K} \ln\{g_k(\bar{x})\} + 1/R\sum_{\ell=1}^{L} h_\ell^2(\bar{x}) \qquad (1.7)$$

and

$$P(\bar{x},R) = f(\bar{x}) + R\sum_{k=1}^{K} \frac{1}{g_k(\bar{x})} + 1/R\sum_{\ell=1}^{L} h_\ell^2(\bar{x}) \qquad (1.8)$$

As the value of R is continuously decreased as required for the convergence of the interior portion of the penalty function the factor 1/R increases. This allows the equality constraints to be violated at the initial stages but as the penalty parameter R is reduced the penalty increases and the successive unconstrained solutions will approach feasibility with respect to the equality constraints.

The exterior penalty approach may also be used to handle the inequality constraints. This type of penalty function allows the solution to initially leave the feasible region with respect to the inequality constraints as well as the equality constraints. A typical example of an exterior penalty function would be:

$$P(\bar{x},R) = f(\bar{x}) + R\sum_{k=1}^{K} \langle g_k(\bar{x})\rangle^2 + R\sum_{\ell=1}^{L} \{h_\ell^2(\bar{x})\} \qquad (1.9)$$

The bracket operator used in this penalty function is used in the first summation so that only the values of the violated constraints are included. That is

$$\langle g_k(\bar{x})\rangle = 0 \quad \text{if} \quad g_k(\bar{x}) \geq 0 \tag{1.10a}$$

and

$$\langle g_k(\bar{x})\rangle = g_k(\bar{x}) \quad \text{if} \quad g_k(\bar{x}) < 0 \tag{1.10b}$$

For this case only a small penalty is added for constraint violations when R is relatively low and an increasingly large penalty is added as R increases. Thus by starting at a relatively low value of R (say 10) and successively performing unconstrained minimizations each time multiplying R by a factor (say 10) the successive stages can be seen to approach feasibility. Again convergence to the constrained optimum can be guaranteed under certain conditions.

In summary then, a penalty function method seeks to replace the original constrained problem by a series of unconstrained problems. The exact form of the penalty term is widely varied and each form produces a different method with different rates of convergence to the constrained minimum. The unconstrained searches may be performed by any of a number of methods. Some of the more frequently used methods are Davidon-Fletcher-Powell [4], Fletcher-Reeves [5], Powell's method [6], Hooke-Jeeves [7] and Broyden-Fletcher-Shanno [8]. The major disadvantage of this approach is that a number of unconstrained minimizations are required with each unconstrained search involving an increasingly distorted version of the original objective function. One method which seeks to reduce this distortion is Schuldt's Biased Penalty Method [9]. A more detailed discussion of penalty functions and the unconstrained methods incorporated with them may be found in references [10-12].

1.2.2 Linearization Methods

The linearization methods cover a rather broad range of algorithms. The first and most basic of the algorithms is that of Griffith and Stewart [13]. In this method the objective function and all constraints are expanded about a point $\bar{x}$ in a Taylor Series expansion with all of the nonlinear terms dropped. The resulting linear programming problem may then be solved by the application of the Revised Simplex Method [14]. The solution to this linear programming problem produces a new point $\bar{x}$ which will lie at the intersection of two or more of the linearized constraints. However, if any of the constraints are very nonlinear in nature the solution to the linear programming problem may be a very poor estimate of the constrained optimum so limits on the maximum change in each design variable are imposed at each stage. The basic advantage of this method is that it makes use of the well developed and readily available simplex method for the successive linear programs. The disadvantage is generally slow convergence for problems which are even moderately nonlinear.

A more recent development in the field of linearization methods is the reduced gradient concept which was developed independently by Wolfe [15] and by Wilde and Beightler [16]. For this method all inequality constraints are converted to equality constraints through the addition of slack variables. The variables, consisting of the original design variables plus the slack variables, are then partitioned into two sets. The first set $\bar{z}$ consists of the decision variables which are completely independent and the second set $\bar{y}$ consists of the state variables which are continuously adjusted to satisfy the constraints. The reduced gradient may then be defined as the rate of change of the objective function with respect to small changes in the decision variables with the state variables adjusted to maintain feasibility. The concept may be thought of as a projection of the original N variables into an N-L dimensional feasible space of the design variables.

The calculation of the reduced gradient is accomplished by linearizing the objective function and the constraints about $\bar{x}$. This gives

$$df(x) = \frac{\partial f}{\partial z} dz + \frac{\partial f}{\partial y} dy \tag{1.11}$$

and

$$dh_m(\bar{x}) = \frac{\partial h_m}{\partial z} dz + \frac{\partial h_m}{\partial y} dy = 0; \quad m = 1,2,3, \ldots, L, L+1, \ldots, L+K \tag{1.12}$$

Equation (1.12) may then be solved to find a linear approximation of the required change in the state variables in order to maintain feasibility.

$$dy = - \frac{\partial h}{\partial y}^{-1} \frac{\partial h}{\partial z} dz \tag{1.13}$$

Now combining equations (1.11) and (1.13) results in a linear approximation to the reduced gradient.

$$\frac{df^T}{dz} = \frac{\partial f^T}{\partial z} - \frac{\partial f}{\partial y} \frac{\partial h}{\partial y}^{-1} \frac{\partial h}{\partial z} \tag{1.14}$$

As the elements of the reduced gradient go to zero no change can be made in the decision variables to improve the objective function without leaving the feasible region.

Currently the generation of search directions has been accomplished through implementation of the Fletcher-Reeves, Davidon-Fletcher-Powell, or Broyden-Fletcher-Shanno unconstrained minimization techniques. The choice of search technique and the method for the selection and subsequent changing of the state and decision variables when a bound is encountered define each individual reduced gradient algorithm. Current published literature [17,18] indicates that the generalized reduced gradient is a robust and efficient method of solving the constrained nonlinear programming problem.

There are several algorithms available which address subsets of the general nonlinear programming problem. Among these algorithms are the method of feasible directions developed by Zoutendijk [19], separable programming and Box's method [20]. These methods will not be considered since they cannot handle the general nonlinear programming problem as represented by equations 1.1 through 1.3.

1.3 Literature Survey

Many of the early comparative studies of nonlinear programming algorithms involved unconstrained methods. Since the unconstrained problem is a subset of the general constrained problem and since many of the constrained algorithms depend upon an unconstrained technique any comparative information on unconstrained algorithms must be considered to be useful. Also different techniques used in conducting these studies have been used in the studies comparing constrained algorithms making their inclusion important when tracing the history of comparative studies for nonlinear programming codes. One of the first such studies was conducted by Brooks [21] in 1959. Brooks compared the Method of Steepest Ascent [22], univariate search, factoral and random methods on a series of four, two variable problems. Each method was started from various points on each problem and the methods were compared on the average improvement in the objective function after sixteen and thirty function evaluations. The results of the study indicated that the Steepest Ascent Method performed best followed by univariate search. The results are interesting only in the fact that even at this early stage gradient and pattern search type methods were demonstrated to be superior to the random based methods for this limited set of two dimensional problems. This study was followed by a group of comparative studies which appeared in the literature in the mid 60's. Among these were studies by Fletcher [23] in 1965, Leon [24] and Box [25] in 1966, and Kowalik and Osborne [26] in 1968. These studies involved the comparison of from three to eight algorithms including gradient and nongradient techniques on a set of five to eight test problems. The test problems were generally unconstrained and the majority of test problems were limited to less than four independent design variables. Box did include problems which involved the solution of simultaneous nonlinear equations which contained up to twenty variables. Box also included results on several constrained problems but the emphasis was on transformations to eliminate several of the constraints. The results of the Kowalik and Osborne study also included two constrained problems. The criteria for evaluation was usually based upon the number of function evaluations although Leon did include the computational time for each solution in his results. However some of the results included the number of function calls which would be required to calculate gradients for the gradient based methods and other results did not, even though analytical gradients were supplied. Also the uniformity of accuracy at the solution was not consistent in these studies although Box did make an attempt to stop the methods at approximately the same accuracy. The general consensus from these studies was that the variable metric algorithms performed somewhat batter than the others, however none of these results could be regarded as being conclusive considering the small sampling of problems and methods. These studies were followed by that of Colville [27] in 1968 which was by far the most comprehensive study attempted up to this time.

Colville sent eight constrained problems ranging from three to sixteen independent variables to the developers of thirty methods. Data was collected which included the solution, the computational time required to achieve the solution, and the number of function and constraint evaluations. The methods were divided into several broad categories including direct search methods, small step gradient methods, large step gradient methods, second derivative methods and miscellaneous methods. The results were based upon the number of problems solved and the mean value and standard deviation of the solution times over all of the problems. In order to compare the times of the different machines used in the study a standard timing routine involving several matrix inversions was sent to each participant and all times reported by each participant were divided by the reported time to run the standard timing routine. While the results of the Colville Study were inconclusive, several interesting observations were made. First of all it was found that the performance of a nonlinear programming algorithm was greatly affected by how efficiently it was implemented. This was demonstrated by several versions of the same type of algorithm producing vastly different results. Colville also pointed out that the number of function evaluations, the primary basis of comparison for earlier studies, was not a good indication of performance. The time required to generate successive points for a given algorithm was found to be far more significant than the time spent evaluating the objective function and constraints for many of the problems tested. Colville stated that the large step gradient methods and the second derivative methods were faster and more robust than the other methods. Unfortunately, most of these methods required analytically computed gradients while other codes did not. Problems encountered in this study include comparison of solutions of different accuracies, the use of the standard timing routine which has since been shown to be invalid for accurate comparison [10] and the use of analytically calculated gradients with some methods and numerically calculated gradients with others. Also by having the developer of each algorithm actually solve each problem the time reported may be significantly lower than that expected by an average user. However, even with these inherent difficulties the Colville study in many ways pointed out how a comparative study should be conducted in order to produce meaningful comparative results.

In 1970 Abadie and Guigou [17] reranked Colville's test data including the results for their updated version of the generalized reduced gradient code tested in the Colville study. The results show the new version to be significantly faster than the old version which already had the best weighted average score of the algorithms tested in the Colville study. Again, however, these results were calculated using the questionable standard timing routine, but the apparent superiority of the generalized reduced gradient technique was demonstrated. Stocker [28] conducted a comparative study including many of the algorithms tested in the Colville study and several new methods in 1969. Complete results from this study are not generally available but some results are presented in [10] along with additional data and recommendations. General performance is indicated for seven algorithms on twenty problems but no specific comparative criteria was used. Again some of the codes were supplied with analytic gradients and others used numerically calculated gradients. Other small scale studies were also conducted during this time period. Among these studies were those conducted by Pearson [29] in 1969 and Huang and Levy [30] and Murtagh and Sargent [31] in 1970. Pearson compared the results of seven algorithms on two unconstrained and three constrained problems. The constrained problems were solved using a logarithmic penalty function. The studies conducted by Huang and Levy and Murtagh and Sargent involved the comparison of several quadratically convergent algorithms on very limited sets of test problems. In all of these studies the main criteria for evaluation was the number of function evaluations and no specific comparative results were presented.

Applications of several nonlinear programming methods to a specific type of problem are common throughout this time period. Applications to least square problems were considered by Bard [32] and Jones [33] in 1970 and to optimal control problems with terminal state constraints incorporated as a penalty term by Pierson and Rajtora [34] in 1970. A more recent comparison of methods for the solution of structural problems was conducted by DeSilva [35] in 1973. These studies are typical of application type comparisons in that they consider a very limited number of algorithms and test problems. For this reason the results do not provide much comparative information. In 1971 at the Conference on Numerical Methods for Nonlinear Optimization held at the University of Dundee in Scotland several comparative papers were presented. Sargent and Sebastian [36] and Himmelblau [37] gave results for unconstrained algorithms. The work of Sargent and Sebastian was directed more at testing the effect of changing the various parameters within an algorithm rather than the actual comparison of different methods, but the significant effect of these parameters on the performance of the tested algorithms pointed out a variability not considered in the past comparative studies. Himmelblau presented results for fifteen methods on fifteen relatively small scale test problems. Solution time was used as the criteria for evaluation and a single termination criteria was applied to all algorithms in an attempt to stop all algorithms at the same approximate level of accuracy. Analytically calculated gradients were supplied to all algorithms requiring gradient information. The results of this study again point out the performance of the variable metric algorithms of Davidon-Fletcher-Powell and Broyden-Fletcher-Shanno along with Fletcher-Reeves to be better than the performance of the other algorithms tested. Other papers from this conference included a comparison of several random search procedures by Schrack and Borowski [38] and a comparison of several penalty function type algorithms by Biggs [39]. Schrack and Borowski did not recommend that random search techniques be used in place of gradient search methods but applications such as locating starting points for other algorithms and for searching about the final point generated by another method were suggested. The study by Biggs involved the comparison of a standard interior and exterior penalty function with two algorithms which replace the successive unconstrained minimizations with a sequence of quadratic programming problems. The results are based on the solution time and on the number of function evaluations required on a series of nine test problems, three of which were used by Colville. Results indicate that the methods involving a sequence of quadratic programming problems can be effective. However comparison with only one interior and one exterior penalty function method on a set of small scale problems is in no way conclusive.

A major study directed at engineering type problems was conducted by Eason and Fenton [40] in 1972. In this study twenty nonlinear programming algorithms were tested on thirteen problems. All of the algorithms were collected and run at the University of Toronto which eliminated the need for the standard timing routine. Advances over previous studies were the use of several solution points to establish an error-time curve which allowed a comparison of all methods at approximately the same accuracy and the use of numerically evaluated gradients for all gradient based algorithms. Several different ranking schemes were employed including those proposed by Colville and Abadie and Guigou, but again conclusive results were not forthcoming. In this study the direct search methods performed significantly better than any of the gradient based methods, a fact which contradicts the results of most of the previous studies. The top three algorithms were two methods which relied on a Hooke-Jeeves pattern search and a method using the Geometric Simplex method of direct search. Eason states that this result may have been due to the small number of independent variables present in all of the test problems and the fact that no analytic gradient information was supplied. Both of these observations are probably valid to some degree, but another point is that only standard parameters were used for all of the codes. The direct search methods generally need only initial and final estimates of the step size, while the gradient methods generally require several input parameters most of which are relatively problem dependent. While recommended values for these parameters are supplied in the users manual for almost every algorithm they are intended only as general starting values and in many cases a procedure for changing these standard parameters is supplied in case trouble is encountered in the solution procedure. It should also be pointed out that the gradient methods which performed the best in the Colville study, the generalized reduced gradient algorithms, were not included in the study by Eason, which may well be the major reason for the direct search methods relatively good performance. Furthermore the criteria for a method to have solved a test problem was sometimes related to the magnitude of the objective function at termination compared to that of the optimal objective function, and sometimes to the distance of the vector of design variables at termination to the optimal design variables. In several cases the required accuracy of solution of the design variables was specified so that the objective function was required to be accuarte to the eighth or ninth significant figure where any gradients would be so small that most gradient methods would stop before reaching this point. Whether any or all of these factors influenced the results of this study remains a matter of conjecture, but the point remains that the results contradicts almost all of the previously established results.

After Eason's study was completed several other small scale studies have been conducted. A slightly different approach toward comparison was attempted by Larichev and Gorvits [41] in 1974 where both an analytical study and experimental study were conducted on several algorithms on unconstrained nonlinear valley functions. This approach is an interesting one, however, it is not possible to apply this type of analysis to the general constrained nonlinear programming problem since no analytical expressions may be developed for the convergence of an algorithm for the general problem. The development of new algorithms has also resulted in several comparisons of methods. Three such studies were published by Pappas and Moradi [42] in 1975 and Schuldt et. al. [43] and Gabriele and Ragsdell [18] in 1977. Unfortunately, these comparisons rated the newly developed algorithms with the results of the study by Eason through the use of the questionable standard timing routine used by Colville. However results indicate the new techniques including a direct search method by Pappas and Moradi, a new penalty function method by Schuldt, and above all a generalized reduced gradient algorithm by Gabriele and Ragsdell perform very well. Also the results from Gabriele and Ragsdell and from Schuldt indicate that the use of numerically calculated gradients does not seriously effect the performance of gradient based methods.

The emphasis in the review of the past comparative studies has been placed on the experimental studies. Several studies of theoretical convergence have been conducted such as those by Wolfe [44] and Luenberger [45]. The theoretical convergence characteristics of the various algorithms are very important but they have several serious drawbacks for general comparison. First of all most theoretical convergence formulations are based on the improvement in the design vector from stage to stage and are in no way related to the time required for each stage. Also for many algorithms a "stage" may not be easily defined and the manner in which a stage is defined will in many cases significantly alter the apparent rate of convergence. Several algorithms such as the Hooke-Jeeves pattern search method would be difficult to classify as to the theoretical rate of convergence since no proof of convergence exists for such a method. Another problem is that even the theoretical convergence rates are problem dependent and may change drastically during the solution process. Also the convergence rate for a method such as the generalized reduced gradient algorithm can be altered on a given problem simply by changing the initial selection of basic variables. These inconsistencies make the theoretical approach very difficult to apply which is the basic reason most of the past studies have been experimental in nature.

1.4 Research Objectives

The shear number of comparative studies conducted to date of which those mentioned in the previous section is only at best a partial list may well be the best indicator of how much this type of information is needed. The fact remains, however, that through all of these attempts at obtaining this comparative information, not one has actually succeeded in providing consistant and conclusive comparative information. In truth the results presented to date give only a partial and often contradictory idea of the relative effectiveness of the algorithms available to date. Different rating criteria, varying termination criteria, inaccurate time comparisons between different machines, the small dimensionality of most test problems, the inconsistent use of analytic and numerically calculated gradients and the somewhat haphazard selection of algorithms all tend to cloud the picture rather than to bring it into focus. But even though every previous comparative study would seem to have several serious flaws, the net result is an indication of what must be done in order to achieve some valid comparative data. It is the major goal of this research to conduct a comparative study of nonlinear programming algorithms with application to engineering design in such a way as to produce some useful comparative information.

The research will be divided into four phases. These phases are the selection of algorithms, the selection of realistic design problems, experimentation and compilation of results, and to synthesize the results into some preliminary algorithm development. The study will be carefully and consistently conducted to avoid as many of the pitfalls of the past studies as is possible.

CHAPTER 2   PROCEDURE FOR GENERATING COMPARATIVE DATA

2.1   Introduction

It can be seen from the results of past studies that it is very important to set down a carefully designed procedure for generating and evaluating the comparative data. Before any data may be generated, however, several initial decisions must be made. These decisions include which algorithms to include, what test problems should be used and in general how the comparative data will be collected. Each of these decisions is extremely important for if any phase of the comparative study is not given adequate attention, the value of the study will be greatly reduced. For this reason each of these initial decisions will be considered in some detail.

2.2   Selection of Algorithms

In the selection of algorithms the main objective is to include as many as possible but to still keep the study to manageable proportions. To meet this objective, algorithms were solicited throughout the world and a wide selection of algorithms was obtained from both industry and the academic community. It should be noted that some of the codes which were initially considered were not included due to the fact that not everyone who was contacted agreed to participate. The overall response, however, was supportive and in all thirty-five algorithms were collected. A brief listing of the algorithms, their general classification and the unconstrained optimization technique used if any are included in Table 2.1. A more complete description of each of the codes is available in Appendix A. To simplify the terminology the numbering of the algorithms in Table 2.1 will be used throughout the study. For example, the code APPROX will be referred to as code number four. It can easily be seen from Table 2.1 that many of the algorithms included in the study are of the same basic class and even use the same unconstrained optimization techniques. This duplication was intentional and was meant to insure a fair testing of each class and type of method since as Colville pointed out two algorithms which are theoretically the same may produce drastically different results. Of course there are many additional algorithms which could have been included in the study and the fact that they were not included is in no way a reflection on the value of the code. The codes included are meant to be a representative sample of the codes currently available.

2.3 Selection of Test Problems

Just as it is important to obtain a wide variety of algorithms, it is equally important to select a wide range of test problems. It is in this aspect that essentially all of the past comparative studies have been lacking. The problems to be selected should include a wide range in the number of variables and the number of constraints and should include

Table 2.1 Algorithms Included in the Comparative Study.

Algorithm	Class	Unconstrained Search
(1) BIAS	EXTERIOR PENALTY	VARIABLE METRIC (DFP)
(2) SEEK1	INTERIOR PENALTY	PATTERN-RANDOM
(3) SEEK3	INTERIOR PENALTY	PATTERN (HOOKE-JEEVES)
(4) APPROX	LINEAR APPROXIMATION	NONE
(5) SIMPLX	INTERIOR PENALTY	PATTERN (SIMPLEX)
(6) DAVID	INTERIOR PENALTY	VARIABLE METRIC (DFP)
(7) MEMGRD	INTERIOR PENALTY	MEMORY GRADIENT
(8) GRGDFP	REDUCED GRADIENT	VARIABLE METRIC (DFP)
(9) RALP	LINEAR APPROXIMATION	NONE
(10) GRG	REDUCED GRADIENT	VARIABLE METRIC (BFS)
(11) OPT	REDUCED GRADIENT	CONJUGATE GRADIENT (F-R)
(12) GREG	REDUCED GRADIENT	CONJUGATE GRADIENT (F-R)
(13) COMPUTE II (0)	EXTERIOR PENALTY	PATTERN (HOOKE-JEEVES)
(14) COMPUTE II (1)	EXTERIOR PENALTY	CONJUGATE GRADIENT (F-R)
(15) COMPUTE II (2)	EXTERIOR PENALTY	VARIABLE METRIC (DFP)
(16) COMPUTE II (3)	EXTERIOR PENALTY	PATTERN (SIMPLEX-HOOKE-JEEVES)
(17) MAYNE (1)	EXTERIOR PENALTY	PATTERN (UNIVARIATE SEARCH)
(18) MAYNE (2)	EXTERIOR PENALTY	STEEPEST DESCENT
(19) MAYNE (3)	EXTERIOR PENALTY	CONJUGATE DIRECTIONS (POWELL)
(20) MAYNE (4)	EXTERIOR PENALTY	CONJUGATE GRADIENT (F-R)
(21) MAYNE (5)	EXTERIOR PENALTY	VARIABLE METRIC (DFP)
(22) MAYNE (6)	EXTERIOR PENALTY	PATTERN (HOOKE-JEEVES)
(23) MAYNE (7)	INTERIOR PENALTY	PATTERN (UNIVARIATE SEARCH)
(24) MAYNE (8)	INTERIOR PENALTY	STEEPEST DESCENT
(25) MAYNE (9)	INTERIOR PENALTY	CONJUGATE DIRECTIONS (POWELL)
(26) MAYNE (10)	INTERIOR PENALTY	CONJUGATE GRADIENT (F-R)
(27) MAYNE (11)	INTERIOR PENALTY	VARIABLE METRIC (DFP)
(28) SUMT IV (1)	INTERIOR PENALTY	SECOND ORDER (NEWTON-RAPHSON)
(29) SUMT IV (2)	INTERIOR PENALTY	SECOND ORDER (NEWTON-RAPHSON)
(30) SUMT IV (3)	INTERIOR PENALTY	STEEPEST DESCENT
(31) SUMT IV (4)	INTERIOR PENALTY	VARIABLE METRIC (DFP)
(32) MINIFUN (0)	MIXED PENALTY	CONJUGATE DIRECTIONS (POWELL)
(33) MINIFUN (1)	MIXED PENALTY	VARIABLE METRIC (BFS)
(34) MINIFUN (2)	MIXED PENALTY	SECOND ORDER (NEWTON-RAPHSON)
(35) COMET	EXTERIOR	VARIABLE METRIC (BFS)

problems with equality as well as inequality constraints. The problems should also include a wide range in the degree of nonlinearity of the objective function and the constraints. These were the basic considerations in selecting the test problems for the study. Several problems were selected from past studies to give the study a sound historical foundation. These problems were included basically as initial test problems to gain familiarity with each of the algorithms, although the set of problems studied by Dembo [46] in a comparison of geometric programming algorithms included some interesting engineering applications. Other problems were selected from a wide range of additional engineering applications and a total of thirty test problems will be considered, seven of which have not appeared in any previous comparative study. A brief listing of the problems and the number of variables and constraints is contained in Table 2.2. A detailed description including complete starting and solution data along with a fortran listing of each problem is included in Appendix B. Thirty test problems may seem like a rather limited test set but to simply run each of the thirty-five algorithms once on the thirty test problems involves over one thousand individual runs. The computational cost and simply the time involved in collecting data on a larger test set is infeasible.

The problems range from two to forty-eight design variables and from four to seventy-five constraints including

Table 2.2 Problems Included in the Comparative Study.

Problem	N	K	L	Variable Bounds
(1) EASON #1	5	10	0	5
(2) EASON #2	3	2	0	6
(3) EASON #3	5	6	0	10
(4) EASON #4	4	0	0	8
(5) EASON #5	2	0	0	4
(6) EASON #6	7	0	4	12
(7) EASON #7	2	1	0	4
(8) EASON #8	3	2	0	6
(9) EASON #9	3	9	0	4
(10) EASON #10	2	0	0	4
(11) EASON #11	2	2	0	4
(12) EASON #12	4	0	0	8
(13) EASON #13	5	4	0	3
(14) COLVILLE #2	15	5	0	15
(15) COLVILLE #7	16	0	8	32
(16) COLVILLE #8	3	14	0	6
(17) DEMBO #1	12	3	0	24
(18) DEMBO #3	7	14	0	14
(19) DEMBO #4	8	4	0	16
(20) DEMBO #5	8	6	0	16
(21) DEMBO #6	13	13	0	26
(22) DEMBO #7	16	19	0	32
(23) DEMBO #8	7	4	0	14
(24) WELDED BEAM	4	5	0	3
(25) COUPLER CURVE	6	4	0	6
(26) WHIRLPOOL	3	0	1	6
(27) SNG	48	1	2	72
(28) FLYWHEEL	5	3	0	10
(29) AUTOMATIC LATHE	10	14	1	20
(30) WASTE WATER	19	1	11	38

N = number of design variables.

K = number of inequality constraints.

L = number of equality constraints.

variable bounds. Larger problems were considered but were eliminated due to the excessive computational cost of each run. Also consideration had to be given to the availability of solution information to allow for comparison, and problems which contained many local optima were excluded since it is very difficult to compare results at different solutions.

It should be noted that this problem set is much larger than that of any previous comparative study and the problems selected cover a wide range of applications including current industrial applications. The results on this problem set should then be representative of the type of problems the engineering designer might face.

2.4 Collection of Data

Even after the algorithms and test problems had been selected, considerable work and several critical decisions had to be made before the accumulation of test data could commence. A major amount of time and effort was required to convert all of the test codes to run on the Purdue University CDC 6500 Computing System. The basic change involved conversion of all double precision arithmetic to single precision. This was done since the number of significant digits available on the CDC 6500 using single precision arithmetic is greater than the number of significant digits available on many other machines using double precision arithmetic. This conversion was carried out for two basic reasons. First it assures uniformity in all of the methods with respect to arithmetic operations. This is important because the time required to perform an arithmetic operation in double precision may be significantly larger than performing the same operation in single precision. The second advantage of the conversion to single precision is in the storage space saved in the compilation of the program. In several cases this resulted in significant storage savings and allowed increasing the size of the problem the code could handle. Another programming change was made to all gradient based methods which required analytical gradients. These codes were converted to calculate gradient information numerically using a forward difference approximation. This conversion was necessary because several of the test problems were the result of a simulation making analytical derivatives very difficult if not impossible to calculate. Another point in favor of numerical derivatives is the amount of work involved in calculating the derivatives and in inputting these derivatives to the algorithm. Supplying analytical gradients for a ten variable, ten constraint problem involves supplying 110 gradient functions for a method requiring first derivatives and over 700 gradient functions for a method requiring second derivatives. This work is simply too time consuming and the possible source of too many input errors to even be considered for the general design environment. Another programming change involved removing all printed output from the basic iterative loop of each algorithm. This was done to eliminate the inclusion of print time in the total solution time. If all printing was delayed until the final solution, however, a significant increase in the required storage would be required, for at each stage the intermediate output would include the vector of design variables, the value of the objective function and constraints and the solution time to this point. What would be desirable would be to calculate the difference in time between the beginning and end of each stage and to print out all intermediate output after the call from the timer for the stage just finished and before the call to the timer for the next stage. The only disadvantage with this procedure is that a timing error is accrued with each stage which could significantly effect the final solution time. In spite of this drawback this timing procedure has been implemented in each of the algorithms through the use of the CDC 6500 systems library routine SECOND which returns the elapsed central processor time in seconds since the start of the job. By placing a call to subroutine SECOND before and after each solution stage the time required for each stage can be calculated from the time difference in the two system subroutine calls. The possible errors involved in computing the time in this fashion are considered in the next section.

The final and probably the most difficult problem to handle is what to do about the input parameters to the codes such as initial penalty parameters and line search criteria. Here a middle of the road approach was followed as compared to the approaches of Eason and Fenton and Colville. Eason and Fenton used only the recommended values for the input parameters and if the solution was not reached to the desired level of accuracy with these parameters the code was considered to have failed on that problem. Colville, on the other hand, allowed the developer of each algorithm a free hand at adjusting the input parameters with no upper limit on the number of trials a code was allowed on each problem. Neither of these approaches would seem to be representative of real world applications. The approach used by Eason and Fenton is convenient in the sense that each code would only have to be run once on each problem, but the input parameters are in general much too problem dependent to allow for a single value to be used throughout the study. No one could practically expect an engineering designer to abandon his attempt at the solution of a problem after a single run. This is especially true if the development of the mathematical model has taken a significant amount of time. At the other end of the spectrum the approach used by Colville may be considered as an indication of the best possible performance for each algorithm on each problem. Again this is not the type of performance an average user would expect in everyday use. To simulate actual usage an initial run was made using the recommended values for the input parameters. If a normal, satisfactory termination was not obtained after this run an attempt was made to adjust the input parameters using the information contained in the users manual as a guide to the adjustment. It should be noted that no attempt was made to decrease solution times by this adjustment of the parameters since once normal termination was achieved with all active constraints reasonably tight (10^{-5} or smaller), the run was accepted for the study. The engineering judgement required to adjust the input parameters was developed by running each code on the Eason and Fenton and Colville problems before attempting to solve any of the more difficult problems.

2.4.1 Timing Accuracy

The system subroutine SECOND returns the execution time to an accuracy of 10^{-3} seconds. This means that the maximum error which can occur on the difference of two consecutive calls would be 10^{-3} seconds. This maximum error would occur if the first call occurred just before the clock was updated and the second call occurred just after the clock was updated or vice versa. For any problem which requires a significant amount of time for a stage to be completed this error is insignificant. For example if one tenth of a second is required for an algorithm to complete a stage the maximum error would be only one percent. The problem becomes more significant when the total solution time is small and many stages are required. This situation does occur on several of the Eason and Fenton problems with the reduced gradient codes which produced extremely fast solution times for these problems. The worst case was found to be for algorithm 11 on problem number five. On this problem algorithm 11 completed twenty-six stages in a total of .142 seconds. If the maximum error occurred at each stage in the same direction (i.e., either always positive or always negative error) the total timing error would be almost 20%. This error although only occurring on two or three problems would still be unacceptable. To determine a more accurate estimate of the expected error, ten consecutive runs for OPT on problem number five were made. The computed time for each of these runs is given in Table 2.3, along with the average time and the standard deviation for the runs. The standard deviation was only .002 seconds. If a normal distribution is assumed then even allowing for a time differing from the average by two standard deviations would produce an error of less than 3%. If the distribution were truly normal the

Table 2.3 Consecutive Runs of OPT on Problem Number Five After Twenty-six Stages.

Run Number	Solution Time
1	.140
2	.143
3	.142
4	.141
5	.146
6	.144
7	.139
8	.144
9	.143
10	.141

Average Time = .1423

Standard Deviation = .002

probability of a run differing from the average by more than two standard deviations would be extremely small. So even if the distribution only approximates a normal distribution significant errors could not be expected from this manner of timing. There are several system dependent factors which can also influence the time required to perform an identical set of calculations. The Purdue Computing System consists of two CDC 6500's sharing the same central memory, and although identical, their speed differs by a very small amount. Also the system load may have an effect on how long certain operations take. To obtain an estimate of the errors introduced by these factors the standard timing routine suggested by Colville was run a total of twelve times. Six runs were made in the middle of the afternoon when the computing load was very heavy. The other six runs were made late at night when the computing load was very light. Each sequence of runs included runs on both of the CDC 6500 machines. For all of these runs the average time was 49.89475 seconds and the standard deviation was .12984 seconds. Again allowing for two standard deviations the error involved would be less than 1%. This small level of error would have to be regarded as being inconsequential.

2.5 Data Reduction

Once each algorithm had been run on a problem intermediate data including $f(\bar{x})$, $g(\bar{x})$, $h(\bar{x})$ and the solution time was available at each stage of the solution procedure. In order to compare the relative effectiveness of different algorithms which may follow completely different paths to the solution some measure of the accuracy of any given point relative to the known solution must be determined. This is necessary so that all of the codes may be compared at some uniform level of accuracy on each problem. Eason and Fenton suggested two accuracy criteria. The first may be defined as the relative error in the objective function. This accuracy criterion may be expressed as:

$$\varepsilon_f = \frac{|f(\bar{x}) - f(\bar{x}^*)|}{|f(\bar{x}^*)|} \quad \text{for } f(\bar{x}^*) \neq 0 \qquad (2.1)$$

or

$$\varepsilon_f = |f(\bar{x})| \quad \text{for } f(\bar{x}^*) = 0 \qquad (2.2)$$

In this expression $f(\bar{x})$ is the value of the objective function at any point $\bar{x}$, and $f(\bar{x}^*)$ is the optimal value of the objective function. The other accuracy criterion suggested by Eason and Fenton is the relative error in the $\bar{x}$ vector. The expression is given by

$$\varepsilon_x = \sqrt{\sum_{i=1}^{N} [t(x_i)]^2} \qquad (2.3)$$

where

$$t(x_i) = \frac{x_i - x_i^*}{x_i^*} \quad \text{for } x_i^* \neq 0 \tag{2.4}$$

or $$t(x_i) = x_i \quad \text{for } x_i^* = 0 \tag{2.5}$$

Here x_i is the value of the ith variable for any point $\bar{x}$ and x_i^* is the optimal value of the ith variable. By plotting either of these accuracy criteria versus time for all algorithms on a given problem, the time for each algorithm to reach a specified level of accuracy may be closely approximated. For example the relative accuracies for three algorithms are plotted for a hypothetical problem in Figure 2.1. Each circled point represents the end of a stage for a given algorithm. Once the required level of accuracy for the problem has been determined then the time for each algorithm to reach this level of accuracy may be read directly from this plot. For example if the relative accuracy for the hypothetical problem is set at 10^{-4} the solution time for algorithm A is found to be approximately 8.6 seconds from Figure 2.1.

Each of the relative error criteria used by Eason and Fenton have certain advantages. The expression for the relative error in the $\bar{x}$ vector can be related to the rate of convergence of an iterative process [47]. On the other hand, the objective function is defined as quantifying the

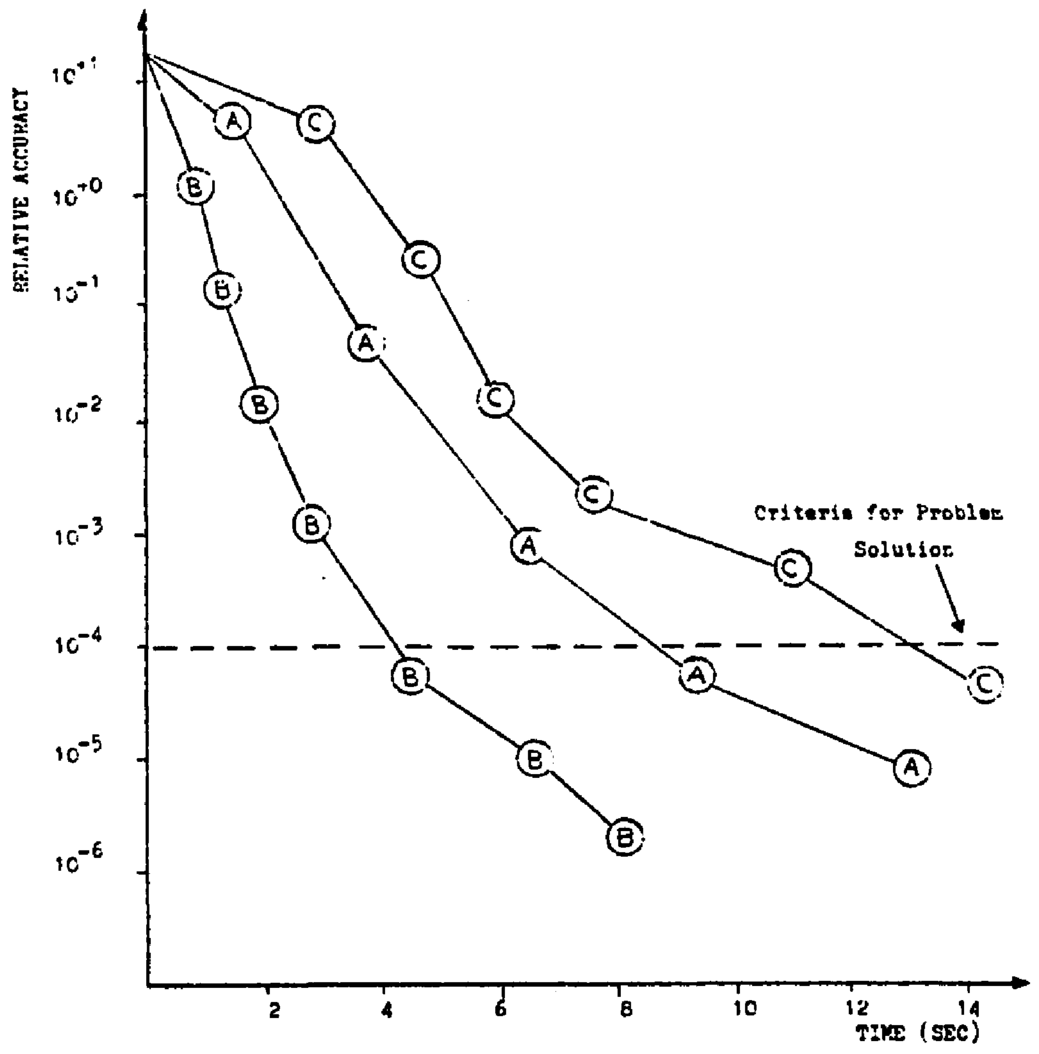

Figure 2.1 The Relative Accuracy of Solution versus Time for a Hypothetical Problem.

"goodness" of a design, so the relative error in the objective function would seem to be an obvious choice for an accuracy or error criterion. Of course the relative error in $f(\bar{x})$ is related to the design variables and thus it is also related to the relative error in the $\bar{x}$ vector. However a one to one correspondence cannot be made which is consistent for all problems. For example a value for ε_f of 10^{-5} may correspond to a value for ε_x of 10^{-3} for one problem and for another problem a value for ε_f of 10^{-3} may correspond to a value for ε_x of 10^{-4}. Eason and Fenton used a value of the relative error in the objective function to obtain solution times for some problems and the relative error in the $\bar{x}$ vector for the other problems. Even for problems where the same error criterion was used the required level of accuracy was not held constant in the Eason and Fenton study. For example on one problem the required relative error in the $\bar{x}$ vector was specified to be 3×10^{-3} and for another problem it was specified to be 10^{-5}. This is another inconsistency in the Eason and Fenton study, for this means that the level of required accuracy was not held constant for all of the problems. The percentage error in the solution times generated by using the relative error in the objective function at $\varepsilon_f = 10^{-4}$ compared to the solution times generated by using a value of ε_x at the same approximate accuracy level is presented for five algorithms on five of the initial test problems in Table 2.4. To remove any bias due to slight inaccuracies in the level of the relative required accuracies used for ε_f and ε_x, the solution time for each algorithm was normalized by dividing by the average solution time of all five methods on that problem using the same error criteria. The results appear to be highly problem dependent but in several cases the difference in the normalized solution times is almost 14%. To avoid this inconsistency only one relative accuracy criterion will be used and the reported solution times for all algorithms on every problem will be at the same level of accuracy. The relative error in the objective function was selected as the basis for the accuracy criterion. This selection was made for several reasons. First of all the objective function represents the quality of a candidate design and if a wide range of points about the optimal design vector produce little change in the objective function, a code should not be penalized for not precisely zeroing in on the optimal point. Secondly for several of the test problems there are several completely different combinations of the design vectors which produce the optimal value of the objective function. This makes no difference to a criterion based upon the relative value of the objective function but it does require special consideration for a criterion based upon the relative error in the design vector.

Table 2.4 Percentage Deviation for the Average Normalized Times Based on ε_f and ε_x.

Algorithm	Problem #1	Problem #3	Problem #7	Problem #11	Problem #14
(9)	4.0%	9.4%	2.8%	—	7.7%
(11)	5.3%	0.0%	13.7%	0.0%	13.4%
(13)	.91%	11.4%	8.8%	1.5%	9.8%
(21)	4.6%	5.2%	13.8%	1.4%	8.4%
(31)	3.1%	5.8%	3.2%	.32%	1.1%

Unfortunately the relative error in the objective function alone does not provide complete information on the value of an intermediate point. Consider the contour plot for the two variable problem shown in Figure 2.2. For this hypothetical problem both points $x^{(1)}$ and $x^{(2)}$ would have an ε_f of 2. However this does not give a clear indication of the relative value of the two points. If the solution procedure was stopped at the point $x^{(1)}$ the solution would be of no use since a constraint is considerably violated. On the other hand if the solution at $x^{(2)}$ was accepted a feasible design would result. This problem would not be encountered if all algorithms remained within the feasible region at all times but this is simply not the case. Therefore if the relative error is to give a complete indication of the value of an intermediate point some indication of constraint violation must be included. This may be accomplished in several ways. One way could be to consider the relative error in the generalized lagrangian function. The generalized Lagrangian function may be expressed as:

$$L(\bar{x},\bar{u},\bar{v}) = f(\bar{x}) + \sum_{j=1}^{L} v_j h_j(\bar{x}) - \sum_{j=1}^{K} u_j g_j(\bar{x}) \qquad (2.6)$$

where $\bar{u}$ and $\bar{v}$ represent the Lagrange Multipliers. The relative error may then be defined as

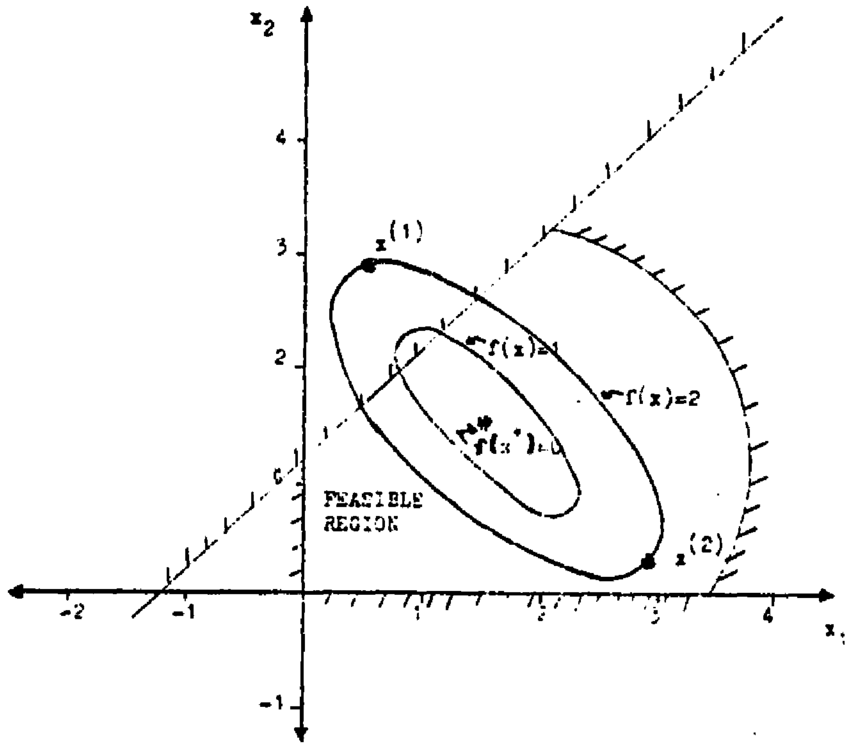

Figure 2.2 A Contour Plot of a Two Dimensional Problem.

$$\varepsilon_L = \frac{|L(\bar{x},\bar{u}^*,\bar{v}^*)|}{|L(\bar{x}^*,\bar{u}^*,\bar{v}^*)|} \qquad (2.7)$$

In this expression u* and v* are the values of the Lagrange Multipliers at the optimal solution. Equation 2.6 may be thought of as the relative error in the objective function with an additional term added on to penalize constraint violations with the Lagrange Multipliers acting as weighting factors for the constraint violations. Unfortunately, there are two problems with this expression for the relative error. First of all the Lagrange Multipliers u_j^* are zero for all $g_j(\bar{x}^*)$ which are not active at the solution. This means that no penalty is added to the relative error for a violation of a constraint which is not active at the solution. The second problem is that in many cases for the test problem set under consideration the Lagrange Multipliers are very small (10^{-2} or 10^{-3}) and do not add a significant amount to the relative error for any constraint violations. Both of these drawbacks may be eliminated if the weighting factors are chosen to be unity. For this case the total relative error may be defined as:

$$\varepsilon_t = \varepsilon_f + \sum_{j=1}^{K} \langle g_j(x) \rangle + \sum_{j=1}^{L} |h_j(x)| \qquad (2.8)$$

Again the bracket operator is used to indicate that the summation of the inequality constraints only includes the violated subset. A value or 10^{-5} in the total error ε_t as given by Equation 2.8 guarantees that the relative error in the objective function and all of the constraints violations are less than 10^{-5}. This straightforward total error criterion was applied uniformly to all of the test data. Plots of the total relative error for the algorithms on each problem may be found in Appendix C.

2.5.1 Elimination of Algorithms

It was apparent after running the algorithms on the Eason and Fenton and Colville problems that some algorithms were not performing well. These codes solved very few of these relatively easy test problems and even when a problem was solved an inordinately large amount of computer time was required. For these reasons any code which did not solve at least seven of these fourteen test problems was not considered further. Table 2.5 lists the number of problems in the initial problem test set solved by each of the algorithms considered for the study. It should be noted that for this table any algorithm which required more than three times the average solution time of all of the algorithms on a given problem was considered to have failed on that problem. The codes eliminated were algorithms 2, 4, 5, 6, 7, 17, 18, 23, 24, 25 and 30. This reduced the number of algorithms to a total of twenty-four. It should be noted however, that no renumbering of the remaining algorithms was done and the numbering scheme of Table 2.1 still holds.

Table 2.5 Number of Problems Solved for the Initial Test Problem Set.

Algorithm	Problems Solved	Algorithm	Problems Solved
(1)	12	(19)	11
(2)	5	(20)	10
(3)	8	(21)	12
(4)	6	(22)	7
(5)	6	(23)	4
(6)	5	(24)	4
(7)	2	(25)	3
(8)	14	(26)	10
(9)	9	(27)	11
(10)	14	(28)	8
(11)	13	(29)	9
(12)	14	(30)	2
(13)	12	(31)	11
(14)	8	(32)	12
(15)	10	(33)	11
(16)	10	(34)	8
(17)	4	(35)	11
(18)	5		

CHAPTER 3 RESULTS

3.1 Introduction

Many attempts have been made at developing a relative ranking criterion for comparing nonlinear programming algorithms. These criteria have been based on many different factors such as the number of problems solved or partially solved, some weighted average of execution times, the total running cost including input and output units and core usage or the estimated preparation times. The use of different rating criteria can significantly effect the relative rankings of the algorithms. This was demonstrated in the Eason and Fenton study where the relative rankings of the algorithms changed considerably depending upon which rating criterion was used. To avoid this problem careful consideration must be given to what characteristics a "good" algorithm should exhibit. Certainly the ability to solve problems must be considered to be the main characteristic of such an algorithm since this is the basic function of any nonlinear programming algorithm. But this quality alone does not represent the total value of an algorithm. Given enough computer time most algorithms will solve a fairly large set of problems. To generate a total picture of the relative effectiveness of an algorithm some consideration must be given to the computational time required for the solution of a test problem. The fact that one algorithm solved a specific problem in seven seconds while another required fifteen seconds is not significant in itself, but if the same algorithm consistently produced lower solution times considerable time savings could result in general usage. This would especially be true for large scale problems or for problems where the objective function or constraints require a considerable amount of time to evaluate. Taken by itself, however, the relative solution times of the different algorithms is not sufficient to comparatively rank algorithms either since one code which was extremely fast but only solved a few problems could rate well using this criterion. These basic criteria, the number of problems solved and the relative solution time, may be seen to be potentially competing objectives and a criterion based solely one or the other may not be a good performance indicator. Rankings involving the ease of use or preparation time were not considered since no algorithm tested could be considered to be significantly more difficult to use or required a significant amount of preparation time to input a problem. With these considerations in mind a relative ranking criterion based on the number of problems solved with a qualification on the relative solution times will be developed and applied to the comparative test data. Results based on this ranking criterion will then be compared to results generated by applying the rating schemes used by Eason and Fenton.

3.2 Rating Criterion

One method for dealing with competing design objectives is to treat the major objective as being the only objective and to treat any secondary objectives as constraints. This will be the approach used to develop the rating criterion to rank the algorithms in this study. Since the ability to solve a large number of problems in a reasonable amount of time is the desired ranking criteria, the rankings will be based on the number of problems solved within a series of specified limits on the relative solution times. The limits on the solution times will be based on a fraction of the average time for all algorithms on each problem. Each solution time for a problem was normalized by dividing by the average solution time on that problem. This produces a low normalized solution time for an algorithm with a relatively fast solution time and a high normalized solution time for an algorithm with a relatively slow solution time. This normalization essentially equalizes the time ratings on the various problems so that the effectiveness of an algorithm on a problem which required a very small amount of time may be directly compared to the effectiveness on a problem that required a large amount of computational time. The number of problems solved may now be directly related to the fraction or percentage of the average solution time of all of the methods tested on each problem. This relationship is demonstrated for all algorithms which solved at least half of the problems in the test set in Figure 3.1 for an accuracy level of 10^{-4} in ε_t. It should be noted that problems 9, 13, 21, 22, 28, 29 and 30 were not included in this analysis since less than five algorithms generated the same solution point. A discussion of the results on these problems is presented in section 3.5. In Figure 3.1, the number of problems solved at any fraction of the average solution time may be determined by drawing a vertical line at that value of the fraction of average time and recording the intersection with each algorithm. It can be seen from Figure 3.1 that the performance of the algorithms varies greatly, but the codes which have a steep slope and attain a high value in the ordinate axis have both a good problem solving capability and a relatively fast solution time on the majority of the problems.

3.3 Relative Rankings

Four values of the fraction of average solution time will be considered for the relative rankings. These values are 25%, 75%, 150% and 250%. The 25% rating is to indicate the codes which are extremely fast and should be considered for use on large and difficult problems. The 75% rating is

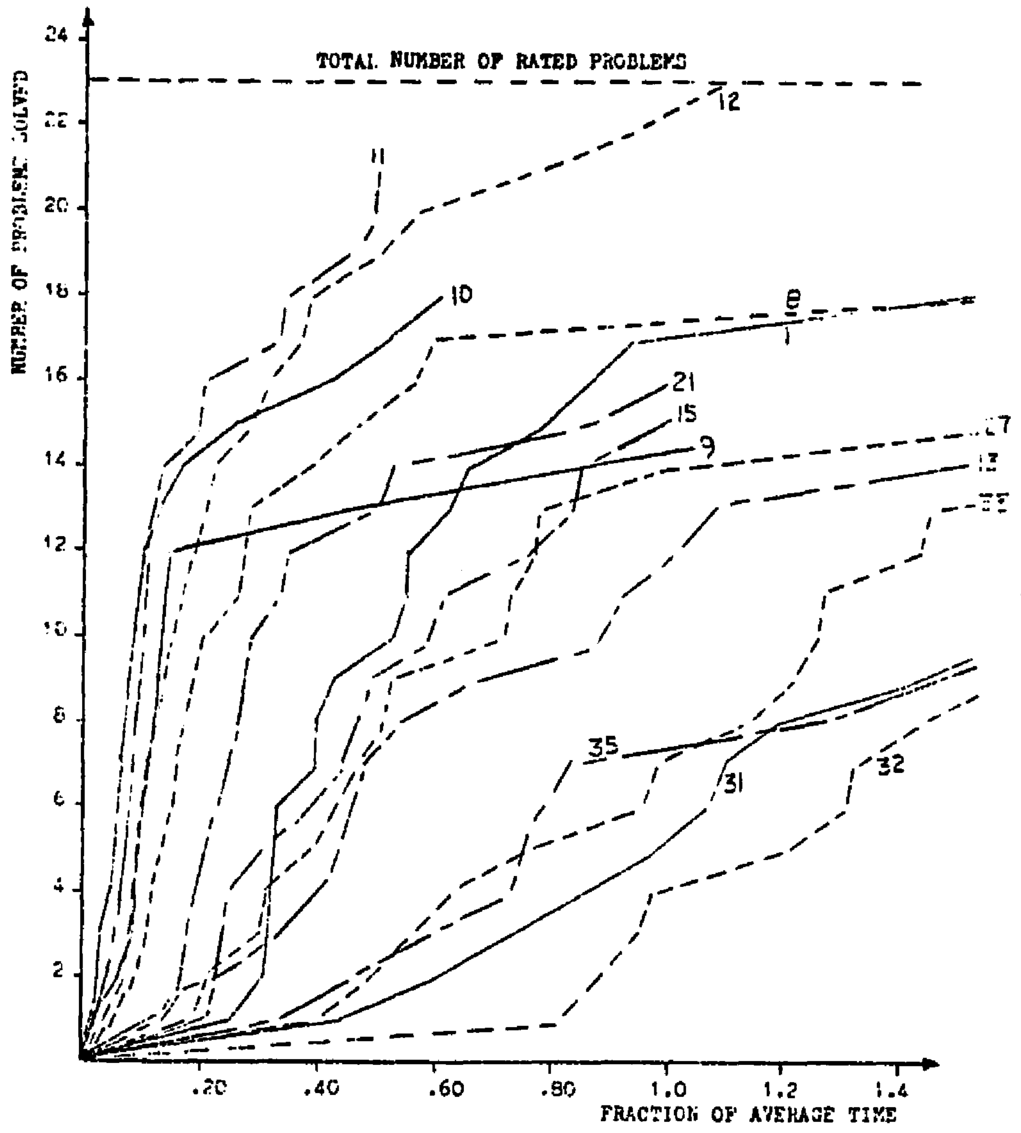

Figure 3.1 Number of Problems Solved versus the Fraction of Average Time for $\varepsilon_t = 10^{-4}$.

perhaps the most informative rating as it demonstrates the performance of the algorithms at a slightly better than average solution time. The codes that perform well at this rating could be considered to be the algorithms best suited for general usage. The final two levels, the 150% and 250% ratings were necessary to distinguish between some of the slower but effective algorithms. Algorithms which rate high at these levels but not at the 75% level could be considered robust but would require a considerable reduction in the average solution time to be recommended for general use. The performance for the values of the fraction of average time will be considered for solutions at accuracy levels of $\varepsilon_t = 10^{-4}$, $\varepsilon_t = 10^{-5}$ and $\varepsilon_t = 10^{-6}$.

3.3.1 Relative Ranking for $\varepsilon_t = 10^{-4}$

The number of the twenty-four remaining problems solved at fractions of the average time from .25 to 2.50 in intervals of .25 may be found listed in Table 3.1. From this table the percentage of the problem set solved at each rating interval can easily be calculated. The relative rankings for the 25% rating are given in Table 3.2. From Table 3.2 it can be seen that in general most methods solve only a very small percentage of total problem set within twenty-five percent of the average solution time. The exceptions are algorithms 10, 11, and 12, all of which solved over 60% of the test problem set at this level. All three of these algorithms

Table 3.1 Number of Problems Solved for Various Average Solution Time Limits for $\varepsilon_t = 10^{-4}$.

Algorithm	Number of Problems Solved					
	.25tavg	.50tavg	.75tavg	1.00tavg	1.50tavg	2.50tavg
1	0	9	14	17	19	20
3	0	1	2	3	6	10
8	10	15	17	17	18	18
9	12	13	13	14	14	16
10	15	17	19	19	19	19
11	16	21	21	21	21	21
12	14	18	20	23	23	23
13	2	7	9	11	13	15
14	2	4	6	6	8	9
15	4	9	11	15	15	15
16	1	5	7	8	9	11
19	0	0	2	3	7	11
20	1	4	10	10	11	11
21	7	13	14	16	16	16
22	2	4	7	8	8	9
26	0	1	4	6	9	10
27	2	7	11	14	14	17
28	0	0	0	0	3	9
29	0	0	0	1	2	7
31	0	1	3	5	9	13
32	0	0	0	4	8	15
33	0	2	4	7	13	15
34	0	1	2	4	8	10
35	0	2	5	7	9	15

Table 3.2 Relative Rankings for Algorithms for 25% Average Time Limit for $\varepsilon_t = 10^{-4}$.

Algorithm	% of Problems Solved
11	69.6
10	65.2
12	60.9
9	52.2
8	43.5
21	30.4
15	17.4
14	8.7
13	8.7
27	8.7
22	8.7
26	4.3
16	4.3

are generalized reduced gradient codes. The other algorithms which rated high at the 25% rating were algorithms 9 and 8 a repetitive linear programming algorithm and another generalized reduced gradient algorithm. The only other algorithm to reach at least a 30% ranking was algorithm 21, an exterior penalty method using the Davidon-Fletcher-Powell technique to solve the sequence of unconstrained problems.

The relative rankings for the 75% rating are presented in Table 3.3. Again the top four algorithms are all generalized reduced gradient type methods. These algorithms are followed by several exterior penalty function methods and the repetitive linear programming method RALP. The highest rating for an interior penalty function was a 37.8% rating achieved by algorithm 27. The relative rankings for the 150% and 250% ratings are presented in Tables 3.4 and 3.5. The main point to note in these Tables is that none of penalty function methods with the exception of algorithm 1 rate as well even at these levels as the three generalized reduced gradient algorithms 10, 11, and 12 did at the 75% rating.

3.3.2 Relative Rankings for $\varepsilon_t = 10^{-5}$ and $\varepsilon_t = 10^{-6}$

Table 3.6 presents the number of problems solved at various fractions of the average solution time for an accuracy in the total error of 10^{-5}. Rankings for the 25%, 75%, 150%, and 250% ratings are presented in

Table 3.3 Relative Rankings for Algorithms for 75% Average Time Limit for $\varepsilon_t = 10^{-4}$.

Algorithm	% of Problems Solved
11	91.3
12	87.0
10	82.6
8	73.9
1	60.9
21	60.9
9	56.5
15	47.8
27	47.8
20	43.5
13	39.1
16	30.4
22	30.4
14	26.1
35	21.7
26	17.4
33	17.4
31	13.0
19	8.7
34	8.7
3	8.7
All others	0

Table 3.4 Relative Rankings for Algorithms for 150% Average Time Limit for $\varepsilon_t = 10^{-4}$.

Algorithm	% of Problems Solved
12	100
11	91.3
1	82.6
10	82.6
8	78.3
21	69.6
9	65.2
15	65.2
27	60.9
13	56.5
33	56.5
20	47.8
16	39.1
31	39.1
35	39.1
26	39.1
14	34.8
22	34.8
32	34.8
34	34.8
19	30.4
3	26.1
28	13.0
29	8.7

Table 3.5 Relative Rankings for Algorithms for 250% Average Time Limit for $\varepsilon_t = 10^{-4}$.

Algorithm	% of Problems Solved
12	100
11	91.3
1	87.0
10	82.6
8	78.3
27	73.9
9	69.6
21	69.6
15	65.2
13	65.2
32	65.2
33	65.2
35	65.2
31	56.5
16	47.8
19	47.8
20	47.8
3	43.5
26	43.8
34	43.5
14	39.1
22	39.1
28	39.1
29	30.4

Table 3.6 Number of Problems Solved for Various Average Solution Time Limits for $\varepsilon_t = 10^{-5}$.

Algorithm	Number of Problems Solved					
	.25tavg	.50tavg	.75tavg	1.00tavg	1.50tavg	2.50tavg
1	0	5	9	10	10	11
3	0	1	2	2	3	7
8	11	14	15	16	16	17
9	11	12	12	12	14	15
10	16	17	19	19	19	19
11	15	20	21	21	21	21
12	12	18	20	23	23	23
13	1	7	8	9	10	10
14	2	4	5	5	6	6
15	2	8	10	12	13	14
16	1	4	6	7	8	9
19	0	0	2	4	5	6
20	0	3	6	6	6	6
21	5	9	11	11	11	11
22	1	2	2	5	5	5
26	0	1	3	5	8	8
27	0	3	9	11	13	14
28	0	0	0	1	2	8
29	0	0	0	0	2	7
31	0	1	2	4	8	12
32	0	0	0	0	4	11
33	0	1	4	5	11	14
34	0	1	1	3	6	10
35	0	2	4	7	8	14

Tables 3.7 through 3.10. These rankings basically demonstrate the same trends as did the rankings for the accuracy in ε_t in 10^{-4}. Throughout the rankings the generalized reduced gradient algorithms dominate with a larger gap developing between the reduced gradient algorithms and the penalty type methods. As the accuracy criteria is raised to an ε_t of 10^{-6}, the performance of all algorithms decreased. This is to be expected for most algorithms are not able to obtain this increase of required accuracy in the objective function and also maintain a sum of constraint violations of less than 10^{-6}. Again the exception may be seen to be the reduced gradient algorithms which still maintain their relative high percentage of problems solved at all relative rankings as demonstrated in Table 3.11.

3.4 Comparison with Other Ranking Schemes

Eason and Fenton proposed several rating schemes to rank the algorithms in their comparative study. The first criterion used was simply the total number of problems solved which would be closely related to the ranking used in this study at the 250% of the average solution time rating. Several other rating schemes which involved the solution times in some weighted fashion were also used. These criteria involved the average ratio of execution time to minimum execution time, the average ratio of execution

Table 3.7 Relative Rankings for Algorithms for 25% Average Time Limit for $\epsilon_t = 10^{-5}$.

Algorithm	% of Problems Solved
11	69.6
10	65.2
12	52.2
8	47.8
9	47.8
21	21.7
15	8.7
14	8.7
13	4.3
16	4.3
22	4.3
All others	0

Table 3.8 Relative Rankings for Algorithms for 75% Average Time Limit for $\epsilon_t = 10^{-5}$.

Algorithm	% of Problems Solved
10	91.3
12	87.0
11	82.6
8	65.2
9	52.2
21	47.8
15	43.5
1	39.1
27	39.1
13	34.8
16	26.1
20	26.1
14	21.7
33	17.4
35	17.4
26	13.0
3	8.7
19	8.7
22	8.7
31	8.7
34	4.3
All others	0

Table 3.9 Relative Rankings for Algorithms for 150% Average Time Limit for $\epsilon_t = 10^{-5}$.

Algorithm	% of Problems Solved
12	100
11	91.3
10	82.6
8	69.6
9	60.9
15	56.5
27	56.5
21	47.8
33	47.8
1	43.5
13	43.5
16	34.8
26	34.8
31	34.8
35	34.8
14	26.1
20	26.1
34	26.1
19	21.7
22	21.7
32	17.4
3	13.0
28	8.7
29	8.7

Table 3.10 Relative Rankings for Algorithms for 250% Average Time Limit for $\varepsilon_t = 10^{-5}$.

Algorithm	% of Problems Solved
12	100
11	91.3
10	82.6
8	73.9
9	65.2
15	60.9
27	60.9
33	60.9
35	60.9
31	52.2
1	47.8
32	47.8
21	47.8
13	43.5
34	43.5
16	39.1
26	34.8
28	34.8
29	30.4
3	30.4
14	26.1
19	26.1
20	26.1
22	21.7

Table 3.11. Number of Problems Solved for Various Average Solution Time Limits for $\varepsilon_t = 10^{-6}$.

Algorithm	Number of Problems Solved					
	.25tavg	.50tavg	.75tavg	1.00tavg	1.50tavg	2.50tavg
1	0	5	8	9	9	9
3	0	0	0	9	1	2
8	9	11	11	13	14	16
9	4	6	6	6	8	9
10	11	13	14	14	14	14
11	9	12	13	15	15	15
12	8	13	14	15	17	17
13	1	4	5	5	5	5
14	2	4	5	5	6	6
15	2	6	8	9	10	10
16	2	5	5	5	7	7
19	0	0	2	4	5	6
20	0	1	4	4	4	4
21	2	5	5	6	6	6
22	0	1	1	3	4	4
26	0	0	3	3	6	6
27	0	1	3	6	8	9
28	0	0	0	0	1	8
29	0	0	0	0	0	8
31	0	1	2	2	8	11
32	0	0	0	1	5	7
33	0	1	4	5	7	12
34	0	1	1	4	5	8
35	0	1	6	6	8	11

time to mean execution time and the sum of execution times. The first ranking system involving the average ratio of execution times to the minimum execution time seeks to compare each method with a hypothetical method which would be fastest on every problem. This ranking was used first in the study conducted by Abadie and Guigou and may be define as

$$f_a = \frac{\sum_p f_{ap}}{S1} \tag{3.1}$$

where

$$f_{ap} = \frac{t_{ap}}{\min_a (t_{ap})} \tag{3.2}$$

Here t_{ap} represents the solution time of method a on problem p, $\min_a (t_{ap})$ represents the lowest solution time of all of the methods which solved problem p, and S1 represents the total number of problems solved by algorithm a. The second ranking system which involves the average ratio of execution times to the mean solution time may be defined as

$$\bar{f}_a = \frac{\sum_p \bar{f}_{ap}}{S1} \tag{3.3}$$

where

$$\bar{f}_{ap} = \frac{\sum f_{ap}}{S1} \tag{3.4}$$

This ranking criteria is very similar to the first system described with the exception of the minimum time on problem p being replaced by the mean time on problem p. Both of these ranking criteria were meant to given an indication of how well each code compared relative to the other algorithms in the study but neither rating scheme gives an indication of how many problems were solved and a code which solved only a very few problems could rate very high using these criteria. The third ranking scheme proposed by Eason and Fenton is simply the sum of the execution times and may be represented by

$$T_e = \sum_p t_{ap} \tag{3.5}$$

When an algorithm failed to solve a problem a time equal to twice the slowest execution time recorded for that problem was substituted for t_{ap} in the summation. The test data was reevaluated using these rating criteria and the results are presented in Table 3.12. The ratings still show methods 10 and 11 to be the best by almost a two to one margin over method 12. All three are generalized reduced gradient algorithms but it should be noted that only method 12 solved all of the problems. Also in the f_{ap} rating method 22 ranks 3rd, even above method 12, while in the $\overline{f}_{ap}$ ratings method 22 would rank ninth. It is interesting to note that method 22 is an exterior penalty function using a pattern search to solve the unconstrained minimization problems and only solved nine out of the twenty-three problems.

Table 3.12 Eason and Fenton Ratings for $\varepsilon_t = 10^{-4}$.

Algorithm	$\bar{f}_{ap}$	f_{ap}	T_e
1	.673	10.25	641
3	1.475	22.65	1901
8	.588	5.19	801
9	.632	9.94	1256
10	.151	1.12	660
11	.175	1.57	396
12	.299	3.00	295
13	.739	12.33	1390
14	.652	8.08	1895
15	.498	6.16	1512
16	.713	13.99	1804
19	1.574	22.69	1862
20	.559	9.33	1664
21	.359	5.78	1220
22	.613	2.75	1605
26	1.453	19.81	1981
27	.654	12.98	1087
28	1.989	17.17	1984
29	2.204	46.24	1971
31	1.764	27.15	1349
32	1.809	40.02	1550
33	1.471	34.60	1377
34	1.962	45.31	1715
35	1.993	36.43	1366

The rating based on the total execution times simply reflects the number of the more difficult problems solved since on a difficult problem the penalty of twice the slowest time is as hard to overcome. The ratings for this criterion follow very closely to the number of problems solved with the reduced gradient algorithms first and with algorithm 1 rating very high which contradicts this method's poor showing on the first two rating criteria. All in all, however, it is clear that even with the inconsistencies generated in the rating criteria used by Eason and Fenton that the generalized reduced gradient algorithms rank very well. This is simply because the reduced gradient algorithms solve a large number of problems in a relatively small amount of computational time and will show up well in almost any kind of rating system.

3.5 Additional Problems

The problems not included in the relative rankings should still be considered, for although very few codes solved these problems many made significant progress or at least found feasible points. Several of the problems have many local minima and others had a very small feasible region making relative comparisons very difficult. Therefore each problem will be considered individually and the performance of each algorithm will be noted. Complete problem descriptions and references may be found in Appendix B.

3.5.1 Problem Number Nine

Problem number nine was used in the comparative study conducted by Eason and Fenton. It involves the design of a chemical reactor and has three design variables, nine functional constraints, and upper and lower bounds on two of the variables. The best solution reported by Eason and Fenton had an objective function value of -4.2446134. No algorithm in this study found a point near this solution. However when several algorithms were started from this point an unbounded solution was found for which the temperature drop in the cooling coil became infinite. An additional constraint was placed on the maximum temperature drop in the cooling coil of 114 degrees and the solution reported by Eason and Fenton then became a local minimum.

The best solution found from the specified starting point was $f(\bar{x}) = -3.995$ by algorithm 16. Three other methods found solutions below $f(\bar{x}) = -3$. These were method 32 with $f(\bar{x}) = -3.43$, method 35 with $f(\bar{x}) = -3.257$ and method 34 with $f(\bar{x}) = -3.100$. So both exterior and interior penalty functions algorithms using gradient and nongradient searching techniques performed well on this problem. Algorithms 9, 10, 19, 20, 21, 22, 26, 27, 28, 29, 31 and 33 all found solutions with a final value of the objective function less than -2.0 which would all be considered to have partially solved the problem in the Eason and Fenton study. The other methods demonstrating any significant amount of progress were algorithms 11 and 13 both of which generated final values in the objective function between -1.7 and -1.8. The remaining algorithms demonstrated little or no progress.

3.5.2 Problem Number Thirteen

Problem number thirteen was also used in the study conducted by Eason and Fenton. It involves selecting gear ratios for an automobile to produce the minimum time to accelerate to 100 mph. The RPM-Torque curve was specified at fourteen points and in the original problem a torque value was interpolated from the data for each RPM value. This procedure was modified by fitting a series of cubic splines through the data so for each range of RPM data the torque was available in a closed form. This change was implemented because the interpolation proved to be very time consuming and produced no increase in accuracy. The objective function is discontinuous over the whole feasible region in finite steps of .0001 seconds. To obtain any gradient information the step increment for the calculation of numerical derivatives had to be on the order of 10^{-3}. Apparently with this large increment the derivatives calculated were not accurate enough to find the optimal solution. This is borne out by the fact that the best solutions were produced by the nongradient methods. Even for the nongradient methods the solutions varied considerably with the best solution recorded by algorithm 16 of $f(\bar{x}) = 26.79$ seconds. Other good solutions were recorded by method 13 with a final $f(\bar{x}) = 27.22$ seconds and by method 22 with a final $f(\bar{x}) = 27.24$ seconds. All three of these methods employ a type of direct search procedure. The best solutions reported by the gradient based methods were centered around 27.50 seconds by algorithms 1, 10, 11, 12, 14, 15, 19, 20, 21 and 33. All other algorithms stopped above 27.70 seconds. The main point made by this problem is the need for an algorithm employing a non-gradient technique for some discontinuous problems.

3.5.3 Problem Number Twenty-one

Problem number twenty-one is a mathematical programming model of a three stage membrane separation process. The problem contains thirteen design variables, thirteen functional inequality constraints and upper and lower bounds on all of the variables. This problem proved to be very difficult because the feasible region is very small and many algorithms were simply unable to locate a feasible point. The solution is reported by Dembo as $f(\bar{x}^*) = 97.591034$. This solution was located by algorithms 12 and 31, while algorithm 13 produced a solution with $f(\bar{x}) = 98.332$. Several algorithms reached the vicinity of the optimal solution but did not terminate at a feasible point. These algorithms include methods 1, 19, 20, 21, 32, 33, 34 and 35. Several other algorithms terminated at a feasible point with objective functions in the range of 102 to 120. These algorithms were methods 9, 15, and 22. No other algorithms terminated at a point yielding an objective function value of less than 200, but it should be noted that methods 8, 10, 11 and 14 did produce feasible points. The remaining algorithms made no progress at all.

3.5.4 Problem Number Twenty-two

This problem is essentially an extension of problem twenty-one only now a five stage membrane separation process is being modeled. The problem contains sixteen variables, nineteen functional inequality constraints, and upper and lower bounds on all of the design variables. As with problem twenty-five many algorithms had difficulty locating a feasible point. The solution reported by Dembo has an optimal value of the objective function of 174.788807. This solution was found by algorithms 9, 15 and 31. Algorithms 14, 16, 32, 33, 34 and 35 terminated in the approximate vicinity of the solution but were unable to locate a feasible point. Algorithms 13 and 22 terminated at feasible points with objective functions under 220. Algorithm 10 also terminated in this vicinity but at an infeasible point. While no other method found a feasible point producing an objective function of less than 600, both algorithms 11 and 12 located feasible points.

Both problem twenty-one and twenty-two are representative of a wide class of problems where just locating a feasible point is difficult. Most of the algorithms tested had great difficulty with these two problems but this could also be attributed in part to the poor relative scaling between the design variables which range in value from 10^{-6} to 10^{3}. The only algorithm to solve both problems was algorithm 31 an interior penalty function method. It should be noted also that the large majority of time spent by any interior-type method was in generating a feasible starting point.

3.5.5 Problem Number Twenty-eight

This problem involves the design of a flywheel of arbitrary shape to generate the maximum kinetic energy for a specified volume and rotational speed. The inside radius of the flywheel was specified as well as the maximum radius and the maximum thickness the flywheel may obtain. The problem involves five design variables including the flywheel thickness at the inner radius, the slope of the thickness function at the inside radius, two Raleigh-Ritz Fourier coefficients and the outside radius of the flywheel. Originally the problem contained seven Raleigh-Ritz Fourier coefficients and the rotational speed was also included as a design variable but the problem had to be reduced to enable practical solution times. The problem contains three functional inequality constraints, one of which is a constraint on the maximum allowable stress at any radial location in the flywheel. The calculation of this constraint involves the solution of a boundary value problem for a second order differential equation. An iterative solution was employed for the solution of this constraint which required approximately .75 seconds of computational time for each evaluation of the constraint. With an upper limit on the allowed computational time of 500 seconds this only allowed for slightly over 650 constraint evaluations. Only two algorithms were able to produce the optimal solution in less than 150 seconds. Both algorithms 10 and 11 found the optimal solution of $f(\bar{x}) = -5.558$. Three other algorithms terminated at feasible points with an objective function value less than -5.0. These algorithms were methods 12 and 15 which generated final solutions with $f(\bar{x}) = -5.3$, and method 1 which terminated at $f(\bar{x}) = -5.1$. Only four other algorithms were able to make any progress within the allowed 500 seconds. These were method 35 with a final objective function value of -4.66, method 20 with $f(\bar{x}) = -4.48$, method 8 with $f(\bar{x}) = -3.6$ and method 29 with $f(\bar{x}) = -2.11$.

This problem was included to represent the large number of engineering problems in which the evaluation of the objective function or constraints involves a time consuming iterative analysis, and the fact that both of the algorithms solving the problem were generalized reduced gradient type methods should be noted.

3.5.6 Problem Number Twenty-nine

This problem involves maximizing the profit rate for the operation of a multi-spindle automatic lathe. The problem contains ten design variables, thirteen inequality constraints and one equality constraint. The overall performance of the algorithms in the study was very poor on this problem. Only two algorithms located the optimal solution of $f(x^*) = 1615$. These algorithms were methods 11 and 12, again both generalized reduced gradient type methods. Two other algorithms, methods 16 and 13 made significant progress with final objective function values of -1542 and -1337 respectively. No other algorithms made any significant progress with the large majority terminating as soon as the equality constraint was located.

3.5.7 Problem Number Thirty

This problem involves the design of a waste water treatment plant to minimize the total construction cost. The problem contains nineteen design variables, one functional inequality constraint, eleven equality constraints, and upper and lower bounds on all of the variables. The presence of eleven nonlinear equality constraints presented an extreme level of difficulty for most of the algorithms and no penalty type method was able to locate a feasible solution. Only algorithm 12 was able to generate the specified solution point of $f(\bar{x}^*) = 24.3841$. The only other algorithms which even located feasible points were methods 10 and 11, both of which terminated with objective value functions in the vicinity of 43.6400. This solution point was a local minimum for the problem as both of the methods terminated when the reduced gradient went to zero. Both this problem and problem 29 demonstrate the difficulty the penalty type methods have in the presence of nonlinear equality constraints.

CHAPTER 4 DISCUSSION AND EXTENDED RESULTS

4.1 Introduction

The results from the last chapter indicate an apparent superiority for the linearization type methods over the penalty-type methods. In this chapter the performance of each general classification of algorithms will be considered with attention given to the algorithms which demonstrated the best performance. Also several details will be considered about the manner in which the comparative study was conducted which could have possibly effected the comparative results. The items to be considered are the choice of system compilers and the possible effects from the variation of the input parameters. Also the portion of the total solution time spent in each computational phase will be considered for several algorithms. This will be done to determine where the algorithms which performed well in the study spend the majority of computational time.

4.2 Discussion of Results

Within each major classification of algorithms, the linear approximation methods, the exterior penalty function methods and the interior penalty function methods the performance of the codes tested varied considerably. To indicate which programming features within each classification proved to be effective on the test problem set the performance of each algorithm will be discussed. It is not the intention of the author to promote the use of any specific algorithm over another but to point out the type of algorithm which was most effective.

4.2.1 The Linear Approximation Methods

This classification includes both the repetitive linear programming algorithms and the generalized reduced gradient algorithms. All of the linear approximation methods with the exception of algorithm 4 performed very well. Algorithm 4, a repetitive linear programming method, was removed from further consideration after only solving six of the initial test problem set. This algorithm had difficulty with problems 4, 5, and 12, all of which had unconstrained solutions, and failed to satisfy the equality constraints on problems 6 and 15. Good progress was made on problems 11, 14 and 16 but the method had difficulty in adequately satisfying all of the constraints active at the solution. The other repetitive linear programming method, algorithm 9, fared much better. Again trouble was encountered on the essentially unconstrained problems 4 and 5 but a method relying on a linear programming routine would not be expected to do well on problems where the solution is not constrained. With the additional programming to handle equality constraints using Newtons method, algorithm 9 did not encounter much difficulty with the problems containing equality constraints, with the exception of problem number 6 for which Newtons method diverged while trying to initially satisfy the equality constraints. The other problems where difficulty was encountered were problem 23 where trouble was encountered in satisfying the inequality constraint, and problem 25 where significant progress was made but much time was consumed in locating the constraint which was tight at the solution. This was probably due to the fact that a significant distance had to be traveled in the feasible domain before the constraint is encountered. It should be noted that algorithm 9 was one of the few codes to solve problem 27 which contained forty-eight design variables. The performance of algorithm 9 on the problems it solved was quite good. For a total error criteria of $\varepsilon_t = 10^{-4}$, algorithm 9 solved over fifty percent of the problems in the study in less than 25% of the average solution time. The general trend was either a relatively fast solution or none at all for only four additional problems were solved after the 25% rating. On the additional problems considerable progress was made on problems 9 and 21 and it was one of the three algorithms to find a solution to problem 22. No progress was made on problem 13, another problem with an unconstrained solution, and on problems 28, 29, and 30. On problem 28 great difficulty was encountered in satisfying the maximum stress constraint which is highly nonlinear, on problem 29 the equality constraint was satisfied but no significant progress was made, and on problem 30 Newtons method again diverged while trying to locate a point with initial equality constraint feasibility. Overall performance then, was very good but the method encountered difficulty moving through unconstrained regions and when in the presence of highly nonlinear constraints. Handling the equality constraints using Newton's method proved to be quite effective with the exception of finding an initial feasible point. Perhaps the introduction of artificial variables for the equality constraints, as is done with several of the reduced gradient algorithms, would help with this problem.

The effectiveness of the reduced gradient algorithms was unmatched by any other type of algorithm. The four generalized reduced gradient algorithms included in the study all performed extremely well. These four algorithms held four of the top five rankings in the relative ratings for all levels of accuracy.

Algorithm 12, employing the Feltcher-Reeves conjugate gradient technique to generate search directions was the only method which was able to solve all twenty-three of the test problems included in the relative rankings. It should also be noted that all problems were solved within 100% of the average time indicating that algorithm 12 is not only very robust, but also very fast. For the additional test problems no progress was made on problem 9 but progress was made on problem 13, problems 21, 29 and 30 were solved, significant progress was made on problem 28, and a feasible point was found for problem 22 although not much progress was made.

Algorithm 11, again employing the Fletcher-Reeves conjugate gradient technique to generate search directions, was also very effective. Out of the twenty-three rated test problems, algorithm 11 solved twenty-one. The two problems which were not solved were problem 12 where trouble was encountered in moving off of a variable bound and problem 17 where significant progress was made but progress near the solution was very slow. While not solving as many problems as algorithm 12, algorithm 11 was slightly faster on most problems which can be seen by the relative rankings for 25 and 75 percent of the average time ratings where algorithm 11 holds down the top position. Performance on the additional problems was again very good, with progress being made on problems 9 and 13, feasible points were found for problems 21, 22 and 30, and the solution was found for problems 28 and 29.

Algorithm 10 which employs the Broyden-Fletcher-Shanno variable metric technique to generate search directions solved all but four of the twenty-three rated test problems. The problems where difficulty was encountered were all from the Dembo study, problems 17, 18, 20 and 23, and in each case progress was made. For each of these problems the termination was caused by the step size going to zero. Again while the number of problems solved is less than for algorithms 11 and 12, algorithm 10 was extremely fast as can be demonstrated by the high relative rankings at the 25% and 75% average time ratings. In fact, this algorithm produced the fastest time on the majority of the test problems. Performance on the additional test problems was also good. Significant progress was made on problems 9 and 13, a solution to problem 28 was found, and a feasible point was located for problems 21 and 30. However, no significant progress was made on problem 29 and a feasible point could not be located for problem 22.

Algorithm 8 employs the Davidon-Fletcher-Powell variable metric technique to generate search directions. In all twenty of the twenty-three rated test problems were solved, but several problems required a significantly large amount of computational time and only eighteen problems were solved within 250% of the average time placing algorithm 8 directly behind algorithms 10, 11, and 12 for the majority of the relative rankings. The problems where difficulty was encountered were problem number 19 where significant progress was made before the step size went to zero, and problems 23 and 27 where a singular matrix was encountered. The general speed on the majority of the problems was good but from the relative rankings for the 25% and 75% average time ratings it is apparent that algorithm 8 was not quite fast as the other reduced gradient algorithms. The performance on the additional problems was not quite up to par either with significant progress made only on problem 8, and feasible points located for problems 21 and 30.

The overall performance of the reduced gradient algorithms could be rated nothing less than outstanding. The performance was good on both large and small scale problems, and the methods handled equality constrained problems and problems with highly nonlinear constraints with relative ease. Algorithms 11 and 12 ranked first or second in all of the relative ratings and algorithms 10 and 8 were never ranked lower than fifth. Also, the rankings for the reduced gradient algorithms were not effected by an increase in the level of required accuracy as were some of the penalty type methods, another outstanding feature of the generalized reduced gradient algorithms.

4.2.2 The Exterior Penalty Function Methods

In all there were twelve algorithms in this classification, with the majority resulting from optimization packages where several different unconstrained search techniques were applied to the same penalty function. The performance of the exterior penalty function methods was quite varied, with several algorithms performing well while others performed quite poorly and could not be recommended for general use. The algorithms which performed poorly and were eliminated after the initial test problem set were methods 17 and 18. The unconstrained search techniques used for these two methods were univariate search and steepest descent, both of which proved to be very time consuming. Algorithm 17 only solved six of the initial test problems, and for two of the six solved the solution time was well over three times the average time. The method had trouble satisfying the equality constrained problems and even had significant difficulty on problems where several inequality constraints were tight at the solution. Also as the number of design variables increased the time consumed became extremely large. Algorithm 18 fared only slightly better solving seven of the fourteen initial test problems and only on problem 1 was the solution time over three times the average. However progress was still very slow especially on the larger problems and while progress was made on the majority of the problems in the initial test set, difficulty was encountered on equality constrained problems and in final constraint satisfaction for the tight inequality constraints. Of the ten algorithms which made it into the final ratings only six solved over half of the rated test problem set. The algorithms which solved less than half of the problems were methods 14, 16, 20 and 22. Algorithm 14 which uses the conjugate gradient technique of Fletcher-Reeves for the unconstrained minimizations only solved nine of the twenty-three rated problems.

No progress was made on problems 6, 14, 18 and 25, slight progress was made on problems 4 and 27, and significant progress was made on problems 8, 11, 16, 17, 19, 20, 23 and 24. The major difficulty on these problems was the final constraint satisfaction was not adequate. For the additional test problems no progress was made on problems 9, 28, 29 and 30, good progress was recorded on problems 13 and 22, and a feasible point was found for problem 22. Algorithm 22 was also only able to solve nine of the rated test problems. Employing a Hooke-Jeeves pattern search for the unconstrained minimizations, algorithm 22 made no progress on problems 6, 12, 26 and 27, slight progress on problems 14 and 17 and significant progress was recorded on problems 1, 2, 3, 16, 17, 19, 20, 23 and 24. The major difficulties were generally slow progress and the inability to attain constraint satisfaction on both equality and tight inequality constraints. For the additional test problems, significant progress was recorded on problems 9 and 13. Progress and termination at a feasible point were accomplished on problems 21 and 22, and no progress was made on problems 28, 29 and 30. Algorithm 16, which employed a combination of Hooke-Jeeves pattern search for the variables near their bounds and the Simplex method for the others as an unconstrained minimization technique, solved a total of eleven of the rated test problems. With the exception of problems 14 and 27, however, significant progress was made on every problem. Even good progress was made on the equality constrained problems 6 and 16 with the only problem being the final level of equality constraint satisfaction. The algorithm was generally very effective at locating the region of the optimal solution but difficulty was encountered in moving in heavily constrained regions. On the additional test problems the best solutions to problems 9 and 13 were recorded by algorithm 16, and significant progress was made on problems 22 and 29. So while the total number of problems solved by algorithm 16 was not very impressive, the ability to make significant progress on problems was. An algorithm such as method 16 would be valuable to apply to the problems where the gradient methods encounter trouble. Algorithm 20, which also solved 11 of the rated test problems, employs the Fletcher-Reeves conjugate gradient technique for the unconstrained minimizations. Trouble was encountered on the equality constrained problems 6, 16 and 29, but good progress was made on all other problems with the exception of problem 27 where only slight progress was made. On each of problems 7, 17, 18, 19, 20, 24, and 25 good progress was made until the actual vicinity of the optimal solution was reached. Again the major problem encountered was the inability of the code to attain adequate constraint satisfaction. On the additional test problems, significant progress was recorded on problems 9, 13, 21, 22 and 28.

The overall speed of algorithm 20 on the problems solved was quite good with ten of the eleven problems solved within 75% of the average time.

The remaining six algorithms employing an exterior penalty function approach solved at least half of the rated test problem set. Algorithm 19, employing the conjugate direction method of Powell, solved thirteen of the rated test problems. The basic problem with this algorithm was that for most cases it was very slow. Of the thirteen problems solved ten required a longer than average time for solution. For the problems not solved, significant progress was made on problems 14, 16, 17, 19, 20, 24 and 25 but convergence toward the optimal solutions was extremely slow, especially for the problems which had over five design variables. In addition no progress was recorded on problem 27, and premature termination occurred on problems 6 and 26 as the equality constraints were satisfied. On the additional test problems progress was made on problems 9, 13 and 21. Algorithm 13 solved a total of fifteen of the rated test problem set. The pattern search method of Hooke and Jeeves is used to solve the successive unconstrained stages. As far as the number of problems solved, algorithm 13 ranked higher than any other nongradient technique. The computational speed was also good with eleven of the fifteen problems being solved within the average solution time. The only problem where no progress was recorded was problem

27 which contained forty-eight design variables. For the other problems where the optimal solution was not reached, some progress was made on problem 14, significant progress was made on problems 17, 19, 23 and 24, and a local minimum was found for problem 4. Progress was also recorded on several of the additional test problems. Good progress was made on problems 9 and 13, a feasible, near optimal solution was recorded on problems 21 and 22 and significant progress was made on problem 29. The performance of this algorithm was exceptional for a nongradient method and movement was even recorded on all of the equality constrained problems, with the exception of problem 30, which can not be said of most of the other penalty function algorithms. All four of the remaining algorithms employ a variable metric technique for the unconstrained minimizations. Applying the Davidon-Fletcher-Powell technique for the unconstrained optimization stages, algorithm 15 solved a total of fifteen of the rated test problems, ten of which were solved within 75% of the average time at an accuracy level of $\varepsilon_t = 10^{-4}$. The major troubles encountered with algorithm 15 were in the satisfaction of the equality constraints for problems 6, 15, and 27, and in the satisfaction of the tight inequality constraints for problems 17 and 20. Also, no progress was made on problems 14 and 16 and only slight progress was made on problem 25. Performance on the additional test problems was also fairly good with the

solution found for problem 22, and good progress was made on problems 13, 21 and 28. Algorithm 21, which also applies the Davidon-Fletcher-Powell technique for the unconstrained stages, solved sixteen of the rated test problems, fourteen of which were solved within 75% of the average time for an accuracy level of $\varepsilon_t = 10^{-4}$. Troubles encountered again involved constraint satisfaction, with the equality constraints on problems 6, 26, and 27, and with the tight inequality constraints on problems 14, 17, 18 and 23. On the additional problems significant progress was made only on problems 9, 13 and 21. Algorithm 1, again employing the Davidon-Fletcher-Powell technique for the unconstrained minimizations, solved twenty of the twenty-three rated problems. The time required for solution was generally longer than required for algorithms 15 and 21 but the fact that significantly more problems were solved resulted in a higher relative ranking for the ratings above the 75% average time ratings. Seventeen of the twenty problems were solved within 100% of the average time and nineteen were solved within 150% of the average time. Two of the three problems where trouble was encountered involved the satisfaction of the equality constraints. These were problems 6 and 26. On problem 23 good progress was made but the final solution was not reached. For the additional test problems significant progress was reported on problems 13, 21, and 28. The biased penalty function used in algorithm 1, which has the effect of reducing the distortion of the penalty surface at successive stages, appeared to enable the algorithm to satisfy the inequality constraints which are tight at the solution to a better extent than the other exterior penalty function methods. This resulted in ratings close to those of the reduced gradient algorithms at the higher average time ratings. The last of the exterior penalty function methods, algorithm 35 employs the Broyden-Fletcher-Shanno technique for the unconstrained penalty stages. The algorithm solved every problem of the rated test set with the exception of problem 27 where good progress was made. The reason the algorithm did not fare well in the ratings was in the length of time required to reach the solution. Only seven of the problems were solved within 100% of the average time and only fifteen problems were solved within 250% of the average time. Performance on the additional test problems was fair to good with significant progress being made on problems 21, 22 and 28. If the rate of convergence could be improved for this algorithm its relative rating would improve considerably for it solved more problems than all but one algorithm on the rated test set.

The overall performance of the exterior penalty function methods was widely varied but the algorithms which employed a variable metric method for the successive unconstrained stages generally performed well. The basic problem encountered was the ability to achieve constraint satisfaction on problems which had several constraints active at the solution. Difficulty in moving on problems containing nonlinear equality constraints was also a problem common to most of the algorithms. In comparison to the generalized reduced gradient algorithms, the exterior penalty function methods were generally slower on most problems, but several of the algorithms, methods 1, 15 and 21 ranked directly behind the reduced gradient algorithms in most of the ratings. As the allowable level of total error was decreased the gap between the exterior penalty function algorithms and the reduced gradient algorithms widened with the inability to achieve the required level of constraint satisfaction for the penalty function algorithms being the major contributing factor.

4.2.3 The Interior Penalty Function Methods

Seventeen algorithms tested in the comparative study were contained in this general classification. Algorithms 32, 33 and 34 were actually mixed penalty functions using an interior penalty form for all inequality constraints satisfied at the starting point and an exterior form for the initially violated constraints, but were included in this classification. Of the seventeen algorithms, eight were removed from final consideration due to poor performance on the initial test problem set. These algorithms were methods 2, 5, 6, 7, 23, 24, 25 and 30. The specific performance of each of these algorithms will not be discussed but it should be noted that all of these algorithms suffered from slow convergence characteristics, the inability to approach the tight inequality constraints to attain the required level of accuracy in the objective function and trouble in solving the equality constrained problems. Of the remaining nine algorithms, four failed to solve half of the rated test problem set. These algorithms include methods 3, 26, 28 and 29.

Algorithm 3 employed a Hooke-Jeeves pattern search technique to solve the successive unconstrained stages and was only able to solve eleven of the rated problem set. No progress was recorded on problems 6, 15, 23, 26 and 27, a local minimum was found for problem 4 and significant progress was made on problems 1, 12, 14, 17, 20 and 24. For the problems where significant progress was made the difficulty was again the inability to attain a high level of constraint satisfaction. No significant progress was recorded on any of the additional test problems.

Algorithm 26 employs Fletcher-Reeves conjugate gradient technique to handle the unconstrained minimizations. In all eleven of the rated test problems were solved. No progress was recorded on problem 23 due to the inability to find a feasible starting point but good progress was made on problems 5, 6, 14, 15, 17, 18, 19, 24, 25, 26 and 27. Trouble was encountered in satisfying the equality constraints and in attaining sufficient inequality constraint satisfaction. On the additional test problems significant progress was recorded only for problem 9.

Algorithms 28 and 29 were both second order methods and although progress was made on most of the problems the methods were both very slow and tended to terminate prematurely, probably due to the fact that numerically calculated gradients were not accurate enough for the second order methods. Another second order method, algorithm 34 solved thirteen of the twenty-three rated test problems but suffered from the same basic problems as algorithms 28 and 29. For this reason methods requiring second derivatives could not be recommended for general usage unless analytical derivatives are available.

The remaining four algorithms performed significantly better, solving at least seventeen of the rated problems. Algorithm 27, solving seventeen of the rated problem set, employs the Davidon-Fletcher-Powell variable metric technique to handle the unconstrained stages. No progress was made on problem 8, due to the fact that a feasible starting point could not be located, but significant progress was made on problems 5, 6, 7, 19 and 26. On three of these problems the difficulty was in satisfying the equality constraints to the required level. The fact that algorithm 27 was able to solve problem 27 should be noted for this was the only interior penalty method to attain the solution to this forty-eight variable problem. The relative speed of the method was also quite good. Fourteen of the seventeen problems solved were solved within 100% of the average time for the total error rating of 10^{-4}. This ranked algorithm 27 in the same group as several of the better exterior penalty functions in most of the ratings. Performance on the additional test problems was not very good, however, with significant progress only reported on problem 9.

Algorithm 31 also employed a variable metric technique to handle the successive unconstrained stages and was able to solve seventeen of the rated test problems. However, only five of these problems were solved within 100% of the average time and only thirteen within 250% of the average time. For this reason the algorithm was not ranked very high in any of the average time ratings. For the problems not solved, no progress was made on problem 27, a local minimum was found for problem 4 and slow but significant progress was recorded on problems 6, 12, 17 and 26. Performance on the additional test problems was better than for any other interior penalty method with a solution recorded for both problems 21 and 22 and significant progress made on problem 9.

Algorithm 32 employs Powell's method of conjugate directions for the unconstrained minimizations and was able to solve eighteen of the rated problems. Progress was again slow however, with only four problems being solved within 100% of the average time. In the 250% rating fifteen problems were solved for $\varepsilon_t = 10^{-4}$, ranking the method with several of the better exterior penalty type methods. No progress was recorded on problems 6 and 27, slight progress was made on problem 26 and significant progress was made on problems 15 and 20. Thus four of the five problems where the solution was not reached involved equality constraints. For the additional test problems excellent progress was made on problem 9, and significant progress was made on problems 21 and 22 due mainly to the fact that the initially unsatisfied inequality constraints were handled by the exterior penalty term.

Algorithm 33 employed the Broyden-Fletcher-Shanno variable metric technique for the unconstrained stages and was able to solve eighteen of the rated test problems. The penalty function employed was the same as for algorithm 32 and the relative performance ratings are quite similar with the exception being that algorithm 33 was generally faster and ranked fairly well for both the 150% and 250% average time ratings for $\varepsilon_t = 10^{-4}$. The problems where difficulty was encountered were problem 6 where no progress was recorded and problems 11, 17, 20 and 27 where good progress was made. Progress on the additional test set was recorded on problems 9, 13, 21 and 22.

As with the exterior penalty function algorithms the basic difficulty with the interior penalty methods was the inability to attain satisfactory constraint satisfaction. The problem is only different in the fact that the constraints are being approached from the feasible region and the difficulty is not in violated constraints but in not attaining the required degree of accuracy in the objective function. Also, as with the exterior penalty function algorithms, most algorithms had great difficulty with equality constrained problems. An additional problem was encountered on several of the highly constrained problems in simply locating a feasible starting point. Timewise several of the algorithms approached the speed of the better exterior penalty methods but in general the interior penalty methods were slower.

4.3 Factors Affecting the Results

The results presented in the last chapter clearly indicate that the linearization type methods and in particular the generalized reduced gradient algorithms are significantly faster, and in most cases significantly more robust than the penalty function methods. To ensure that it was not the methodology of collecting the data which led to this conclusion several additional factors will be considered. The selection of the system compiler will be considered since it has recently been demonstrated that the optimization level of a compiler may significantly affect the relative speed of different algorithms [48]. Also the effects of the variation of the parameters on the total solution time will be considered. The input parameters were not varied in order to produce any time savings and were only adjusted when trouble was encountered during the solution procedure or when the final solution was not acceptable. This adjustment of the input parameters could have had some effect, however, on the reported solution times, and the sensitivity of the input parameters about the final values for several problems will be considered for a sampling of the algorithms. Finally the percentage of the total time spent within the various computational stages will be considered for both the linear approximation and penalty function methods in an effort to determine why the linear approximation methods and in particular the generalized reduced gradient algorithms have faster convergence characteristics.

4.3.1 Compiler Selection

Several different compilers are available at the Purdue University Computing Center. The FUN compiler which is a merger between the original fortran compiler for the 6000 series machines written by CDC and an updated fortran compiler from CDC was selected because it uses extensive statement and code optimization. While requiring a large amount of compilation time, the compiled binary instructions will execute faster. The increased compilation time was not important since each algorithm tested in the study was compiled and stored in binary form and only the subroutines pertaining to the specific problem had to be compiled continually. To investigate relative time differences which could occur by using other compilers, four algorithms, a generalized reduced gradient algorithm, an interior penalty function method, an exterior penalty function method and a repetitive linear programming method were applied to four problems on two other compilers. The other compilers consisted of the FTN compiler which is an early version of CDC's latest fortran compiler which produces a reasonably efficient code and the MNF compiler which is a user oriented compiler for the 6000 series machines which was developed by a group of staff members at the University of Minnesota. The ratio of the time required for the exterior penalty method, the interior penalty method and the repetitive linear programming method to the time required for the reduced gradient algorithm for each of the compilers on the four test problems is presented in Table 4.1. From the table it is apparent that the ratios vary from compiler to compiler but the amount of variation is quite small. No trend is evident that the FUN compiler favored the reduced gradient algorithms over the other methods, in fact in several cases the time ratios increase by over 25% while the time ratios never decrease by even 10% indicating that by using either the FTN or MNF compilers might have actually increased the relative performance of the reduced gradient algorithms.

Table 4.1 Ratios of Solution Times for Algorithms 1, 9 and 27 to Algorithm 11 for Various Compilers.

	Problem			
Compiler	14	15	18	24
FUN	5.36	7.68	7.79	3.72
MNF	5.68	7.10	8.88	4.47
FTN	5.13	7.69	8.46	3.68

Algorithm 1

	Problem			
Compiler	14	15	18	24
FUN	4.01	10.42	1.59	13.23
MNF	4.05	10.13	1.70	14.58
FTN	5.13	13.99	2.13	12.11

Algorithm 27

	Problem			
Compiler	14	15	18	24
FUN	.717	1.32	.820	.654
MNF	.810	1.42	.997	.700
FTN	.793	1.39	1.04	.644

Algorithm 9

4.3.2 Variation of Parameters

To study the effects on the solution time due to variations in the input parameters about their final values, four test problems were selected from the study. On each of these problems the input parameters for algorithms 1, 9, 10, 11, 15, 21, 27 and 31 were varied about their final values. The final values are the values of the input parameters used for the accepted run for each problem. The problems were selected for their wide range in the number of design variables and in the number and type of constraints. The selected problems include problem 14 which contains fifteen design variables, five functional inequality constraints and fifteen variable bounds, problem 15 which contains sixteen design variables, eight equality constraints and thirty-two variable bounds, problem 18 which contains seven design variables, fourteen functional inequality constraints and fourteen variable bounds, and problem 24 which contains four design variables, five functional inequality constraints and three variable bounds. The algorithms were selected so as to include several of the better algorithms from each general classification. Algorithms 9, 10 and 11 are linearization methods with algorithm 9 representing the successive linear programming technique and algorithms 10 and 11 representing the generalized reduced gradient methods. Algorithms 1, 15 and 21 represent the exterior penalty function methods and algorithms 27 and 31 represent the interior penalty function methods. All time data was recorded at a total relative error, ε_t, of 10^{-4}, unless it is stated otherwise.

The input variables to algorithm 9 included the initial step size, the final step size and the increment with which to calculate the partial derivatives. The percentage change in the solution time from the run accepted for the final results is presented in Table 4.2 for algorithm 9 on the four test problems. A dashed line for any of the parameters indicates that this is the value used in the final results. Of the parameters effecting the solution time the initial step size was found to be very sensitive, and small changes resulted in thirty to forty percent deviations in the solution time. However, the trend was extremely problem dependent with no common trend encountered. It should be realized that by increasing or decreasing the initial step size the path followed to the solution is altered which can drastically effect the solution time. The minimum step size was included in Table 4.2 to demonstrate how the total solution time was effected. The percentage change in the total solution time was used as a comparison since altering the minimum step size would not change the time to reach the required accuracy level of $\varepsilon_t = 10^{-4}$. It is apparent from Table 4.2 that as the minimum step size decreases the total solution time increases significantly, pointing out the fact that for general usage

Table 4.2 Percentage Change in Solution Time for Algorithm 9 for Variations in the Input Parameters.

Problem	Initial Step Size				
	.50	1.25	2.50	5.00	10.0
14	+15.4	- 3.77	—	-31.5	-11.0
15	-16.3	-26.6	—	-22.9	+ 7.70
18	- 2.92	+ 6.40	+18.42	+42.3	—
24	+30.4	—	+23.4	+22.3	+ 7.34

Problem	Minimum Step Size				
	.001	.005	.01	.05	.10
14	+686	+129	+50.5	—	No soln
15	+ 45.4	+ 8.4	—	No soln	No soln
18	+534	—	No soln	No soln	No soln
24	+218	—	- 1.06	-1.02	No soln

Problem	Linearization Increment				
	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
14	-6.81	-7.25	—	+ .49	+ .84
15	- .42	- .06	—	- .42	- .71
18	+ .86	+ .27	—	+ .63	+2.40
24	- .80	-2.60	—	-1.86	No soln

the minimum step size should be chosen with care and not specified as an extremely small value. The final input parameter for algorithm 9 was the size of the linearization increment and was not found to cause any significant change in the solution time. The general conclusion then is that the times recorded for the test study could be altered by 30% to 40% by variation of the input parameters, and the variation could either increase or decrease the reported solution times.

Table 4.3 presents the percentage change in the solution time for the reduced gradient algorithm 10 for changes in the variable bound satisfaction criteria, the constraint satisfaction criteria and the basis pivot criteria. All other programming parameters were handled internally to the algorithm. Results for problem 18 are not included in Table 4.3 since no solution was recorded for this algorithm. The general trend, as should be expected, is that as a higher degree of bound or constraint satisfaction is desired a resulting increase in the solution time is noted. The ability to select the level of constraint satisfaction was unique to the reduced gradient algorithms and made parameter adjustment almost unnecessary. The other parameter, the basis pivot criteria, determines the minimum value of a pivot element in the generation of the basis inverse. The suggested value of 10^{-3} was used throughout the study and

Table 4.3 Percentage Change in Solution Time for Algorithm 10 for Variations in the Input Parameters.

Problem	Bound Satisfaction Criteria 10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}
14	No soln	+10.6	—	-20.2	-16.3
15	- .36	- .20	—	- .27	- .59
24	+25.4	—	-11.2	-30.3	No soln

Problem	Constraint Satisfaction Criteria 10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}
14	No soln	No soln	—	-6.11	-14.5
15	+ .10	- .64	—	- .10	+ 1.93
24	-1.12	-1.60	—	- .22	- .43

Problem	Basic Pivot Criteria 10^{-6}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
14	+ .84	+ .72	—	+1.03	+ .55
15	+ .39	+ .32	—	- .21	+ .43
24	- .45	+1.80	—	+2.69	+15.34

from Table 4.3 it can be seen that this parameter had little effect on the solution time. The other reduced gradient algorithm tested for input parameter variations was algorithm 11. For this algorithm the input parameters involved the line search criteria, the constraint satisfaction criteria and the forward difference increment for the calculation of numerical derivatives. The results for variations in these parameters are presented in Table 4.4. The line search criteria was found to effect the solution time to some extent but not by more than ten to fifteen percent over the range tested. A value of 10^{-4} was successful on the large majority of problems but occasionally the line search criteria was decreased to achieve a higher level of accuracy in the solution. This was the case for problem 14. The constraint satisfaction criteria followed the same trend as for algorithm 10 requiring a greater amount of time for an increased level of constraint satisfaction. The final input parameter, the forward difference increment was found to produce only insignificant differences in the solution times for all reasonable values. Variation of the input parameters for the reduced gradient algorithms point out an increase in the amount of control the user has for a problem and seldom was more than one run required for each problem on these algorithms.

No standard set of input parameters was found to work well for the exterior penalty function methods. For

Table 4.4 Percentage Change in Solution Time for Algorithm 11 for Variations in the Input Parameters.

	Line Search Criteria				
Problem	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
14	+14.6	-10.2	-6.8	—	-1.32
15	+12.6	—	-9.6	+14.3	+4.9
18	+ 1.10	—	+ .20	- .50	+2.68
24	- .17	—	-2.1	+ .70	+1.04

	Constraint Satisfaction Criteria				
Problem	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
14	No soln	No soln	-2.30	—	+11.8
15	-18.9	-20.6	-1.4	—	- 1.2
18	No soln	No soln	-4.8	—	+ 4.6
24	No soln	-20.6	-7.3	—	+29.0

	Forward Difference Increment				
Problem	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
14	-2.9	+1.1	-1.3	—	-3.1
15	+1.1	+ .70	+ .90	—	+ .04
18	No soln	0.0	- .25	—	+9.9
24	-2.3	-2.3	-1.2	—	- .86

algotihm 1 the required input parameters included the line search criteria, the initial penalty parameter, and the forward difference increment. No penalty multiplication factor for each stage was required because algorithm 1 employed a biased penalty function which maintains a constant value for the penalty parameter. The percentage change in the solution time for variations in these parameters is presented in Table 4.5. The method can be seen to be quite sensitive to the line search criteria with wide variations possible in the solution time. The penalty parameter determined to a large extent whether a solution was found to the required level of accuracy. This was due to the fact that to satisfy the inequality constraints to the required level the penalty parameter had to be increased and for several problems the line search criteria had to be decreased to provide for a more accurate solution. Generally this did not seriously effect the solution time recorded but as can be seen from Table 4.5 some large variations were possible for problem 15. Problem 18 was not included in the parameter study since the penalty parameter had to be raised to an extremely large value to achieve any solution and with such a large penalty parameter the algorithm acted more as an interior method than an exterior method. The forward difference increment for this algorithm had to be maintained at a relatively small value to insure a solution. Algorithm 15 was unique in the sense that all parameters were set

Table 4.5 Percentage Change in Solution Time for Algorithm 1 for Variations in the Input Parameters.

Problem	Line Search Criteria				
	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
14	No soln	-22.9	+109	—	+ 7.80
15	-43.4	-28.4	+ 5.70	—	+12.60
24	No soln	+ 1.29	—	+11.2	+20.3

Problem	Penalty Parameter			
	25	50	100	200
14	-3.04	—	No soln	+13.1
15	No soln	+105	—	- 9.30
24	No soln	+ 7.3	—	- 2.58

Problem	Forward Difference Increment				
	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
14	No soln	No soln	No soln	+27.3	—
15	-19.1	-6.80	-9.70	- 5.60	—
24	No soln	-2.10	-1.72	- 6.00	—

internally, and for the initial run the user did not have to select any values. The initial penalty parameter is calculated so that at the first stage the penalty function is appreciably more sensitive to the objective function than the constraints. The internally selected parameters worked very well for the problems in the test set and rarely did any parameter have to be changed. To determine how sensitive the penalty function parameter, the penalty multiplication at each stage and the forward difference increment were to variations in the parameters from the values selected internally a parameter study was conducted. The results appear in Table 4.6. Potential time savings from decreasing the number of stages required by increasing the initial penalty coefficient or the penalty multiplication at each stage were canceled out by an increase in the time required per stage. In no case did the solution time fluctuate by more than twenty percent, so the internal selection of parameters proved to be quite sufficient. The success of the selection of internal parametesr is probably due to the fact that the constraints are scaled at the beginning of each stage to reduce the domination of the penalty function by any one constraint and allow for a uniform set of program parameters to be selected. The same three parameters, the initial penalty coefficient, the penalty multiplication factor at each stage and the forward difference increment were required for algorithm 21. The

Table 4.6 Percentage Change in Solution Time for Algorithm 15 for Variations in the Input Parameters.

Problem	Penalty Function Parameter (FP)			
	%/FP	%/FP	%/FP	%/FP
15	—/10^{-4}	-4.67/10^{-3}	-3.70/10^{-2}	+8.88/10^{-1}
18	—/.15	+7.27/1.0	+14.7/10.0	+18.7/20.0
24	—/10^{-6}	-8.40/10^{-5}	-6.72/10^{-3}	-6.70/10^{-1}

Problem	Penalty Multiplication at Each Stage			
	8	15	30	45
15	—	-17.8	No soln	No soln
18	—	- .82	-8.36	-18.9
24	—	- 5.90	-1.03	- 2.24

Problem	Forward Difference Increment		
	10^{-10}	10^{-7}	10^{-5}
15	—	+ .44	+ 2.36
18	—	+ .07	+12.10
24	—	+6.19	No soln

percentage change in the solution time for variation in these parameters is presented in Table 4.7. Again no variations of over 25 percent were noted for changes in the initial penalty coefficient or the penalty multiplication factor, but as for algorithm 1, both parameters had a significant effect on whether the solution was located to the sufficient degree of accuracy. This again was due to the fact that a combination of the initial penalty coefficient and the multiplication factor had to be determined which would achieve constraint satisfaction to the required level of accuracy. This adjustment procedure usually involved several trial runs. The forward difference increment was again required to be fairly small, another common factor for the exterior penalty function methods.

For the interior penalty function methods the suggested input parameters had significantly more success than for the exterior penalty function methods. This was due to the fact that the inequality constraints were approached from the feasible region and no parameter adjustment was required because of slightly violated inequality constraints at the solution. Algorithm 27 was the interior counterpart to algorithm 21. The same input parameters were required with the exception that a the penalty coefficient multiplication factor is required to be less than one which actually produces a penalty coefficient reduction factor. The percentage change in the solution time for

Table 4.7 Percentage Change in Solution Time for Algorithm 21 for Variations in the Input Parameters.

	Penalty Function Parameter					
Problem	.01	.50	1.0	10.0	50.0	100.0
14	No soln	No soln	-10.5	- 1.56	—	-20.3
15	+5.27	—	- 9.50	- 7.32	No soln	No soln
24	No soln	No soln	—	+28.9	+19.1	+21.4

	Penalty Multiplication at Each Stage				
Problem	10	20	50	100	200
14	+1.80	No soln	+ 7.45	—	-11.0
15	No soln	No soln	+24.4	—	No soln
24	No soln	No soln	No soln	—	-13.3

	Forward Difference Increment			
Problem	10^{-4}	10^{-6}	10^{-8}	10^{-10}
14	No soln	-29.4	-9.10	—
15	No soln	No soln	No soln	—
24	No soln	No soln	+7.10	—

variations in the input parameters is presented in Table 4.8. Variations of thirty to forty percent are possible by adjusting the initial penalty coefficient or the multiplication factor but these large variations represent an increase in the solution time not a decrease. Only time savings of up to fifteen percent were noted by parameter variations which would indicate that the standard input parameters are quite good for this algorithm at least on these test problems. The forward difference increment was again required to be on the order of 10^{-7} to allow for an efficient solution. For algorithm 31 the same three parameters were required as for algorithm 27 plus an additional parameter regarding the subproblem convergence criteria. The effect on the solution time resulting from variations in these parameters is presented in Table 4.9. The results are much the same as for algorithm 27 with large increases in solution times resulting for most combinations, although for problem 24 a possible forty percent reduction in solution time did result for a reduction in the initial penalty coefficient. The overall results from Table 4.9 indicate that the suggested parameters work well for this algorithm also. The subproblem convergence criteria was not found to significantly effect the solution time and the forward difference increment was fairly insensitive over the range from 10^{-4} to 10^{-7}.

Table 4.8 Percentage Change in Solution Time for Algorithm 27 for Variations in the Input Parameters.

	Penalty Function Parameter			
Problem	.005	.05	.50	5.00
14	- 4.70	+ 6.80	—	+17.8
18	-11.7	-15.0	—	+30.0
24	- 4.40	- 7.50	—	+36.8

	Penalty Multiplication Factor				
Problem	.01	.05	.10	.25	.50
14	-14.9	- 6.70	—	+ 3.15	+17.4
18	+ 7.02	+10.30	—	- 3.40	+30.3
24	+ .06	- 4.50	—	+24.2	+44.0

	Forward Difference Increment		
Problem	10^{-7}	10^{-5}	10^{-3}
14	—	+31.7	No soln
18	—	+29.0	No soln
24	—	+22.1	No soln

Table 4.9 Percentage Change in Solution Time for Algorithm 31 for Variations in the Input Parameters.

	Penalty Function Parameter			
Problem	.01	.10	1.0	10.0
14	+484	+95.3	—	-15.5
18	+ 22.7	+ 1.94	—	+36.2
24	+ 38.5	-40.2	—	No soln

	Penalty Reduction Factor				
Problem	8	12	16	20	24
14	- 4.76	-2.86	—	- .30	+1.02
18	+ 9.61	+6.50	—	+1.24	-6.35
24	+26.24	+5.02	—	-3.78	+ .07

	Subproblem Convergence Criteria			
Problem	10^{-7}	10^{-6}	10^{-5}	10^{-4}
14	+.08	—	+.41	-.53
18	+.08	—	+.12	0.0
24	+.04	—	-.02	+.50

	Forward Difference Increment			
Problem	10^{-7}	10^{-6}	10^{-5}	10^{-4}
14	+7.53	—	-19.1	+1.64
18	-5.34	—	- 4.40	+8.59
24	-4.73	—	+ 7.84	+8.94

The frequently significant changes in the solution times which were observed for variations in the input parameters demonstrate the importance these parameters have in the solution procedure. If the results were based on the best time for each algorithm for each problem time savings of from twenty to thirty percent could be expected from the times reported in the study. However, this type of time data would not be at all representative of what the average user might expect, and even if the best times were used for the relative ratings there is no evidence that the penalty function algorithms would even approach the level of performance demonstrated by the generalized reduced gradient algorithms. So the major result of the parameter study was to demonstrate that the generalized reduced gradient algorithms and the interior penalty function methods seem to perform well with the suggested input parameter values, while the exterior penalty methods generally require some manipulation of the initial penalty coefficient or the penalty multiplication factor at each stage in general usage.

4.3.3 Time Study for Computational Phases

During the solution process for each algorithm several distinct computational stages are employed. If improvement of an algorithm is to be attempted, it would be most promising to concentrate on the computational stage from which the greatest savings in time could result. Also by comparing the time spent in the various computational stages for both the linearization and the penalty function algorithms an indication of where the linearization methods achieve their savings might result. To generate this type of information the basic computational phases were determined for the algorithms which performed well in the study and timing routines were inserted in these algorithms to find out what portion of the total solution time is spent in each phase. The algorithms were then run on the four problems which were used in the parameter study.

For algorithm 9, a repetitive linear programming method, the only major computational phases were the generation of the linear programming problems, the solution of the linear programming problems, and the adjustment of the variables by Newton's method which is relevant only for problems involving equality constraints. The total time required to solve each of the four test problems and the percentage of the total time spent in each computational phase for algorithm 9 is presented in Table 4.10. The results, as might be expected are highly problem dependent, but the interesting fact is that the percentage of the total time spent in generating the linear programming problems is greater than or equal to the time spent in solving the linear programming problems except for problem 24 which contained only four variables and five functional constraints. If the solution of the linear programming problems had required the large majority of the solution time the number

Table 4.10 Percentage of Time Spent in Various Phases of the Solution Procedure for Algorithm 9.

Problem Phase	Percentage of Total Time 14	15	18	24
Generating L.P. Problems	58.2	86.7	48.7	33.4
Solution of L.P. Problems	39.4	9.1	47.9	64.9
Newton's Method	0.0	2.5	0.0	0.0
Total Time (sec)	5.673	13.248	3.451	.400

of constraints would probably be the important factor in the total time required for a repetitive linear programming algorithm since the number of constraints is the determining factor in how long a linear programming problem solution requires [49]. However, since the generation of the linear programming problems requires the calculation of the partial derivatives of the objective function and the constraints with respect to the design variables the product of the number of design variables and the constraints would become the deciding factor to the solution time required.

For a reduced gradient algorithm there are many major computational phases involved in the solution process. These phases include the generation of the partial derivatives, matrix inversion and basis changing, the calculation of the reduced gradient, the calculation of the search direction and the time spent in the line search. Of the time spent in the line search a certain portion is spent in Newton's method to maintain constraint satisfaction. Many of these phases could be considered as part of the total generation of the search direction but each phase was accounted for separately to provide for the best division of time. The total time required and the percentage of time spent in each of these computational phases for algorithms 10 amd 11 are presented in Table 4.11. The major portion of the time spent for both of the algorithms is in the generation of the partial derivatives, the matrix

Table 4.11 Percentage of Time Spent in Various Phases of the Solution Procedure for Algorithms 10 and 11.

Problem Phase/Algorithm	Percentage of Total Time							
	14		15		18		24	
	10	11	10	11	10	11	10	11
Generating Constraint Partials	34.1	30.5	61.4	31.7	21.4	8.3	15.7	11.8
Matrix Inversion and Basis Changing	13.2	4.0	10.3	6.6	20.7	45.3	13.9	10.8
Calculation of Reduced Gradient	2.1	12.9	1.8	35.4	2.9	2.7	2.4	3.0
Calculation of Search Direction	4.4	1.6	2.6	1.4	4.5	1.4	3.1	1.4
Line Search	46.1	42.7	23.9	18.9	50.5	29.6	64.4	71.3
Percentage of Line Search Time Spent in Newtons Method	81.4	79.3	71.0	30.8	50.6	83.1	75.3	80.5
Total Time (sec)	4.471	8.101	4.997	8.943	.319*	3.616	.445	.575

*Very little progress

inversion and basis changing and in the line search. Algorithm 11 also spent a significant portion of time in the calculation of the reduced gradient for problems 14 and 15. An interesting point to note is that the majority of the time spent in the line search was actually used in the satisfaction of the constraints by Newton's method. Again the percentage of time spent in each phase of the algorithm was highly problem dependent and also varied between the two reduced gradient algorithms. This points out the fact that the computational procedures employed to execute each phase can effect the relative solution times of two algorithms which are theoretically almost identical.

For both the interior and exterior penalty function algorithms the solution procedure may be divided into two general phases. These phases are the generation of the search direction and the line search. Other phases such as the updating of the penalty parameters or for several algorithms the extrapolation of successive solutions were present but the time spent in these phases did not account for any significant percentage of the total solution time. Table 4.12 presents the total time and the percentage of time spent in these computational phases for the exterior penalty function methods 1, 15 and 21, and Table 4.13 presents the same information for the interior penalty function methods 27 and 31. A basic trend for both the interior and exterior penalty function algorithms is that as the number of variables increased the time spent in generating the search directions

Table 4.12 Percentage of Time Spent in Various Phases of the Solution Procedure for Algorithms 1, 15 and 21.

Problem	Percentage of Total Time											
	14			15			18			24		
Phase/Algorithm	1	15	21	1	15	22	1	15	22	1	15	22
Calculation of Search Direction	67.7	87.0	52.9	75.0	79.4	73.8	37.9	71.0	42.3	33.8	54.0	33.5
Line Search	29.6	11.1	46.8	24.2	20.4	22.1	49.3	28.8	56.1	60.2	42.6	63.7
Total Time (sec)	40.08	.270*	31.88	62.59	137.7	62.26	32.0	10.99	7.05$^+$	2.33	3.218	1.263

*No progress

+Significant progress

Table 4.13 Percentage of Time Spent in Various Phases of the Solution Procedure for Algorithms 27 and 31.

Problem	Percentage of Total Time							
	14		15		18		24	
Phase/Algorithm	27	31	27	31	27	31	27	31
Calculation of Search Direction	50.9	54.6	65.2	67.0	43.1	36.5	10.2	27.9
Line Search	47.5	43.9	34.1	31.4	55.9	61.5	87.2	67.5
Total Time (sec)	29.85	47.73	93.57+	200.2	5.95	19.41	9.320	11.731

+Significant progress

became the major element of the total solution time. The calculation of the search direction is largely composed of the time to generate the partial derivatives of the penalty function with respect to the design variables so no major time savings could be expected for this phase in future algorithm development. Time savings for the penalty function algorithms would have to be achieved by reducing the time spent in the line search, and the time savings would have to be extremely large to have any significant effect on the total solution time. The relatively small solution times for the reduced gradient algorithms as compared to the penalty function algorithms must then be related to the rate of overall convergence rather than to a significant time difference in any computation phase of the algorithms. There is apparently a significant time savings inherent in handling the constraints directly. If all the phases but the time spent in the line search are grouped to form the time to generate the search direction for the reduced gradient algorithms, which is not precisely the case since the inverse of the basis is also used in the line search in Newton's method, the percentage of the total time spent in each phase is not that different from the penalty function algorithms. The major difference is that for the reduced gradient only a small portion of the line search time is spent in the actual line searching since the majority of the line search time is consumed by Newton's method. A comparison between the repetitive linear programming algorithm and the penalty function algorithms is difficult, but the percentage of the total time spent generating the problem for the repetitive linear programming algorithm is roughly equivalent to the time spent in generating the search directions for the penalty function algorithms.

CHAPTER 5 AN ALTERNATE APPROACH OF COMPARISON

5.1 Introduction

The results of the comparative study so far have been based on the number of problems solved within a reasonable amount of computational time. The problems in the test set were intentionally selected to be widely varied in nature since the performance on the test set was to indicate the performance one would expect in general usage. It would still be beneficial, however, to have additional information pertaining to the performance of some of the better algorithms on a specific type of problem. This type of information cannot be generated from the results on the selected test problems because very few of the test problems were closely enough related to make much of a performance judgement on a specific type of problem. To obtain this information, a different approach was taken. The performance of the algorithms was rated on the basis of how the solution time varied as a function of the type of problem considered. A problem containing five variables and ten inequality constraints with a quadratic objective function and constraints was selected as the standard problem. This standard problem was then altered by changing one problem factor such as the number of variables, the number of inequality constraints, the number of equality constraints or the degree of nonlinearity to form another problem class. The solution times reported for each problem class are the average time for each algorithm on a set of ten randomly generated problems within that problem class to an accuracy of $\varepsilon_t = 10^{-4}$.

5.2 Problem Generation

It would be desirable to base the problems on some practical engineering applications, but to find ten problems with the same number of variables, constraints and general degree of nonlinearity is not at all an easy task. If the additional restriction is imposed that the distance from the starting vector to the solution vector remains constant, the task of gathering the problems becomes even more difficult. For this reason the problems were selected to have a convenient mathematical form. The selection of a quadratic objective function and quadratic constraints for the standard problem has several distinct advantages. First of all, all of the algorithms tested were able to solve this type of problem with the recommended values for the input parameters so only one run per problem was required. Also the degree of nonlinearity can be raised simply by considering higher order terms such as cubic or quartic terms. In addition the quadratic functions may easily be selected so that the constrained region is a convex set, which would guarantee the presence of only one optimal solution.

The test problems were generated following the procedure of Rosen and Suzuki [50]. For example consider the standard test problems consisting of five variables and ten constraints. The quadratic form for the objective function may be expressed as

$$f(\bar{x}) = \bar{x}^T Q_o \bar{x} + a\bar{x} \tag{5.1}$$

and for each constraint as

$$g_i(\bar{x}) = \bar{x}^T Q_i \bar{x} + b_i \bar{x} + c_i \geqq 0; \; i = 1, 2, \ldots, 10 \tag{5.2}$$

For these expressions the Q_o and Q_i are randomly generated N by N matrices, or five by five matrices for this case. Q_o is forced to be positive definite to guarantee unimodality and the Q_i are forced to be negative definite to guarantee a convex feasible region. The a, b_i and c_i are all column vectors containing N elements. Not only are the Q matrices selected but the b_i vectors, and the Lagrange multipliers are randomly generated, and the solution vector is also selected. The Lagrange multipliers are either set to zero if a constraint is not to be active at the solution or to a random number between .5 and 10. So that the problem is not unconstrained or overconstrained the number of constraints allowed to be active at the solution was also selected as a random integer between one and N-1.

Now the a_i and c_i may be determined by the conditions required to make the selected optimal vector a Kuhn Tucker point. Thus a sequence of ten problems which have a common number of variables and constraints, a quadratic objective function and constraints, a common starting vector and solution vector, and a convex feasible set with a unimodal function may be easily generated.

The procedure to guarantee that Q_o is positive definite and the Q_i were negative definite was not very complicated either. The Q matrices were generated by submatrices with an additional row and column added to each submatrix to form the next. In this fashion a one by one matrix could be generated randomly to form a positive definite matrix. For a one by one matrix all that is required is that the element be positive. Next a randomly generated row and column are added and continually regenerated until the determinant of the two by two matrix is positive. By continuing the procedure a row and column at a time with positive elements in all diagonal locations a positive definite matrix can be built up gradually. In this fashion only one row and column need be regenerated for each submatrix which was far more efficient than regenerating an entire N by N matrix until a positive definite matrix resulted, an occurrence which may never occur with randomly generated coefficients. The procedure for generating negative definite matrices was built on the same princple.

The procedure for the problem generation may be easily extended to generate variations of the standard problem. An increase in the number of inequality constraints simply requires the generation of additional Q_i matrices, and an increase in the number of variables only increases the size of the matrices. The basic procedure for the addition of equality constraints remains unchanged with the exception that the lagrange multipliers for the equality constraints may be positive or negative. However the addition of nonlinear equality constraints introduces the possibility of local minima. This problem was handled by including only those problems generated where all of the algorithms reached the selected optimal vector. For the increase in nonlinearity additional higher order terms were added to the basic quadratic form. No change was required for the problem generation with the exception of the check for positive or negative definiteness. The matrix of second derivatives no longer consisted of constant elements and computationally there was no easy way of guaranteeing that the functions were postive or negative definite. To circumvent this problem, the Q matrices were generated as for the quadratic functions and the additional terms were then added without an additional check as to the positive or negative definiteness of the matrices. This introduced the possibility of producing a nonconvex feasible region but as for the addition of equality constraints the problems where alternate optimal solutions were found were not included.

The starting vector for all problems was the origin and the solution for all of the five variable problems was $x_i = 2.0$. As the number of variables increased, however, the solution was adjusted so that the distance from the origin to the solution vector was the same as for the five variable problem. The performance of the algorithms will now be considered for each variation of the standard problem.

5.3 Increase in Design Variables

An increase in the number of design variables was by far the most critical factor in increasing the solution time for all of the algorithms. The average solution times for the seven selected algorithms on the standard test problem set along with the average solution times for the 10 and 15 variable problem sets are presented in Table 5.1. Also included in Table 5.1 is the percentage increase in solution time over the standard problem set for the ten and fifteen variable problem sets. For the five, ten and fifteen variable problem sets the linear approximation methods were significantly faster than the penalty function methods, with the reduced gradient algorithms 10 and 11 producing solution times on the order of one half to one third the time required by the penalty function algorithms. The solution times for the repetitive linear programming

Table 5.1 Solution Time and Percentage Increase in Solution Time for an Increase in the Number of Design Variables.

Algorithm	Number of Variables		
	5 time	10 time/% increase	15 time/% increase
1	6.306	23.482/272.4	70.288/1014.6
9	4.153	16.356/293.8	42.865/932.1
10	2.171	10.217/307.6	23.307/973.6
11	3.491	15.025/330.4	26.318/653.9
15	9.081	42.248/365.2	132.320/1357.6
21	6.639	33.637/406.7	75.191/1032.6
31	9.050	32.650/260.8	128.086/1315.3

algorithm, method 9, were slightly slower than those produced by the reduced gradient algorithms, especially on the problems involving fifteen design variables, but again produced considerable time savings over the majority of the penalty function algorithms. These results are in direct agreement with the results previously generated from the comparative study.

An interesting point is that the percentage increase in time was not that different for the various classes of algorithms. The increase in the number of design variables from five to ten produced a percentage increase of slightly under 300% for algorithms, 9 and 31, and an increase of from 330 to slightly over 400% for the other algorithms. The increase to fifteen design variables generally produced a percentage increase in solution time of approximately 1000% over the five variable problem set. The exception was algorithm 11 which only produced an increase of approximately 650%. This drastic increase in the solution time for an increase in the number of variables indicates that all of the nonlinear programming algorithms are extremely sensitive to the number of design variables and as the number of design variables is increased the computational time saved by applying the reduced gradient algorithms over the penalty function algorithms becomes increasingly significant. This can be seen clearly if the results are extrapolated to higher values of N. For an increase to 100 design variables, the extrapolated time for the penalty function algorithms ranges from 3000 to 13,000 seconds on the CDC 6500 while the generalized reduced gradient algorithms have extrapolated solution times of under 1000 seconds.

5.4 Increase in Inequality Constraints

Figure 5.2 presents the solution time and the percentage increase in solution time as the number of inequality constraints was raised from ten to fifteen to twenty. The number of constraints allowed to be active at the solution was held between one and four as for the standard problem to prevent an overconstrained solution. The effect on the solution time for an increase in the number of inequality constraints was not as drastic as for an increase in the number of variables. Again the linear approximation algorithms only required one half to one third the amount of computational time the penalty function algorithms required for all cases. For the penalty function algorithms doubling the number of constraints produced an increase of approximately one hundred percent in the solution time, while the linear approximation algorithms were slightly less effected with the exception of algorithm 11 which demonstrated the largest percentage increase in solution time out of all of the algorithms. Upon investigation it was found that algorithm 11 employed Newton's method to adjust all of the slack variables instead of just the slack variables for the tight constraints. This resulted in increasing the amount of computational effort needlessly as the number of constraints increased, due to the increased size of the matrix to be inverted. Another interesting point is that the method of repetitive linear programming was the

Table 5.2 Solution Time and Percentage Increase in Solution Time for an Increase in the Number of Inequality Constraints.

Algorithm	Number of Constraints		
	10 time	15 time/% increase	20 time/% increase
1	6.306	8.938/41.7	11.304/79.3
9	4.153	4.750/14.4	6.602/59.0
10	2.171	3.475/60.1	3.737/72.1
11	3.491	6.716/92.4	8.404/140.7
15	9.081	13.523/48.9	17.258/90.0
21	6.639	12.270/84.8	15.616/135.2
31	9.050	11.417/26.2	17.773/96.4

least sensitive algorithm to an increase in the number of inequality constraints. This again points out the fact that the solution time required for this algorithm is dependent upon the number of variables to a greater extent than the number of constraints even though the solution time required for a linear programming problem is known to increase as a function of the number of constraints. This result agrees with the results from the timing study where it was demonstrated that the solution of the linear programming problems was not as time consuming as the generation of the linear programming problems.

5.5 Addition of Equality Constraints

The equality constraints were introduced to the problems by removing one inequality constraint for each equality constraint added so that the total number of constraints remained constant. Also for each equality constraint added the maximum number of inequality constraints allowed to be active at the solution was reduced by one. The solution times and the percentage increase in solution time for the addition of one and three equality constraints is presented in Table 5.3. The effect on the solution times were not that evident with the exception of algorithms 9 and 31. Algorithm 9, the repetitive linear programming algorithm was able to handle the addition of a single equality constraint but failed to solve the problems involving three equality constraints due to difficulty with

Table 5.3 Solution Time and Percentage Increase in Solution Time for an Increase in the Number of Equality Constraints.

Algorithm	Number of Equality Constraints		
	0	1	3
	time	time/% increase	time/% increase
1	6.306	6.892/9.3	7.278/15.4
9	4.153	5.382/29.6	*
10	2.171	2.976/37.1	2.749/26.6
11	3.491	6.234/78.6	3.405/-2.5
15	9.081	10.108/11.3	9.956/9.6
21	6.639	8.504/28.1	10.320/55.4
31	9.050	18.380/103.1	27.156/200.2

*Could not locate a feasible starting point.

Newton's method in finding a feasible starting point. Algorithm 31, an interior penalty function algorithm, demonstrated a significant increase in solution time for the addition of equality constraints with a 100% increase for the addition of one equality constraint and a 200% increase for three equality constraints.

The effect on the exterior penalty function algorithms was not as great as might be expected, but the difficulty noted on the test problems from the comparative study was not an increase in solution time, but a failure to satisfy the equality constraints to the required level. This difficulty was not noticed on the problems generated for this test.

Once again the reduced gradient algorithms produced the best solution times although the time savings for the addition of a single equality constraint were not that great for algorithm 11. The interesting point is that the increase to three equality constraints produced a decrease in solution time from the problems involving one equality constraint which demonstrates another advantage of the reduced gradient algorithms over the penalty function algorithms. The increase in solution time for the reduced gradient algorithms on the equality constrained problems was due to the time spent in finding a feasible starting point for there is essentially no difference in the way an equality constraint is handled by these algorithms than a tight inequality constraint.

5.6 Increase in Nonlinearity

The degree of nonlinearity is hard term to quantify. Comparing a problem consisting of polynomial terms to another involving trigonometric or exponential terms as to the degree of nonlinearity is not an easy task. It is possible, however, to gain an understanding of the effects of increasing nonlinearity by concentrating on a single problem form. The standard problem set involves problems constructed with the most nonlinear term being quadratic. By adding cubic and quartic terms to these problems, the degree of nonlinearity is increased, and it was in this manner that the problems were constructed.

The effects on the solution time for the addition of the cubic terms were not dramatic for any of the algorithms, as can be seen from Table 5.4 where the solution times and percentage increase over the standard problem solution time are presented. The only algorithm affected to any major extent by the addition of cubic terms was algorithm 11 for which the solution time increased 64% over the standard problem. It should be noted, however, that the total solution time for algorithm 11, a reduced gradient algorithm, was still well below the times recorded by the penalty function algorithms. The addition of quartic terms again affected algorithm 11 but also the time for algorithm 9, the repetitive linear programming algorithm increased dramatically. This increase in time would be expected

Table 5.4 Solution Time and Percentage Increase in Solution Time for an Increase in Problem Nonlinearity.

Algorithm	Highest Nonlinear Term		
	QUADRATIC time	CUBIC time/% increase	QUARTIC time/% increase
1	6.306	8.263/31.0	10.680/69.4
9	4.153	5.324/28.2	9.386/126.0
10	2.171	2.306/6.2	2.890/33.1
11	3.491	5.735/64.3	7.247/107.6
15	9.081	9.787/28.1	11.023/66.7
21	6.639	8.507/7.8	11.069/21.4
31	9.050	10.393/14.8	12.453/37.6

for a method based on linear programming solutions, but the effect on algorithm 11 is more difficult to explain. It is true that the reduced gradient algorithms are linear approximation methods, but the increase in nonlinearity did not effect algorithm 10, another reduced gradient algorithm to any significant extent. A possible explanation for the difference is the type of unconstrained search directions generated by the algorithms. All of the penalty function algorithms and also algorithm 10 employed a variable metric technique to generate the search directions and were not significantly effected by the increase in nonlinearity. Algorithm 11, on the other hand, employed the conjugate gradient technique of Fletcher-Reeves and this technique, while producing a smaller percentage increase in solution time for an increase in the number of design variables, did show a significant increase in solution time as the nonlinearity of the problem increased.

5.7 Discussion

The major point to note from the results presented in this chapter is that they are consistent with the previously presented results. The linear approximation methods and in particular the generalized reduced gradient algorithms are significantly faster than the penalty function algorithms. The time required by the reduced gradient algorithms was consistently one half to one-third the time required by the penalty function algorithms.

As far as the problem factors are concerned, all of the algorithms demonstrated the largest increase in solution time as the number of variables was increased. Increasing the number of inequality constraints, the addition of equality constraints or increasing the level of nonlinearity generally had a minor effect on the solution time required by the algorithms as compared to an increase in the number of design variables.

CHAPTER 6 COMBINATION OF ALGORITHMS

6.1 Introduction

The ability to handle constraints directly has been demonstrated to be more efficient than the penalty function approach in all of the results presented in the previous chapters. This brings up the question of whether the penalty function algorithms can be modified to produce the same computational time savings. The penalty function algorithms had difficulty in obtaining constraint satisfaction on problems where several constraints were active at the solution. To solve this type of problem the input parameters had to be selected carefully so that each stage was solved accurately and so the penalty function contours at successive stages were not altered so drastically that progress to the optimal solution was impossible at the final stages. This is a very time consuming process, but if the problem is considered from another approach several improvements can be made. Consider a method consisting of two distinct phases. The first phase is to locate the vicinity of the solution and to identify the active constraint set. The second phase is to satisfy any violated constraints and to locate the optimum to the desired accuracy. The exterior penalty function algorithms are a natural choice for the first phase. The convergence criteria for each unconstrained stage may be very loose and the increase in the penalty parameter could also be fairly large. This would result in the location of the vicinity of the minimum in a few stages each of which requiring a small amount of time due to the large value of the convergence criteria at each stage. The active constraint set would be easy to obtain since after several "loose" penalty stages the active constraints will generally be slightly to moderately violated for an exterior penalty algorithm. The second phase would then involve locating the feasible optimal point. The logic of this phase could be very similar to that of a reduced gradient algorithm only now the selection of the decision and state variables may be made with a good idea of the active constraint set. Another approach would be to initiate a repetitive linear programming algorithm at this point since this type of algorithm has demonstrated extremely good performance once the constrained region has been located. Both of these approaches will be considered by combinations of existing algorithms. The biased penalty function algorithm, method 1, will first be coupled with a reduced gradient algorithm, method 11, and then with a repetitive linear programming algorithm, method 9. The improvement over the normal penalty function approach as well as the relative ranking with the algorithms tested in the comparative study will be considered.

6.2 Combination of Biased Penalty and Reduced Gradient Algorithms

The biased penalty function method was selected for phase one for several reasons. First of all the method is designed to minimize the distortions in the penalty function contours which allows for several penalty stages to be solved very rapidly. Secondly the method was extremely adept at locating the vicinity of the solution after the first few stages, thus the transfer to the phase two method can be made quickly. Also only the initial penalty multiplication parameter had to be selected which reduced the required number of input parameters.

Algorithm 11 was selected for the second phase solution since the input format was very similar to the phase one method, algorithm 1. This reduced the effort in the implementation of a program to interface the two algorithms. The interface program initially sets up the required starting information for algorithm 1. The only changes made from a normal run for the algorithm were that the line search criteria was set to a value of 10^{-2}, the overall convergence criteria was set to a value of 10^{-3}, and the initial penalty parameter was set to a value of 10. The problem was then solved by algorithm 1 and the resulting solution was modified to be input to algorithm 11. This modification basically involved the addition of artificial variables to satisfy the violated constraints and the reordering of the design variables to insure that the artificial variables were contained in the initial decision variable set. The artificial variable approach was used since the search for a feasible point would generally remain in the vicinity located by the phase one algorithm. Using this procedure the artificial variables are set to the exact constraint violations and added to the constraints to artificially satisfy the violated constraints. An additional term is then appended to the objective function which essentially penalizes the objective function for nonzero values of these artificial variables. The effect of this procedure is to rapidly reduce the artificial variables in the initial solution stages to generate a feasible point. The normal approach for location of a feasible starting point for algorithm 11 was to simply minimize the sum of the violated constraints. This type of procedure was not appropriate for the combined solution procedure because in satisfying the constraints in this manner no attempt is made to stay in the vicinity which the phase one algorithm located, thus essentially wasting much of the information supplied by phase one.

The combination of these two algorithms was applied to the test problem set from the comparative study and the results were quite good. First of all the combined method solved all twenty-three of the rated test problem set, which neither algorithm 1 or 11 was able to do alone. This was because the combination of the two methods complimented one another nicely. Algorithm 1 was generally able to locate the vicinity of the solution but had trouble satisfying the constraints and especially the equality constraints. Algorithm 11 occasionally had difficulty in moving through the design space as on problem 12 and had a tendency to terminate at local minima near the starting point for many of the additional problems in the test set. The method did perform well by maintaining constraint satisfaction and in locating the optimum to a very accurate degree. Thus some of the weaknesses one method had alone were compensated for by the combination with the other method.

The recorded solution times were very fast for the combined methods. The percentage decrease in solution time for the combined algorithms over the normal solution times for algorithms 1 and 11 on the rated test problem set is presented in Table 6.1.

The percentage reduction for the combined methods over the solution times reported for algorithm 1 is quite large for all of the problems in the study with the exception of problems 5 and 12 which had unconstrained solutions. For problems with unconstrained solutions no improvement would be expected, but a slight time savings resulted on these problems due to the loose line search in algorithm 1 at the initial stages. Time savings on these problems are not that important to the overall performance of the combined algorithms since these problems required only a small amount of computational time for either of the algorithms alone. The interesting point is that as the problems become more difficult

Table 6.1 Percentage Improvement in Solution Time for the Biased Penalty-Reduced Gradient Combined Algorithm over Algorithms 1 and 11.

Problem	Percentage over Algorithm 1	Improvement over Algorithm 11
1	69.6	-32.7
2	81.3	-180
3	92.6	-80
4	43.8	-42.1
5	14.7	1.5
6	†	†
7	44.4	-25.0
8	80.3	-87.7
10	69.2	-90.9
11	75.4	-15.7
12	2.6	†
14	75.4	-7.4
15	92.1	-20.0
16	†	4.9
17	71.9	†
18	90.9	35.0
19	77.0	22.9
20	97.3	33.5
23	61.1	61.1
24	74.9	6.1
25	79.9	68.2
26	†	-.3
27	74.1	5.3

† No solution was found with algorithm 1 alone.

(problems 14-27) the percentage improvement is consistently well above 60%. In fact the average percentage time savings over the entire test problem set was approximately 68% which is slightly better than a two-thirds reduction in computational time from the amount of time recorded for algorithm 1 in the comparative study. Now the average solution time for the reduced gradient algorithms has been demonstrated to be one-half to one-third of the time required for a penalty function algorithm so an average reduction in the solution time to one-third of the time required by algorithm 1 in the comparative study should place the combined method on the same level of computational speed as the reduced gradient algorithms. This fact is not that obvious from Table 6.1. The combined method produced slower solution times on most of the easier problems but on the more difficult problems the solution times were generally better than for algorithm 11. The problems where the combined method was significantly slower were generally problems in which only one or two constraints were active and algorithm 11 had extremely fast solution times. Even for these problems, however, the solution time for the combined methods was well below the average solution times for all of the methods.

The overall performance of the combined algorithms is best described by relative rankings based on the problems solved within a given percentage of the average time. The combined method solved 78.3% of the problems within 25% of the average solution time, 87% within 50% of the average solution time and 100% within 75% of the average solution time. This would rank the combination of the algorithms at the top in every ranking, even above the reduced gradient algorithms.

Performance on the additiaional test problem set was also excellent. Progress on problem 9 was slightly better than for either algorithm 1 or 11 alone, and for problem 13 the same solution was located as by all of the better gradient based algorithms. It was on problems 21, 22 and 30 where the combined method really demonstrated its advantage over either algorithm applied separately. Algorithm 1 alone was not able to satisfy all of the inequality constraints on problems 21 and 22 but was able to locate the vicinity of the optimal solution. On problem 30, algorithm 1 was simply unable to satisfy the eleven equality constraints but with a loose line search criteria and a relatively low penalty factor the algorithm was able to locate a point in the vicinity of the reported solution. Algorithm 11, on the other hand was able to satisfy the constraints for all of these problems but it terminated after making very little progress on each one. The combined methods located feasible points very close to the reported solution for problems 21 and 22 and was able to find the reported solution to problem 30. An excellent solution was also recorded for problems 28 and 29 although for problem 29 algorithm 1 made very little progress and the solution was mainly due to algorithm 11.

The overall results point out that a combination of the biased penalty function algorithm and a generalized reduced gradient algorithm can produce a computationally fast and robust method. The method is as fast or slightly faster on the average as the reduced gradient algorithm was alone and added a measure of robustness to both of the algorithms applied separately. It should be noted, however, that the strength of the combined method is due to the application of the generalized reduced gradient algorithm to handle the constraints directly in phase two, for it was this phase of the algorithm which allowed the implementation of the phase one algorithm in its present form. Actually the location of the optimal region by the phase one algorithm acts as a computational extension of the reduced gradient algorithm, allowing initial explorations to occur in the infeasible region.

6.3 Combination of Biased Penalty and Repetitive Linear Programming Algorithms

The results from the combination of the biased penalty-reduced gradient algorithms provides the motivation to investigate other combinations which have the potential of producing a similar algorithm. The implementation of the phase one algorithm could be identical to the previous combined method if another algorithm which handles the constraints directly could be applied as a phase two algorithm. What would be desirable would be to implement an algorithm which is as readily available as is a penalty function algorithm and which is less computationally complex than the reduced gradient algorithms. Such a method is the repetitive linear programming method. From the comparative study it was determined that the basic cause of failure for this type of algorithm was the inability to move through an unconstrained region to locate the active constraints. The solution times on heavily constrained problems were generally quite good for the repetitive linear programming methods. It would seem then, that this type of method would be an excellent choice for a phase two algorithm. The application of the biased penalty function method, algorithm 1, as a phase one method will be used again to locate the vicinity of the optimal solution, and the repetitive linear programming method, algorithm 9, will be started from this point.

Interfacing these two algorithms was even an easier task than for the biased penalty-reduced gradient combination, because the repetitive linear programming algorithm did not require a feasible starting point with respect to the inequality constraint set. Artificial variables were used, however, to provide initial feasibility for the equality constraint set. The artificial variables were introduced to eliminate the possibility of divergence of Newton's method in initially satisfying the equality constraints.

The recorded solution times were again very good for the combined algorithm. The percentage decrease in solution time for the combined algorithms over the normal solution times for algorithms 1 and 9 on the rated test problem set is presented in Table 6.2. As for the combination of algorithms 1 and 11, this combination solved all of the rated test problem set, and again for the large majority of the test problems a significant reduction in the solution time required by algorithm 1 was recorded. No improvement was recorded over algorithm 1 for problems 4, 5, 10 and 12 since these problems only contained variable bounds and no functional constraints. The normal solution of algorithm 1 was used for these problems since the method of repetitive linear programming does not handle this type of problem well. Again, however, the normal solution time for algorithm 1 was quite small for these problems. The only other problem where an improvement in the solution time was not produced over algorithm 1 was problem 7 where the solution time actually increased over 40%. Even with this increase the solution time on problem 7 was less than 50% of the average time for all of the tested algorithms on that problem. On all of the other problems, and especially for the more difficult problems the percentage reduction over algorithm 1 was quite similar to the biased penalty-reduced gradient algorithm combination. The average percentage time savings over algorithm 1 for this algorithm on the rated set of test problems was found to be on the order of 50%. This is again a significant time reduction and places the computational speed of the algorithm close to the level of the reduced gradient algorithms. The percentage improvement over algorithm 9 is much the same as for the biased penalty-reduced gradient combination over algorithm 11 as can be seen

Table 6.2 Percentage Improvement in Solution Time for the Biased Penalty-Repetitive Linear Programming Combined Algorithm over Algorithms 1 and 9.

Problem	Percentage over Algorithm 1	Improvement over Algorithm 9
1	64.4	-26.8
2	79.6	-53.3
3	91.8	-250.0
4	0.0	†
5	0.0	†
6	†	†
7	-41.1	-1.6
8	74.1	8.0
10	0.0	64.2
11	81.9	†
12	0.0	93.9
14	59.3	2.2
15	85.6	-2.9
16	†	84.4
17	36.4	-83.9
18	95.8	30.7
19	77.0	43.8
20	92.9	-18.1
23	70.2	†
24	74.8	-17.6
25	79.7	†
26	†	-120
27	56.6	32.6

† Algorithm applied alone did not reach the solution.

in Table 6.2. The solution time on the easier problems increased by rather large amounts but the solution times on these problems were still well below the average solution times. Again the best measure of performance is the relative rankings. The fact is that the relative rankings for this combination are on par with the reduced gradient algorithms with 65.2% of the problems solved within 25% of the average time, 82.6% of the problems solved within 50% of the average, 91.3% within 75% of the average time, and 100% of the problems were solved within 100% of the average time.

The combination of the biased penalty-repetitive linear approximation was generally slightly slower than the biased penalty-reduced gradient combination. This can be seen in Table 6.3 where the percentage increase in solution time for the biased penalty-repetitive linear programming combined method over the biased penalty-reduced gradient combined algorithm is presented. On the average a twenty percent increase in solution time was noted. So while the combination of algorithms 1 and 9 ranked slightly behind the combination of algorithms 1 and 11 the total rankings are at the level of the generalized reduced gradient algorithms and significantly above the rankings of either algorithm 1 or algorithm 9 in the comparative study.

The performance on the additional test problems was also fairly good. The solutions found for problems 9 and 13 were basically the same as for the combination method of algorithms 1 and 11. The optimal solution was located for problems 21

Table 6.3 Percentage Increase in Solution Time for the Biased Penalty-Repetitive Linear Programming Combined Algorithm over the Biased Penalty-Reduced Gradient Combined Algorithm.

Problem	Percentage Increase
1	14.6
2	8.7
3	10.7
4	43.7
5	20.0
6	43.7
7	60.6
8	23.9
10	69.1
11	-36.1
12	2.5
14	39.6
15	44.8
16	68.8
17	55.9
18	-118.3
19	0.0
20	61.6
23	-30.5
24	.6
25	1.0
26	55.3
27	40.4

and 22 and excellent progress was reported on problem 29. The highly nonlinear maximum stress constraint in problem 28, and the presence of the eleven equality constraints in problem 30 were not handled well by the repetitive linear programming algorithm and no feasible point could be found on either of these problems.

6.4 Discussion

The combination of the biased penalty function algorithm with either the reduced gradient algorithm or the repetitive linear programming algorithm demonstrated an improved level of performance over any of the algorithms applied singly. This was basically due to the fact that the combined algorithms complimented each other, with each algorithm performing a specific function in a computationally efficient manner. The penalty function algorithm, with a loose line search criteria and a loose overall convergence criteria was able to locate the vicinity of the solution quickly. The generalized reduced gradient algorithm or the repetitive linear programming algorithm was then able to satisfy the violated constraints and to locate the optimal solution. The generalized reduced gradient algorithm produced a slightly faster combination and was able to handle even highly nonlinear problems and problems which contained many nonlinear equality constraints. The repetitive linear programming algorithm performed quite well also but was unable to follow extremely nonlinear constraints or to make much progress when many nonlinear equality constraints were present. The basic simplicity and the general good performance of the penalty function-repetitive linear programming algorithm are the basic advantages of this combination, while the unmatched overall performance of the penalty function-generalized reduced gradient combination is the advantage of this combination.

CHAPTER 7 CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions

The first and most noteworthy conclusion which can be drawn from the comparative study is that the linear approximation methods and in particular the generalized reduced gradient algorithms were more effective than any of the penalty function algorithms. The generalized reduced gradient algorithms solved a greater percentage of the test problems and were significantly faster than any other type of method. This superiority was demonstrated in all of the relative rankings based on the number of problems solved within a given percentage of the average solution time where the generalized reduced gradient algorithms consistently ranked at the top. In the study of how the algorithms were affected by the type of problem being solved the generalized reduced gradient algorithms were consistently two to three times faster than the penalty function algorithms. The generalized reduced gradient algorithms also demonstrated a superior level of control from a users standpoint. This increase in control was due to the ability to specify the level of tolerance allowed in the active constraint set and for the variables at a bound. The penalty function algorithms had no such feature and there was no way of guaranteeing any specific level of constraint satisfaction. Problem dependent user parameters were not found to be as important to the generalized reduced gradient algorithms as to the penalty function algorithms. There was no need to specify any penalty or penalty reduction factors so although more control over problem solution was available very few input parameters had to be selected by the user. The standard values of the input parameters as suggested in the users manual for the generalized reduced gradient algorithms worked very well for the test problems in the comparative study and rarely was any parameter adjustment required. The generalized reduced gradient algorithms were also able to obtain a very high level of accuracy in the final solution. This is demonstrated by the high relative rankings even when the total allowed error was very small. Again no other type of algorithm was able to reproduce this feature. The generalized reduced gradient algorithms were also found to be effective on a wide variety of problems including problems containing equality constraints which were handled in a particularly effective fashion. So the generalized reduced gradient algorithms have clearly demonstrated a superior level of performance over the other classes of methods being both faster and able to solve more of the test problem set than the penalty function algorithms. The only problems encountered with the generalized reduced gradient algorithms were the tendency to follow constraints to local minima on several of the additional test problems and the inability of the presently available generalized reduced gradient algorithms to handle problems where derivative information may not be calculated accurately. The problem of the generalized reduced gradient algorithms following constraints to local minima is not a serious drawback since essentially all algorithms available today search for local minima, but following the constraints on several of the highly constrained problems resulted in very little progress being made by the generalized reduced gradient algorithms. If the generalized reduced gradient algorithms stop after making little progress on a problem the user is encouraged to try an alternate starting point.

The repetitive linear programming methods, especially algorithm 9 (RALP), produced excellent solution times on a majority of the problems but their behavior became very erratic when required to move through any unconstrained region, or on extremely nonlinear problems. This type of behavior would have to be expected of an algorithm based on a linear programming technique.

The penalty function algorithms were found to be somewhat slower than the linear approximation algorithms. Out of all of the penalty function algorithms tested, however, those applying a variable metric technique for the generation of search directions for the successive penalty stages proved to be most effective, but even these methods required two to three times more computational time than the generalized reduced gradient algorithms. The basic trouble encountered with the penalty function algorithms was in obtaining sufficient constraint satisfaction. The exterior penalty function algorithms generally approached the solution from the infeasible region and frequently were unable to satisfy all of the constraints to any degree of accuracy. The interior penalty function algorithms had the same basic difficulty with the exception that the solution was approached from the feasible region and the problem was not with inequality constraint violation but in not being able to obtain sufficient constraint tightness to zero in on the optimal value of the objective function. A similar difficulty was noted for both the interior and exterior penalty functionson equality constrained problems, along with a tendency to get hung up on an equality constraint and not being able to move. To obtain any significant level of accuracy in the objective function and constraints at the solution tight convergence was required at each successive stage and the penalty parameter had to be increased or decreased in relatively small levels to prevent the contours from becoming overly distorted. This resulted in the penalty function algorithms requiring an excessive amount of solution time. The biased penalty function algorithm had slightly less difficulty in obtaining constraint satisfaction which was most likely due to the fact that the distortions of the successive penalty contours are less than for the other penalty function algorithms, but even the biased penalty method did require some parameter adjustment to achieve constraint satisfaction. This type of parameter adjustment was necessary for almost every exterior penalty function algorithm. The one advantage of the penalty function algorithms was that methods are available which do not require derivatives. This type of method, as was previously mentioned, is not currently available in any method employing a generalized reduced gradient approach. These nongradient penalty function algorithms are the only methods available to solve the problems where gradient information does not exist.

All of the algorithms tested were found to be extremely sensitive to the number of design variables in a problem. The solution times increased dramatically for any increase in the total number of design variables. As the number of variables is increased to the level of 75 to 100 a solution time of well over 1000 seconds of time could be expected (on a CDC 6500 machine). This estimation is based on the extrapolated time from the solution time required on a set of quadratic problems which would have to be considered as a lower bound on the solution time expected in general usage. The largest number of design variables contained in any problem considered in the comparative study was forty-eight and only six of the algorithms tested were able to produce a solution. The recorded solution times on this problem ranged from approximately sixty seconds to well over two hundred seconds, with three best solution times on the problem achieved by generalized reduced gradient algorithms. Another limitation regarding the number of design variables which can be handled by the majority of codes currently available is due to the program dimensioning. Several of the algorithm are limited to a maximum of fifty design variables and others to one hundred. The two exceptions are the generalized reduced gradient algorithm OPT and the biased penalty function algorithm BIAS, both of which are variably dimensioned and the only physical limitation on the size of problem which can be handled is the amount of storage space available on the computer system being used. An increase in the number or type of constraints or an increase in the nonlinearity did not have as large an effect on the solution times as did an increase in design variables. The percentage change for variations in the problems was remarkably similar for all of the various types of algorithms. This means that the generalized reduced gradient algorithms would be expected to hold the 100% to 200% time savings over the penalty function algorithms on all types of problems. This result simply points out the desirability of the generalized reduced gradient algorithms on large scale problems or for problems where the objective function or constraints require a large amount of computational time to evaluate.

A combination of a penalty function algorithm with a method to handle the constraints directly was found to be very effective. The biased penalty function algorithm with a loose line search criteria and a loose overall convergence criteria was found to be very effective in locating the vicinity of the optimum quickly. A generalized reduced gradient algorithm, starting from the point located by the penalty function algorithm, was then very effective at satisfying the constraints and converging to the optimal solution. This combination proved to be very fast and reduced the tendency of the generalized reduced gradient algorithm to follow constraints to a local minimum. The total performance of this combination proved to be superior to the performance of any single algorithm in the relative rankings, including the generalized reduced gradient algorithms. This superior level of performance was due more to an increase in the number of test problems solved than to a decrease in computational speed although on several of the more difficult problems a significant reduction in time over the generalized reduced gradient algorithm was recorded. The application of a repetitive linear programming in place of the reduced gradient algorithm for the combined method also proved to be effective. The performance of this combination on the test problem set was as good as the generalized reduced gradient algorithms in the relative rankings for the comparative study. This combination was not quite as fast as the penalty function-generalized reduced gradient combination, requiring an average of approximately twenty percent more time than the penalty function-generalized reduced gradient combination on the test problems. The combination still had difficulty with highly nonlinear problems but the overall simplicity, the availability of the methods, and the relatively good performance are the strengths of this combination. Thus the performance of even the best methods can seem to be en-ehanced by the combination of algorithms.

In summary then there are methods currently available which will handle a broad range of nonlinear programming problems in an efficient manner. Difficulties with large scale problems and the possibility of locating local minima exist but the overall usefulness of the available algorithms is sufficient to encourage their general use.

7.2 Recommendations

Although no single algorithm can be expected to solve every problem encountered, it is the recommendation of this study that the generalized reduced gradient algorithms should be applied to solve the problem before any other method is tried. The ability of this class of algorithms to solve problems in an efficient manner is not matched by any other class of algorithms. The high degree of user control inherent in the generalized reduced gradient algorithms also lends itself to general problem solution. The author would also recommend further development of the generalized reduced gradient algorithms. Most of the reduced gradient algorithms have been developed recently and little work has been done to determine the best method for generating search directions, the manner of basic variable selection and subsequent basis changing, and the numerical technique of solving the set of nonlinear equations. Another area of possible research is the application of the same type of logic contained in the generalized reduced gradient algorithms to produce an algorithm which is not dependent upon gradient information. Perhaps this research would result in a generalized reduced gradient optimization package. This type of package could then be applied to problems with or without gradient information available.

Another recommendation would be that any future comparison of nonlinear programming algorithms should include at least one of the generalized reduced gradient algorithms tested in this study to provide a reference to the relative performance of the additional algorithms. Finally further investigation of the combination of penalty function algorithms and the linearization type methods is recommended since this type of combination has demonstrated a level of performance which is superior to any single algorithm.

REFERENCES

REFERENCES

1. Mylander, W. C., Holmes, R. L., and McCormick, G. P., A Guide to SUMT - Version 4, Research Analysis Corporation, RAC-P-63, 1974.

2. Carroll, C. W., "The Created Response Surface Technique for Optimizing Nonlinear Restrained Systems," Operations Research, Vol. 9, 1961.

3. Fiacco, A. V., and McCormick, G. P., Nonlinear Programming: Sequential Unconstrained Minimization Techniques, Wiley, New York, 1968.

4. Davidon, W. C., "Variable Metric Method for Minimization," Argonne National Laboratory, Report No. ANL-5990, 1959.

5. Fletcher, R. and Reeves, C. M., "Function Minimization by Conjugate Gradients," Computer Journal, Vol. 7, No. 2, 1964.

6. Powell, M. J. D., "An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives," Computer Journal, Vol. 7, No. 4, 1964.

7. Hooke, R. and Jeeves, T. A., "Direct Search Solution of Numerical and Statistical Problems," JACM, Vol. 8, No. 2, 1961, 212-229.

8. Broyden, C. G., "The Convergence of a Class of Double-Rank Minimization Algorithms; 1. General Considerations," J. Inst. Math. Appl., 6:76, 1970.

9. Schuldt, S. B., "A Method of Multipliers for Mathematical Programming Problems with Equality and Inequality Constraints," Journal of Optimization Theory and Applications, Vol. 17, No. 1/2, 1975, 155-162.

10. Himmelblau, D. M., Applied Nonlinear Programming, McGraw-Hill, New York, 1972, 368-369.

11. Gottfried, B. S., and Weisman, J., Introduction to Optimization Theory, Prentice-Hall, New Jersey, 1973.

12. Fox, R. L., Optimization Methods for Engineering Design, Addison-Wesley, Mass., 1971.

13. Griffith, F. E., and Stewart, R. A., "A Nonlinear Programming Technique for Optimization of Continuous Processing Systems," Management Science, Vol. 7, 1961, 379-392.

14. Dantzig, G. B., Linear Programming and Extensions, Princeton University Press, 1963.

15. Wolfe, P., "Methods for Linear Constraints," in Nonlinear Programming, ed. by Adadie, North-Holland Publishing Co., 120-124, 1967.

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

17. Abadie, J., and Guigou, J., "Numerical Experiments with the GRG Method," in Integer and Nonlinear Programming, ed. by Adabie, North-Holland Publishing Co., 1970, 529, 536.

18. Gabriele, G. A., and Ragsdell, K. M., "The Generalized Reduced Gradient Method: A Reliable Tool for Optimal Design", Journal of Engineering for Industry, Trans. ASME, Series B, Vol. 99, No. 2, May 1977, 394-400.

19. Zoutendikj, G., Methods of Feasible Directions, Elsevier Publishing Co., Amsterdam, 1960.

20. Box, M. J., "A New Method of Constrained Optimization and a Comparison with Other Methods," Computer Journal, Vol. 8, No. 1, 1965, 42-51.

21. Brooks, S. H., "A Comparison of Maximum Seeking Methods," J. Ope. Res., Vol. 7, No. 4, 1959, 430-437.

22. Cauchy, A., "Method generale pour la resolution of systems d' equations simultanees," Comptes Rendus de L' Academic des Science, 25, 1847, 536-538.

23. Fletcher, R., "Function Minimization Without Evaluating Derivatives," Computer Journal, Vol. 8, No. 1, April 1965, 33-41.

24. Leon, A., "A Comparison Among Eight Known Optimizing Procedures," in Recent Advances in Optimization Techniques, ed. by Lavi and Vogl, Wiley, New York, 1966, 23-42.

25. Box, M. J., "A Comparison of Several Current Optimization Methods and the Use of Transformations in Constrained Problems," Computer Journal, Vol. 9, May 1966, 67-77.

26. Kowalik, J., and Osborne, M. R., Methods for Unconstrained Optimization Problems, American Elsevier, New York, 1968.

27. Colville, A. R., "A Comparative Study of Nonlinear Programming Codes," Technical Report No. 320-2949, IBM New York Scientific Center, June 1968.

28. Stocker, D. C., "A Comparative Study of Nonlinear Programming Codes," M.S. Thesis, The University of Texas, Austin, Texas, 1969.

29. Pearson, J. D., "Variable Metric Methods of Minimization," Computer Journal, Vol. 12, 1969, 171-178.

30. Huang, H. Y., and Levy, A. V., "Numerical Experiments on Quadratically Convergent Algorithms for Functional Minimization," J. Optim. Theory Applns., Vol. 6, 1970, 269-282.

31. Murtagh, B. A., and Sargent, R. W. H., "Computational Experience with Quadratically Convergent Minimization Methods," Computer Journal, Vol. 13, 1970, 185-194.

32. Bard, Y., "Comparison of Gradient Methods for the Solution of Nonlinear Parametric Estimation Problems," SIAM J. Numer. Anal., Vol. 7, 1970, 159-186.

33. Jones, A., "Spiral - A New Algorithm for Nonlinear Parameter Estimation Using Least Squares," Computer Journal, Vol. 13, 1970, 301-308.

34. Pierson, B. L., and Rajtora, S. G., "Computational Experience with the Davidon Method Applied to Optimal Control Problems," TEEE Transactions SSC4, 1970, 240-242.

35. DeSilva, B. M. E., and Grant, G. N. C., "Comparison of Some Penalty Function Based Optimization Procedures for the Synthesis of a Planar Truss," International Journal for Numerical Methods in Engineering, Vol. 7, No. 2, 1973, 155-173.

36. Sargent, R. W. H., and Sebastian, D. J., "Numerical Experience with Algorithms for Unconstrained Minimization," in Numerical Methods for Non-Linear Optimization, ed. by Lootsma, Academic Press, London, 1972.

37. Himmelblau, D. M., "A Uniform Evaluation of Unconstrained Optimization Techniques," in Numerical Methods for Non-linear Optimization, ed. by Lootsma, Academic Press, London, 1972.

38. Schrack, G., and Borowski, N., "An Experimental Comparison of Three Random Searches," in Numerical Methods for Non-linear Optimization, ed. by Lootsma, Academic Press, London, 1972.

39. Biggs, M. C., "Constrained Minimization Using Recursive Equality Quadratic Programming," in Numerical Methods for Non-linear Optimization, ed. by Lootsma, Academic Press, London, 1972.

40. Eason, E. D., and Fenton, R. G., "Testing and Evaluation of Numerical Methods for Design Optimization," UTME-TP 7204, University of Toronto, Sept. 1972.

41. Larichev, O. I., and Gorvits, G. G., "New Approach to Comparison of Search Methods Used in Nonlinear Programming Problems," Journal of Optimization Theory and Applications, Vol. 13, No. 6, June 1974, 635-659.

42. Pappas, M., and Moradi, J. Y., "An Improved Direct Search Mathematical Programming Algorithm," Journal of Engineering for Industry, Trans. ASME, Series B, Vol. 97, No. 4, Nove. 1975, 1305-1310.

43. Schuldt, S. B., Ragsdell, K. M., Gabriele, G. A., Root, R. R., and Sandgren, E., "Application of a New Penalty Function Method to Design Optimization," Journal of Engineering for Industry, Trans. ASME, Series B, Vol. 99, No. 1, Feb. 1977, 31-36.

44. Wolfe, P., "Convergence Theory in Nonlinear Programming," in Integer and Nonlinear Programming, ed. by Abadie, North-Holland Publishing Co., Amsterdam, 1970, 1-36.

45. Luenberger, D. G., Introduction to Linear and Nonlinear Programming, Addison Wesley, Mass., 1973.

46. Dembo, R. S., "A Set of Geometric Programming Test Problems and their Solutions," Working Paper #87, Department of Management Sciences, University of Waterloo, Waterloo, Ontario, June 1974.

47. Gill, P. E., and Murray, W., Numerical Methods for Constrained Optimization, Academic Press, 1974.

48. Eason, E. D., "Validity of Colville's Time Standardization for Comparing Optimization Codes," ASME Paper 77 DET 116, 1977.

49. Phillips, D. T., Ravindran, A., and Solberg, J. J., Operations Research, Wiley, 1976.

50. Rosen, J. B., and Suzuki, S., "Construction of Nonlinear Programming Test Problems," ACM Communications, Volume 8, No. 2, Feb., 1965.

51. Siddall, J. N., 'OPTISEP' Designers Optimization Subroutines, Faculty of Engineering, McMaster University, Hamilton, Ontario, Canada, ME/71/DSN/REP1.

52. Miele, A., and Cantrell, J. W., "Study on a Memory Gradient Method for the Minimization of Functions," Journal of Optimization Theory and Application, Vol. 3, No. 6, 1969.

53. LaFrance, L. J., "User's Manual for GRGDFP, an Optimization Program," Herrick Laboratories Report No. 45, April 1974.

54. Schuldt, S., Honeywell Corporation, Bloomington, Minnesota, Personal Communication.

55. Lasdon, L. S., Waren, A. D., Ratner, M. W., and Jain, A., "GRG User's Guide", Cleveland State University Technical Memorandum CIS-75-02, Nov. 1975.

56. Lasdon, L. S., Waren, A. D., Ratner, M. W., and Jain, A., "GRG System Documentation," Cleveland State University Technical Memorandum CIS-75-01, Nov. 1975.

57. Gabriele, G. A., and Ragsdell, K. M., "OPT, A Nonlinear Programming Code in Fortran IV," The Modern Design Series, Vol. 1, Purdue Research Foundation, 1976.

58. Guigou, J., "Presentation Et Utilisation Du Code GREG," Depårtment Traitement De L'information Et Estudes Mathematiques, Electricite De France, Paris, France, 1971.

59. Gulf Oil Corporation, "COMPUTE II, A General-Purpose Optimizer for Continuous, Nonlinear Models - Description and User's Manual," Gulf Computer Sciences Incorporated, Houston, Texas, 1972.

60. Keefer, D. L., "SIMPAT" A Self-Bounding Direct Search Method for Optimization, I&EC Process Design & Development, Vol. 12, Jan. 1973, 92.

61. Nelder, J. A., and Mead, R., "A Simplex Method for Function Minimization," Computer Journal, Vol. 7, 1964, 308.

62. Afimiwala, K. A., "Program Package for Design Optimization," Department of Mechanical Engineering, State University of New York at Buffalo, 1974.

63. The University of Liverpool Computer Laboratory, E04HAF," Numerical Algorithms Group, Document No. 737, 1974.

64. Lootsma, F. A., "A Survey of Methods for Solving Constrained Minimization Problems vai Unconstrained Minimization," in Numerical Methods for Non-linear Optimization, ed. by Lootsma, Academic Press, London, 1972.

65. Staha, R. L., "Documentation for Program Comet, A Constrained Optimization Code," The University of Texas, Austin, April 1973.

66. Ragsdell, K. M., and Phillips, D. T., "Optimization of a Class of Welded Structures Using Geometric Programming," Journal of Engineering for Industry, Trans. ASME, Series B, Vol. 98, No. 3, 1021-1025.

67. Tomás, J., "The Synthesis of Mechanisms as a Nonlinear Programming Problem," Journal of Mechanisms, Vol. 3, 1968, 119-130.

68. Williams, D., Whirlpool Corporation, Benton Harbor, Michigan, Personal Communication.

69. Philipson, R., Dept. of Industrial Engineering, Purdue University, W. Lafayette, Indiana, Personal Communication.

70. Himmelblau, D. M., "Optimal Design via Structural Parameters and Nonlinear Programming," Engineering Optimization, Vol. 2, No. 1, 1976, 17, 27.

APPENDICES

Appendix A

Description of Algorithms

(1). METHOD OF MULTIPLIERS (BIASED PENALTY FUNCTION)

AVAILABILITY: K. M. Ragsdell, School of Mechanical Engineering, Purdue University, W. Lafayette, Indiana.

REFERENCE: Schuldt et all [43]

METHOD: Exterior penalty function of the form

$$P(\bar{x},\sigma^{(m)},\tau^{(m)}) = f(\bar{x}) + R\sum_{k=1}^{K}\{<g_k(\bar{x}) + \sigma_k^{(m)}>^2 - [\sigma_k^{(m)}]^2\}$$

$$+ R\sum_{\ell=1}^{L}\{[h_\ell(\bar{x}) + \tau_\ell^{(m)}]^2 - [\tau_\ell^{(m)}]^2\}$$

where

$$\sigma_k^{(m+1)} = <g_k(\bar{x}^{(m)}) + \sigma_k^{(m)}>; \quad k = 1,2,3, \ldots, K$$

and

$$\tau_\ell^{(m+1)} = h_\ell(\bar{x}^{(m)}) + \tau_\ell^{(m)}; \qquad \ell = 1,2,3, \ldots, L$$

FEATURES: This form of penalty function seeks to minimize distortions of the successive penalty function contours. The method leaves the curvature of the contours unchanged from stage to stage for linear constraints and a second order influence on the curvature is present for nonlinear constraints. The computational algorithm applies the Davidon-Fletcher-Powell technique to generate the unconstrained search directions and handles variable bounds internally. The penalty parameter R does not vary from stage to stage.

(2). SEEK1

AVAILABILITY: J. N. Siddall, School of Mechanical Engineering, McMaster University, Hamilton, Ontario, Canada.

REFERENCE: Siddall [51]

METHOD: Semi interior penalty function (once constraints are satisfied they generally will remain satisfied) of the form

$$P(\bar{x}) = f(\bar{x}) + 10^{20} \sum_{k=1}^{K} |<g_k(\bar{x})>| + 10^{20} \sum_{\ell=1}^{L} |h_\ell(\bar{x})|$$

FEATURES: A Hooke-Jeeves direct search is made followed by a random check. If the random check produces a better point the method is restarted from that point.

(3). SEEK3

AVAILABILITY: J. N. Siddall, School of Mechanical Engineering, McMaster University, Hamilton, Ontario, Canada.

REFERENCE: Siddall [51]

METHOD: Interior penalty function of the form

$$P(\bar{x}) = f(\bar{x}) + R \sum_{k=1}^{K} \frac{1}{g_k(\bar{x})} + \sum_{\ell=1}^{L} \frac{h_\ell(\bar{x})}{\bar{R}}$$

FEATURES: To obtain a feasible starting point the penalty function of algorithm (2) is applied. The successive stages are generated by updating the penalty parameter by R*REDUCE where the value of REDUCE is much less than one.

(4). APPROX

AVAILABILITY: J. N. Siddall, School of Mechanical Engineering, McMaster University, Hamilton, Ontario, Canada.

REFERENCE: Siddall [51]

METHOD: Successive linear programming algorithm.

FEATURES: The method of Griffith and Stewart is employed, with the Simplex algorithm to handle the successive linear programming problems (Note: variable bounds are treated as inequality constraints).

(5). SIMPLX

AVAILABILITY: J. N. Siddall, School of Mechanical Engineering, McMaster University, Hamilton, Ontario, Canada.

REFERENCE: Siddall [51]

METHOD: Interior penalty function (same as for algorithm 3).

FEATURES: The simplex direct search method is employed for the unconstrained stages. The stages are updated as for algorithm 3.

(6). DAVID

AVAILABILITY: J. N. Siddall, School of Mechanical Engineering, McMaster University, Hamilton, Ontario, Canada.

REFERENCE: Siddall [51]

METHOD: Interior penalty function (same as for algorithm 3).

FEATURES: The Davidon-Fletcher-Powell technique is applied to the unconstrained stages. The stages are updated as for algorithm 3.

(7). MEMGRD

AVAILABILITY: J. N. Siddall, School of Mechanical Engineering, McMaster University, Hamilton, Ontario, Canada.

REFERENCE: Siddall [51]

METHOD: Interior penalty function (same as for algorithm 3).

FEATURES: Miele's memory gradient algorithm [52] is employed for the unconstrained penalty stages. The stages are updated as for algorithm 3.

(8). GRGDFP

AVAILABILITY: Proprietary code owned by the Whirlpool Corporation, Benton Harbor, Michigan.

REFERENCE: LaFrance [53]

METHOD: Generalized reduced gradient algorithm.

FEATURES: The Davidon-Fletcher Powell technique is used to generate the search directions. The introduction of slack variables for the inequality constraints and artificial variables for the violated constraints is required of the user.

(9). RALP

AVAILABILITY: S. B. Schuldt, Honeywell Corporate Research Center, Bloomington, Minnesota.

REFERENCE: Schuldt [54]

METHOD: Successive linear programming algorithm.

FEATURES: The method of Griffith and Stewart is employed with Newton's method applied to maintain equality constraint satisfaction. The bounded simplex method is employed to solve the successive linear programming problems.

(10). GRG

AVAILABILITY: Computer and Information Science Department, Cleveland State University, Cleveland, Ohio.

REFERENCE: Lasdon et all [55, 56]

METHOD: Generalized reduced gradient algorithm.

FEATURES: The Broyden-Fletcher-Shanno technique is used to generate the search directions. The search for a feasible starting point is accomplished by minimizing the sum of constraint violations again applying the BFS technique. Variable bounds are handled internally.

(11). OPT

AVAILABILITY: K. M. Ragsdell, School of Mechanical Engineering, Purdue University, W. Lafayette, Indiana.

REFERENCE: G. A. Gabriele [57]

METHOD: Generalized reduced gradient algorithm.

FEATURES: The conjugate gradient technique of Fletcher-Reeves is employed to generate the search directions. The search for a feasible starting point may be handled internally with the sum of the constraint violations minimized by Powells conjugate direction method or by the user by the introduction of artificial variables. The variable bounds are handled internally.

(12). GREG

AVAILABILITY: J. Abadie, Electricite' De France, Paris, France.

REFERENCE: Guigou [58]

Method: Generalized reduced gradient algorithm.

FEATURES: The conjugate gradient technique of Fletcher-Reeves is employed to generate the search directions. The search for a feasible starting point is handled internally by the introduction of artificial variables. The variable bounds are also handled internally.

(13). COMPUTE II METHOD 0

AVAILABILITY: Gulf Oil Corporation, Houston, Texas.

REFERENCE: Gulf Oil Corporation [59]

METHOD: Exterior penalty function of the form

$$P(\bar{x},\alpha,\delta) = f(\bar{x}) + \lambda\{ \sum_{\ell=1}^{L} \alpha_\ell h_\ell^2(\bar{x}) + \sum_{k=1}^{K} \alpha_k \delta_k g_k^2(\bar{x}) \}$$

where λ is the penalty coefficient, α_j are positive scale factors, and

$$\delta_k = 0 \text{ if } g_k(\bar{x}) \le 0$$

$$\delta_k = 1 \text{ if } g_k(\bar{x}) > 0$$

FEATURES: The Hooke-Jeeves direct search technique is applied to the unconstrained stages. The penalty stages are generated by multiplication of λ by a constant at each stage. Scale factors are employed to avoid domination of the penalty function by any constraint or group of constraints and are recalculated at the beginning of each stage. All input parameters are set internally and the initial penalty coefficient is set so that at the initial stage the penalty function is most sensitive to the objective function. The input parameters may be set by the user if he so desires. Variable bounds are handled internally.

(14). COMPUTE II METHOD 1

AVAILABILITY: Gulf Oil Corporation, Houston, Texas.

REFERENCE: Gulf Oil Corporation [59]

METHOD: Exterior penalty function (same as for algorithm 13).

FEATURES: Same as for algorithm 13 with the exception that the Conjugate Gradient technique of Fletcher-Reeves is employed to generate the search directions for the penalty stages.

(15). COMPUTE II METHOD 2

AVAILABILITY: Gulf Oil Corporation, Houston, Texas.

REFERENCE: Gulf Oil Corporation [59]

METHOD: Exterior penalty function (same as for algorithm 13).

FEATURES: Same as for algorithm 13 with the exception that the Davidon-Fletcher-Powell technique is employed to generate the search directions for the penalty stages.

(16). COMPUTER II METHOD 3

AVAILABILITY: Gulf Oil Corporation, Houston, Texas.

REFERENCE: Gulf Oil Corporation [59]

METHOD: Exterior penalty function (same as for algorithm 13).

FEATURES: Same as for algorithm 13 with the exception that Keefer's Simpat algorithm [60] is employed to generate search directions for the penalty stages. This method utilizes a pattern search technique for the variables near their bounds and the Simplex method of Nelder and Mead [61] for the rest of the variables.

(17). EXPEN #1

AVAILABILITY: R. W. Mayne, School of Mechanical Engineering, State University of New York at Buffalo.

REFERENCE: Afimiwala [62]

METHOD: Exterior penalty function of the form

$$P(x) = f(x) + R \sum_{k=1}^{K} \langle g_k(x)^2 \rangle + R \sum_{\ell=1}^{L} h_\ell(\bar{x})^2$$

FEATURES: A univariate search technique is applied to the successive penalty stages. Each stage is updated by multiplication of the penalty factor R.

(18). EXPEN #2

AVAILABILITY: R. W. Mayne, School of Mechanical Engineering, State Univesrity of New York at Buffalo.

REFERENCE: Afimiwala [62]

METHOD: Exterior penalty function (same as for algorithm 17).

FEATURES: The method of steepest descent is applied to the successive stages. The quadratic line search was used.

(19). EXPEN #3

AVAILABILITY: R. W. Mayne, School of Mechanical Engineering, State University of New York at Buffalo.

REFERENCE: Afimiwala [62]

METHOD: Exterior penalty function (same as for algorithm 17).

FEATURES: The Conjugate Direction method of Powell is applied to generate search directions for the successive stages. Again the quadratic line search was used.

(20). EXPEN #4

AVAILABILITY: R. W. Mayne, School of Mechanical Engineering, State University of New York at Buffalo.

REFERENCE: Afimiwala [62]

METHOD: Exterior penalty function (same as for algorithm 17).

FEATURES: The Conjugate Gradient technique of Fletcher-Reeves is applied to generate search directions for the successive stages. The quadratic line search was used.

(21). EXPEN #5

AVAILABILITY: R. W. Mayne, School of Mechanical Engineering, State University of New York at Buffalo.

REFERENCE: Afimiwala [62]

METHOD: Exterior penalty function (same as for algorithm 17).

FEATURES: The Variable Metric search technique of Davidon-Fletcher-Powell is applied to generate search directions for the successive stages. The quadratic line search was used.

(22). EXPEN #6

AVAILABILITY: R. W. Mayne, School of Mechanical Engineering, State University of New York at Buffalo.

REFERENCE: Afimiwala [62]

METHOD: Exterior penalty function (same as for algorithm 17).

FEATURES: A Hooke-Jeeves pattern search is applied to the successive stages.

(23). IPENAL #1

AVAILABILITY: R. W. Mayne, School of Mechanical Engineering, State University of New York at Buffalo.

REFERENCE: Afimiwala [62]

METHOD: Interior penalty function of the form

$$P(\overline{x}) = f(\overline{x}) - \sum_{k=1}^{K} \frac{R}{g_k(\overline{x})} + \frac{1}{\sqrt{R}} \sum_{\ell=1}^{L} h_\ell^2(\overline{x})$$

FEATURES: A univariate search technique is applied to the successive penalty stages. An extrapolation scheme is employed between stages to predict the minimum. The penalty was reduced by a constant to generate the next stage.

(24). IPENAL #2

AVAILABILITY: R. W. Mayne, School of Mechanical Engineering, State University of New York at Buffalo.

REFERENCE: Afimiwala [62]

METHOD: Interior penalty function (same as for algorithm 23).

FEATURES: The method of steepest descent is applied to the successive stages. The quadratic line search was used and the extrapolation scheme was used as on algorithm 23.

(25). IPENAL #3

AVAILABILITY: R. W. Mayne, School of Mechanical Engineering, State University of New York at Buffalo.

REFERENCE: Afimiwala [62]

METHOD: Interior penalty function (same as for algorithm 23).

FEATURES: The Conjugate Direction method of Powell is applied to generate search directions for the successive stages. The quadratic line search was used and the extrapolation scheme was used as on algorithm 23.

(26). IPENAL #4

AVAILABILITY: R. W. Mayne, School of Mechanical Engineering, State University of New York at Buffalo.

REFERENCE: Afimiwala [62]

METHOD: Interior penalty function (same as for algorithm 23).

FEATURES: The Conjugate Gradient techniqu of Fletcher-Reeves is applied to generate search directions for the successive stages. The quadratic line search was used and the extrapolation scheme was used as on algorithm 23.

(27). IPENAL #5

AVAILABILITY: R. W. Mayne, School of Mechanical Engineering, State University of New York at Buffalo.

REFERENCE: Afimiwala [62]

METHOD: Interior penalty function (same as for algorithm 23).

FEATURES: The Variable Metric search technique of Davidon-Fletcher-Powell is applied to generate search directions for the successive stages. The quadratic line search was used and the extrapolation scheme was used as an algorithm 23.

(28). SUMT-1

AVAILABILITY: Research Analysis Corporation, McLean, Virginia.

REFERENCE: Mylander et all [1]

METHOD: Interior penalty function of the form

$$P(\bar{x},R) = f(\bar{x}) - R \sum_{k=1}^{K} \ln g_k(\bar{x}) + \sum_{\ell=1}^{L} [h_\ell^2(\bar{x})/R]$$

FEATURES: The generalized Newton-Raphson method modified to handle indefinite Hessian matrices is employed to generate the search directions for the successive stages. The value of the penalty parameter R is reduced by a constant factor to form the successive stages. An acceleration procedure using the Lagrange extrapolation technique is applied.

(29). SUMT-2

AVAILABILITY: Research Analysis Corporation, McLean, Virginia.

REFERENCE: Mylander et all [1]

METHOD: Interior penalty function (same as for algorithm 28).

FEATURES: Same as for algorithm 28 with the exception that when an orthogonal move is made because of an indefinite Hessian matrix a negative gradient component is added to the orthogonal move vector.

(30). SUMT-3

AVAILABILITY: Research Analysis Corporation, McLean, Virginia.

REFERENCE: Mylander et all [1]

METHOD: Interior penalty function (same as for algorithm 28).

FEATURES: Same as for algorithm 28 with the exception that the method of steepest descent is used to generate search directions for the successive stages.

(31). SUMT-4

AVAILABILITY: Research Analysis Corporation, McLean, Virginia.

REFERENCE: Mylander et all [1]

METHOD: Interior penalty function (same as for algorithm 28).

FEATURES: Same as for algorithm 28 with the exception that a modified Fletcher-Powell Variable Metric technique is used to generate the search directions for the successive stages.

(32). E04HAF - 0

AVAILABILITY: The University of Liverpool Computer Laboratory, Liverpool, England.

REFERENCE: The University of Liverpool Computer Laboratory [63]

DESCRIPTION: Interior-Exterior penalty function of the form

$$P(\bar{x},R) = f(\bar{x}) - R \sum_{j=1}^{J} \log g_j(\bar{x}) + 1/R \sum_{j=J}^{K} \{\min(0,g_j(\bar{x}))\}^2$$

$$+ 1/R \sum_{\ell=1}^{L} h_\ell^2(\bar{x})$$

This penalty form was previously used by Lootsma [64]. The inequality constraints are divided into two subsets. The inequality constraints which are not violated at the starting point are contained in the log penalty term and those inequality constraints which are violated at the starting point are contained in the $g_j(\bar{x})^2$ penalty term.

FEATURES: The Conjugate Direction algorithm of Powell is employed to generate search directions for the successive stages. Each new stage is generated by reducing the penalty parameter by a constant. Polynomial extrapolation of the successive cycles is employed up to order 6 which is employed to accelerate the convergence of the method.

(33). E04HAF - 1

AVAILABILITY: The University of Liverpool Computer Laboratory, Liverpool, England.

REFERENCE: The University of Liverpool Computer Laboratory [63]

DESCRIPTION: Interior-Exterior penalty (same as for algorithm 32)

FEATURES: Same as for algorithm 32 with the exception that the Variable Metric technique of Broyden-Fletcher-Shanno is employed to generate search directions for the successive stages.

(34). E04HAF - 2

AVAILABILITY: The University of Liverpool Computer Laboratory, Liverpool, England.

REFERENCE: The Univesrity of Liverpool Computer Laboratory [63]

DESCRIPTION: Interior-Exterior penalty (same as for algorithm 32).

FEATURES: Same as for algorithm 32 with the exception that a modified Newton approach is employed to generate search directions for the successive stages.

(35). COMET

AVAILABILITY: D. M. Himmelblau, The University of Texas, Austin, Texas.

REFERENCE: Staha [65]

DESCRIPTION: Exterior penalty function of the form

$$P(\overline{x},R) = \text{Min}[0,\{t-f(\overline{x})\}]^2 + \sum_{k=1}^{K} \text{Min}\,[0,g_k(\overline{x})]^2 + \sum_{\ell=1}^{L} h_\ell^2(\overline{x})$$

FEATURES: This is another penalty technique which seeks to reduce the distortion of the penalty surface. The variable metric technique of Fletcher is employed to generate the search directions for the successive stages.

Appendix B

Test Problem Descriptions and Fortran Listings

(1). Test problem #1 used by Colville [27] and test problem #1 used by Eason and Fenton [40].

GENERAL INFORMATION:

5 Variables

10 Functional inequality constraints

5 Variable bounds; $x_j \geq 0 \quad j = 1, 2, \ldots, 5$

STARTING INFORMATION:

$\bar{x}_o = [0,0,0,0,1]$	$f(\bar{x}_o) = 2.0$
$g_1(\bar{x}_o) = 40$	$g_6(\bar{x}_o) = 1$
$g_2(\bar{x}_o) = 4$	$g_7(\bar{x}_o) = 39$
$g_3(\bar{x}_o) = .25$	$g_8(\bar{x}_o) = 59$
$g_4(\bar{x}_o) = 3$	$g_9(\bar{x}_o) = 0$
$g_5(\bar{x}_o) = 1.2$	$g_{10}(\bar{x}_o) = 0$

SOLUTION:

$\bar{x}^* = [.3, .33329998, .4, .42790241, .22435808]$

$f(\bar{x}^*) = 3.234866708$

Constraints #3, 5, 6 and 9 are active.

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #1

```
      FUNCTION F(X)
      DIMENSION X(1)
      DIMENSION E(5),C(5,5),D(5)
      DATA (C(I),I=1,25) /30.,-20.,-10.,32.,-10.,-20.,39.,-6.,
     1-31.,32.,-10.,-6.,10.,-6.,-10.,32.,-31.,-6.,39.,-20.,-10.,
     232.,-10.,-20.,30./
      DATA (E(I),I=1,5) /-15.,-27.,-36.,-18.,-12./
      DATA (D(I),I=1,5) /4.,8.,10.,6.,2./
      U1=0.0
      U2=0.0
      U3=0.0
      DO 10 J=1,5
      U1=U1+E(J)*X(J)
      U3=U3+D(J)*X(J)*X(J)*X(J)
      DO 10 I=1,5
      U2=U2+C(I,J)*X(I)*X(J)
   10 CONTINUE
      F=(U1+U2+U3)/10.0
      RETURN
      END
```

```
      SUBROUTINE CONST(X,NCONS,PHI)
      DIMENSION X(1),PHI(1),A(10,5),B(10)
      DATA (A(I),I=1,50) /-16.,0.,-3.5,0.,0.,2.,-1.,-1.,1.,1.,
     12.,-2.,0.,-2.,-9.,0.,-1.,-2.,2.,1.,0.,0.,2.,0.,-2.,-4.,-1.,
     2-3.,3.,1.,1.,4.,0.,-4.,1.,0.,-1.,-2.,4.,1.,0.,2.,0.,-1.,
     3-2.8,0.,-1.,-1.,5.,1./
      DATA (B(I),I=1,10) /-40.,-2.,-.25,-4.,-4.,-1.,-40.,-60.,5.,1./
      DO 5 I=1,10
      PHI(I)=0.0
      DO 2 J=1,5
    2 PHI(I)=PHI(I)+A(I,J)*X(J)
    5 PHI(I)=PHI(I)-B(I)
      DO 10 I=1,5
   10 PHI(10+I)=X(I)
      RETURN
      END
```

(2). Test problem #2 used by Eason and Fenton [40].

GENERAL INFORMATION:

3 Variables

2 Functional inequality constraints

6 Variable bounds $x_j \geq 0 \quad j=1,2,3$

$x_1 \leq 20 \quad x_2 \leq 11 \quad x_3 \leq 42$

STARTING INFORMATION:

$\bar{x}_0 = [10,10,10]$

$f(\bar{x}_o) = -1$

$g_1(\bar{x}_o) = 50 \qquad g_2(\bar{x}_o) = 22$

SOLUTION:

$\bar{x}^* = [20,11,15]$

$f(\bar{x}^*) \doteq -3.3$

Constraint #2 is active at the solution.

COMMENTS: This problem seeks to design a rectangular box to maximize volume, subject to post office restrictions on the length plus the girth and individual limits on the length, depth and height of the box.

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #2

```
FUNCTION F(X)
DIMENSION X(1)
F=-X(1)*X(2)*X(3)/1000.0
RETURN
END
```

```
SUBROUTINE CONST(X,NPHI,PHI)
DIMENSION X(1),PHI(1)
PHI(1)=X(1)+2.0*(X(2)+X(3))
PHI(2)=72.0-X(1)-2.0*(X(2)+X(3))
RETURN
END
```

(3). Test problem #3 used by Colville [27] and test problem #3 used by Eason and Fenton [40].

GENERAL INFORMATION:

5 Variables

6 Functional inequality constraints

10 Variable bounds

$78 \leq x_1 \leq 102$

$33 \leq x_2 \leq 45$

$27 \leq x_3 \leq 45$

$27 \leq x_4 \leq 45$

$27 \leq x_5 \leq 45$

STARTING INFORMATION:

$\bar{x}_o = [78.62, 33.44, 31.07, 44.18, 35.22]$

$f(\bar{x}_o) = 3.037395$

$g_1(\bar{x}_o) = 91.7927319$ $\quad g_4(\bar{x}_o) = 11.1070673$

$g_2(\bar{x}_o) = .207268111$ $\quad g_5(\bar{x}_o) = .131578229$

$g_3(\bar{x}_o) = 8.89293266$ $\quad g_6(\bar{x}_o) = 4.86842177$

SOLUTION:

$\bar{x}^* = [78, 33, 29.995256, 45, 36.775813]$

$f(\bar{x}^*) = 3.06655387$

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #3

```
      FUNCTION F(X)
      DIMENSION X(1)
      F=(5.3578547*X(3)*X(3)+.8356891*X(1)*X(5)+37.293239*X(1)
     1-40792.141)/10**4
      RETURN
      END
```

```
      SUBROUTINE CONST(X,NPHI,PHI)
      DIMENSION X(1),PHI(1)
      R1=85.334407+.0056858*X(2)*X(5)+.0006262*X(1)*X(4)-.0022053*
     1X(3)*X(5)
      R2=80.51249+.0071317*X(2)*X(5)+.0029955*X(1)*X(2)+.0021813*
     2X(3)*X(3)
      R3=9.300961+.0047026*X(3)*X(5)+.0012547*X(1)*X(3)+.0019085*
     3X(3)*X(4)
      PHI(1)=R1
      PHI(2)=92.0-R1
      PHI(3)=R2-90.
      PHI(4)=110.0-R2
      PHI(5)=R3-20.0
      PHI(6)=25.0-R3
      RETURN
      END
```

(4). Test problem #4 used by Colville [27] and test problem #4 used by Eason and Fenton [40] (Wood's Test Function).

GENERAL INFORMATION:

4 Variables

8 Variable bounds $-10 \leq x_j \leq 10 \quad j = 1,2,3,4$

STARTING INFORMATION:

$\bar{x}_o = [-3, -1, -3, -1]$

$f(\bar{x}_o) = 1.9192 \times 10^4$

SOLUTION:

$\bar{x}^* = [1, 1, 1, 1]$

$f(\bar{x}^*) = 0$

COMMENTS: This problem is a four dimensional version of Rosenbrock's test function.

FUNCTION LISTING FOR PROBLEM #4

```
      FUNCTION F(X)
      DIMENSION X(1)
      F=100.0*(X(2)-X(1)*X(1))**2+(1.-X(1))**2+90.*(X(4)-X(3)*X(3))**2
     1+10.1*((X(2)-1.0)**2+(X(4)-1.)**2)+19.8*(X(2)-1.)*(X(4)-1.)
     2+(1.0-X(3))**2
      RETURN
      END
```

(5). Test problem #5 used by Eason and Fenton [40] (Rosenbrock's Test Function).

GENERAL INFORMATION:

2 Variables

4 Variable bounds $-2 \le x_j \le 2$

STARTING INFORMATION:

$\bar{x}_o = [-1.2, 1]$

$f(\bar{x}_o) = 24.2$

SOLUTION:

$\bar{x}^* = [1, 1]$

$f(\bar{x}^*) = 0$

FUNCTION LISTING FOR PROBLEM #5

```
FUNCTION F(X)
DIMENSION X(1)
F=100.0*(X(2)-X(1)*X(1))**2+(1.0-X(1))**2
RETURN
END
```

(6). Test problem #6 used by Colville [27] and test problem #6 used by Eason and Fenton [40].

GENERAL INFORMATION:

6 Variables

4 Equality Constraints

12 Variable bounds

$0 \le x_1 \le 400$	$340 \le x_4 \le 420$
$0 \le x_2 \le 1000$	$-1000 \le x_5 \le 1000$
$340 \le x_3 \le 420$	$0 \le x_6 \le .5236$

STARTING INFORMATION:

$\bar{x}_o = [390, 1000, 419.5, 340.5, 191.175, .5]$

$f(\bar{x}_o) = 42.09$

$h_1(\bar{x}_o) = -577.149733$	$h_3(\bar{x}_o) = 505.628021$
$h_2(\bar{x}_o) = -485.540341$	$h_4(\bar{x}_o) = -389.350454$

SOLUTION #1:

$\bar{x}^* = [201.78617, 100, 382.96324, 419.9228, -10.784454, .07317686]$

$f(\bar{x}^*) = 8.85358521$

SOLUTION #2:

$\bar{x} = [107.8034355, 196.3274, 373.82968, 420, 21.311091, .15329950]$

$f(\bar{x}^*) = 8.927597736$

COMMENTS: This problem concerns the optimization of an electrical network (two nodes).

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #6

```
FUNCTION F(X)
DIMENSION X(1)
F=0.0
IF(X(1).GE.0.0.AND.X(1).LT.300.) F=F+30.0*X(1)
IF(X(1).GE.300.0) F=F+31.0*X(1)
IF(X(2).GE.0.0.AND.X(2).LT.100.0) F=F+28.0*X(2)
IF(X(2).GE.100.0.AND.X(2).LT.200.0) F=F+29.0*X(2)
IF(X(2).GE.200.0) F=F+30.0*X(2)
F=F/10**3
RETURN
END
```

```
SUBROUTINE EQUAL(X,PHI,NPSI)
DIMENSION X(1),PHI(1)
A=.90798
B=131.078
AA=.00889
BB=1.48477
C=300.0
D=200.0
PHI(1)=C-X(3)*X(4)/B*COS(BB-X(6))+X(3)*X(3)*A/B*COS(BB-AA)-X(1)
PHI(2)=-X(3)*X(4)/B*COS(BB+X(6))+X(4)*X(4)*A/B*COS(BB-AA)-X(2)
PHI(3)=D-X(3)*X(4)/B*SIN(BB-X(6))+X(3)*X(3)*A/B*SIN(BB-AA)
PHI(4)=-X(3)*X(4)/B*SIN(BB+X(6))+X(4)*X(4)*A/B*SIN(BB-AA)-X(5)
RETURN
END
```

(7). Test problem #7 used by Eason and Fenton [40].

GENERAL INFORMATION:

2 Variables

1 Functional inequality constraint

4 Variable bounds $0 \le x_j \le 5 \quad j = 1,2$

STARTING INFORMATION:

$\bar{x}_o = [2.5, 2.5]$

$f(\bar{x}_o) = .519472$

$g_1(\bar{x}_o) = -52.875$

SOLUTION:

$\bar{x}^* = [1.28667635, .53046168]$

$f(\bar{x}^*) = 1.62058332$

Constraint #1 is active.

COMMENTS: This problem concerns the design of a journal bearing to minimize a weighted function of frictional moment, angle of twist of the shaft, and lubrication oil temperature rise. The functional constraint expresses a minimum load-carrying requirement at a given speed. x_1 = radius of bearing; x_2 = bearing half length.

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #7

```
      FUNCTION F(X)
      DIMENSION X(1)
      F=(.44*X(1)*X(1)*X(1)/(X(2)*X(2))+10.0/X(1)+.592*X(1)/(X(2)
     1*X(2)*X(2)))/10.0
      RETURN
      END
```

```
      SUBROUTINE CONST(X,NPHI,PHI)
      DIMENSION X(1),PHI(1)
      PHI(1)=1.0-8.62*X(2)*X(2)*X(2)/X(1)
      RETURN
      END
```

(8). Test problem #8 used by Eason and Fenton [40].

GENERAL INFORMATION:

3 Variables

2 Functional inequality constraints

6 Variable bounds

$0 \le x_1 \le 36$ $\qquad$ $0 \le x_3 \le 125$

$0 \le x_2 \le 5$

STARTING INFORMATION:

$\bar{x}_0 = [22.3, .5, 125]$

$f(\bar{x}_o) = -3.88334111$

$g_1(\bar{x}_o) = 426.355000$ $\qquad$ $g_2(\bar{x}_o) = -.358015625$

SOLUTION:

$\bar{x}^* = [17.79933636, 2.1305717, 115.00142]$

$f(\bar{x}^*) = -5.6847825$

Constraints #1 and 2 are active.

COMMENTS: This problem concerns the design of a solid disk flywheel for maximum energy storage subject to constraints on the weight, diameter, speed of rotation and width. The distortion energy theory of failure provides a constraint based on the internal stresses.

- Alternate optima may be found -

x_1 = flywheel diameter; x_2 = flywheel thickness; x_3 = rotational speed

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #8

```
FUNCTION F(X)
DIMENSION X(1)
F=-.0201*X(1)*X(1)*X(1)*X(1)*X(2)*X(3)*X(3)/10**7
RETURN
END
```

```
SUBROUTINE CONST(X,NPHI,PHI)
DIMENSION X(1),PHI(1)
PHI(1)=675.0-X(1)*X(1)*X(2)
PHI(2)=.419-X(1)*X(1)*X(3)*X(3)/10**7
RETURN
END
```

(9). Test problem #9 used by Eason and Fenton [40].

GENERAL INFORMATION:

3 Variables

9 Functional inequality constraints

5 Variable bounds

$1000 \le x_1 \le 8000$ $100 \le x_2 \le 500$

$x_3 \le 114$

STARTING INFORMATION:

$\overline{x}_o = [5000, 200, 100]$

$f(\overline{x}_o) = -.8756765$

$g_1(\overline{x}_o) = 1.9917673$

$g_2(\overline{x}_o) = 6.4282327$

$g_3(\overline{x}_o) = 12.289098$

$g_4(\overline{x}_o) = 3937.7109$

$g_5(\overline{x}_o) = 57.796032$

$g_6(\overline{x}_o) = 1026.2167$

$g_7(\overline{x}_o) = 5780.1968$

$g_8(\overline{x}_o) = 53531.059$

$g_9(\overline{x}_o) = 175369.93$

SOLUTION:

$\overline{x}^* = [7828.7954, 188.81406, 113.81406]$

$f(\overline{x}^*) = -4.2446134$

No constraints active.

COMMENTS: This problem involves the design of a chemical reactor to maximize profit. The reactor flow rate, x_1, the reactor temperature, x_2, and the temperature drop in the cooling coil are to be selected. The constraints include an overall energy balance, a stability requirement, and a heat exchanger analysis. An additional variable bound was added to restrict the temperature drop in the cooling coil from approaching an infinite value.

FUNCTION LISTING FOR PROBLEM #9

```
      FUNCTION F(X)
      DIMENSION X(1)
      DATA P2,C1F,C2F,H1,H2,E1,E2,CPP,P/ 10.,.075,.025,8000.,8000.,
     11000.,1000.,3.6938503,20./
    9 P1=100.
      DO 35 I=1,3
      IF(X(I).LT.1.E-06) X(I)=1.E-06
   35 CONTINUE
      XK1=P1*EXP(-E1/(460.+X(2)))
      XK2=P2*EXP(-E2/(460.+X(2)))
      V=P*X(1)/(XK2*(X(1)*C2F-P))
      C1=(X(1)*C1F-P)/(X(1)+V*XK1)
      UT=43.+.0452*X(2)
   39 ARGU=(X(2)-X(3)-75.)/(X(2)-100.)
      IF(ARGU.EQ.0) GO TO 48
      XLMTD=(25.-X(3))/ALOG(ABS(ARGU))
      HEAT=X(1)*CPP*(100.-X(2))+XK1*(X(1)*C1F-P)*V*H1/(X(1)+V*XK1)+P*
      AREA=HEAT/(UT*XLMTD)
      ARE=ABS(AREA)
      HEA=ABS(HEAT)
      DIA=(V/12.72)**.33333333
      IF(X(2).LT.200.) GO TO 40
      PRESS=23.6+3.3E-06*(X(2)**3)
      GO TO 41
   48 X(2)=X(2)*1.0001
      GO TO 39
   40 PRESS=50.
   41 WATE=(.0909*(DIA**3)+.482*(DIA**2))*PRESS+36.6*(DIA**2)+160.5*D
      C1=4.8*(WATE**.782)
      IF(X(2).LT.200.) GO TO 42
      C2=(17.2+.0133*X(2))*DIA**2
      GO TO 43
   42 C2=0.
   43 IF(PRESS.LT.150.) GO TO 44
      C3=270.*(ARE**.546)*(.962+168.E-09*(X(2)**3))
      GO TO 45
   44 C3=270.*(ARE**.546)
   45 C4=1400.+140.*DIA
      C5=875.*((.05*V)**.3)
      C6=812.*(((6.95E-04+4.59E-11*(X(2)**3))+X(1))**.467)
      IF(X(2).LT.250) GO TO 46
      C7=1291.*((298.*HEA/X(3))**.467)
      GO TO 47
   46 C7=812.*((298.*HEA/X(3))**.467)
   47 COST=C1+C2+C3+C4+C5+C6+C7
      VEST=5.*COST
      C0=22000.+.18*VEST+3.10*V+61.1*((6.95E-04+4.59E-11*(X(2)**3))
     1*X(1))+.00115*HEAT+6.92*HEAT+574.*X(1)*(C1F-C1)+114800.
      F=(688000.-C0)/(2.*VEST)*(-1.E-03)
      RETURN
      END
```

CONSTRAINT LISTING FOR PROBLEM #9

```
      SUBROUTINE CONST(X,NPHI,PHI)
      DIMENSION X(1),PHI(1)
      DATA P2,C1F,C2F,H1,H2,E1,E2,CPP,P/ 10.,.075,.025,8000.,8000.,
     11000.,1000.,3.6938503,20./
    9 P1=100.
      DO 37 I=1,3
      IF(X(I).LT.1.E-06) X(I)=1.E-06
   37 CONTINUE
      XK1=P1*EXP(-E1/(460.+X(2)))
      XK2=P2*EXP(-E2/(460.+X(2)))
      V=P*X(1)/(XK2*(X(1)*C2F-P))
      C1=(X(1)*C1F-P)/(X(1)+V*XK1)
      UT=43.+.0452*X(2)
   36 ARGU=(X(2)-X(3)-75.)/(X(2)-100.)
      IF(ARGU.EQ.0.) GO TO 49
      XLMTD=(25.-X(3))/ALOG(ABS(ARGU))
      HEAT=X(1)*CPP*(100.-X(2))+XK1*(X(1)*C1F-P)*V*H1/(X(1)+V*XK1)+P*H2
      AREA=HEAT/(UT*XLMTD)
      DIA=(V/12.72)**.33333333
      IF(X(2).LT.200.) GO TO 44
      PRESS=23.6+3.3E-06*(X(2)**3)
      GO TO 45
   49 X(2)=X(2)*1.0001
      GO TO 36
   44 PRESS=50.
   45 PHI(1)=DIA-1.25
      PHI(2)=9.67-DIA
      PHI(3)=AREA-50.
      PHI(4)=4000.-AREA
      A11=XK1+X(1)/V
      A12=XK2
      A13=(X(1)*C1F-PRESS)*XK1*E1/((X(1)+V*XK1)*((X(2)+460.)**2))
     1+PRESS*E2/(V*((X(2)+460.)**2))
      A22=XK2+X(1)/V
      A23=PRESS*E2/(V*((X(2)+460.)**2))
      A31=-H1*XK1/CPP
      A32=-H2*XK2/CPP
      A33=X(1)/V+UT*AREA/(V*CPP)-(X(1)*C1F-PRESS)*XK1*E1*H1/((X(1)+V*
     1XK1)*CPP*((X(2)+460.)**2))-PRESS*E2*H2/(V*CPP*((X(2)+460.)**2))
      TEMP1=A11+A22+A33
      PHI(5)=TEMP1
      TEMP2=A11*A22+A22*A33+A33*A11-A13*A31-A23*A32
      PHI(6)=TEMP2
      TEMP3=A11*A22*A33+A12*A23*A31-A13*A31*A22-A23*A32*A11
      PHI(7)=TEMP3
      PHI(8)=TEMP1*TEMP2-TEMP3
      PHI(9)=HEAT
      RETURN
      END
```

(10). Test problem #10 used by Eason and Fenton [40].

GENERAL INFORMATION:

2 Variables

4 Variable bounds $1 \le x_j \le 3 \quad j = 1,2$

STARTING INFORMATION:

$\bar{x}_o = [.5, .5]$

$f(\bar{x}_o) = 2563.325$

SOLUTION:

$\bar{x}^* = [1.74347038, 2.02963554]$

$f(\bar{x}^*) = 1.744152006$

COMMENTS: This problem concerns the allocation of gear ratios in a triple reduction spur-gear train to achieve an overall reduction ratio of 10 with minimum gear train inertia.

FUNCTION LISTING FOR PROBLEM #10

```
      FUNCTION F(X)
      DIMENSION X(1)
      F=(10.0+1.0*(1.0+X(1)*X(1)+(1.0+X(2)*X(2))/(X(1)*X(1))+
     1(X(1)*X(1)*X(2)*X(2)+100.)/(X(1)*X(2))**4)+100./100.)/10.
      RETURN
      END
```

(11). Test problem #11 used by Eason and Fenton [40].

GENERAL INFORMATION:

2 Variables

2 Functional inequality constraints

4 Variable bounds $0 \leq x_j \leq 1.5 \quad j = 1,2$

STARTING INFORMATION:

$\bar{x}_o = [.75, .75]$

$f(\bar{x}_o) = 2.17360794$

$g_1(\bar{x}_o) = 15.5648030$

$g_2(x_o) = 44.4351970$

SOLUTION:

$\bar{x}^* = [.911398818, .02927999]$

$f(\bar{x}^*) = 1.1495014726$

Constraint #1 is active.

COMMENTS: This problem involves the design of a cam of minimum plate area. It involves specification of the offset between the cam center and the knife-edge follower in the initial position. A logarithmic follower function is required, and the pressure angle between the cam and the follower is limited to ±30° during a specified portion of one revolution.

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #11

```
      FUNCTION F(X)
      DIMENSION X(1)
      COMMON/A1/ PBIG
   11 CON=180./3.1415927
      PBIG=-360.
      DTR=1./CON
      TR=60./CON
      F=0.
      DO 49 I=1,100
      G=ALOG(TR)
      DY=1./TR
      XXX=(G+X(2))*SIN(TR)+X(1)*COS(TR)
      YYY=(G+X(2))*COS(TR)-X(1)*SIN(TR)
      RR=XXX*XXX+YYY*YYY
      F=F+.5*RR*DTR
      PANGLE=CON*ATAN(ABS((DY-X(1))/(G+X(2))))
      IF(PANGLE.GT.PBIG) PBIG=PANGLE
   49 TR=TR+DTR
      RETURN
      END
```

```
      SUBROUTINE CONST(X,NPHI,PHI)
      DIMENSION X(1),PHI(1)
      COMMON/A1/ PBIG
      CON=180./3.1415927
      PBIG=-360.
      DTR=1./CON
      TR=60./CON
      DO 49 I=1,100
      G=ALOG(TR)
      DY=1./TR
      PANGLE=CON*ATAN(ABS((DY-X(1))/(G+X(2))))
      IF(PANGLE.GT.PBIG) PBIG=PANGLE
   49 TR=TR+DTR
   11 PHI(1)=30.-PBIG
      PHI(2)=PBIG+30.
      RETURN
      END
```

(12). Test problem #12 used by Eason and Fenton [40].

GENERAL INFORMATION:

4 Variables

8 Variable bounds

$0 \le x_1 \le 150$ $\quad$ $0 \le x_3 \le 100$

$0 \le x_2 \le 50$ $\quad$ $0 \le x_4 \le 100$

STARTING INFORMATION:

$\bar{x}_o = [136, 0, 74.8, 75.5]$

$f(\bar{x}_o) = .36733873$

SOLUTION:

$\bar{x}^* = [136.00762, .031371415, 73.594390, 72.187426]$

$f(\bar{x}^*) = .35845660$

COMMENTS: This problem involves guiding a light weight assembly-line tool along a specified path. The crank is rotated in 10° intervals at a constant angular velocity and the objective function is the sum of the squared deviations of the generated points to the desired ones, and a function of the link lengths.

FUNCTION LISTING FOR PROBLEM #12

```
      FUNCTION F(X)
      DIMENSION X(1),XPT(36),YPT(36)
      DATA XPT/113.,110.1,106.2,101.3,95.4,88.8,81.6,74.,66.1,58.4,51.,
     144.3,38.7,34.5,32.4,32.9,36.4,42.8,50.9,59.,65.8,71.5,76.5,81.1,
     285.6,90.2,94.6,98.9,103.,106.7,109.9,112.5,114.4,115.5,115.7,114.
     39/,YPT/40.2,46.8,53.3,59.4,65.,69.9,73.9,76.9,78.9,79.8,79.7,78.5,
     476.5,73.6,70.2,66.,60.9,54.3,45.8,36.1,26.5,18.1,11.4,6.2,2.6,
     5.3,-.7,-.6,.7,3.1,6.4,10.5,15.5,21.,27.1,33.6/
      DATA P0,Q0,R0,S0/90.,0.,0.,0./
   12 DALPHA=3.141527/18.
      SUM=0.
      P1=X(1)
      Q1=X(2)
      R1=X(3)
      S1=X(4)
      DO 54 I=2,36
      ALPHA=DALPHA*(I-1)
      CA=COS(ALPHA)
      SA=SIN(ALPHA)
      PI=P1*CA-Q1*SA+P0*(1.-CA)+Q0*SA
      QI=P1*SA+Q1*CA+Q0*(1.-CA)-P0*SA
      A=R0*S1-S0*R1-Q1*R0+P1*S0+PI*Q1-P1*QI+QI*R1-PI*S1
      B=-R0*R1-S0*S1+P1*R0+Q1*S0-P1*PI-Q1*QI+PI*R1+QI*S1
      C=-R1*R0-S1*S0+PI*R0+QI*S0+P1*R1+Q1*S1-(P1*P1+Q1*Q1+PI*PI+
     1 QI*QI)/2.
      AABB=A*A+B*B
      IF(AABB.LT.1.E-30) GO TO 50
      TEST=C/SQRT(AABB)
      IF(ABS(TEST).GT.1.) GO TO 51
      J=1
   52 PH=ASIN(TEST)-ATAN(B/A)
   55 SP=SIN(PH)
      CP=COS(PH)
      RI=R1*CP-S1*SP+PI-P1*CP+Q1*SP
      SI=R1*SP+S1*CP+QI-P1*SP-Q1*CP
      TEST1=(R1-R0)**2+(S1-S0)**2
      IF(TEST1.LT.1.E-10) TEST1=1.E-10
      IF(ABS((TEST1-(RI-R0)**2-(SI-S0)**2)/TEST1).LT.0.001) GO TO 53
      IF(J.EQ.2) GO TO 51
      TEST=-TEST
      J=2
      GO TO 52
   50 PH=-ATAN(B/A)
      GO TO 55
   51 F=1.E20
      RETURN
   53 CALCX=XPT(1)*CP-YPT(1)*SP+PI-P1*CP+Q1*SP
      CALCY=XPT(1)*SP+YPT(1)*CP+QI-P1*SP-Q1*CP
   54 SUM=SUM+(CALCX-XPT(I))**2+(CALCY-YPT(I))**2
      SQL=(R1-R0)**2+(S1-S0)**2+(R1-P1)**2+(S1-Q1)**2
     1 +(P1-P0)**2+(Q1-Q0)**2
      F=SUM/100.+SQL/62500.
      RETURN
      END
```

(13). Test problem #13 used by Eason and Fenton [40].

GENERAL INFORMATION:

5 Variables

4 Functional inequality constraints

3 Variable bounds

$15 \leq x_1 \leq 20$ $\quad$ $x_5 \geq 2$

STARTING INFORMATION:

$\bar{x}_o = [15, 9.05, 6.14, 4.55, 3.61]$

$f(\bar{x}_o) = .2802$

$g_1(\bar{x}_o) = 5.95$ $\quad$ $g_3(\bar{x}_o) = 1.59$

$g_2(\bar{x}_o) = 2.91$ $\quad$ $g_4(\bar{x}_o) = .94$

SOLUTION:

$\bar{x}^* = [15.2632, 8.63583, 6.34419, 5.12674, 4.36837]$

$f(\bar{x}^*) = .2679$

No constraints active.

COMMENTS: This problem concerns modifying the gear ratios for a five speed automotive transmission in order to accelerate from rest to 100 MPH in minimum time.

The problem was modified in that the interpolation of torque values for engine speed was replaced by a series of cubic least squared error fits. This change was made to avoid the time consuming interpolation.

FUNCTION LISTING FOR PROBLEM #13

```
      FUNCTION F(X)
      DIMENSION X(1),RPM(1),TORQUE(1)
      DATA RAD,CON1,CON2,RPMIN,RPMAX,EI,VI,DT,VMAX,V0,TSHIFT,TMAX/
     1 1.085,1.466667,12.90842,600.,5700.,.600,98.,.01,100.
     2 ,5.,.25,100./
   13 IT=0
      ACC=0.0
      V=V0
      I=1
  302 FORCE=.0293*V**2+31.2
  301 RPM(1)=V*CON2*(X(I))
      IF(RPM(1).LT.RPMIN) GO TO 300
      IF(RPM(1).GT.RPMAX) GO TO 305
      IF(RPM(1).GE.RPMAX) GO TO 305
      IF(RPM(1).GE.600..AND.RPM(1).LE.1900.) TORQUE(1)=
     1 .00000003846154*RPM(1)**3-.0002108974359*RPM(1)**2+
     2 .42455128205133*RPM(1)-187.11538461540295
      IF(RPM(1).GE.1900..AND.RPM(1).LE.3000.) TORQUE(1)=
     1-.00000000492424*RPM(1)**3+.00001867424242*RPM(1)**2+
     2 .01229545454547*RPM(1)+64.999999999986
      IF(RPM(1).GE.3000..AND.RPM(1).LE.4500.) TORQUE(1)=
     1 -.0000000026667*RPM(1)**3+.000003*RPM(1)**2-
     2 .01263333333336*RPM(1)+155.1000000002947
      IF(RPM(1).GE.4500..AND.RPM(1).LT.5600.) TORQUE(1)=
     1 -.00000000664141*RPM(1)**3+.00008337626263*RPM(1)**2-
     2 .34351868688129*RPM(1)+597.3636363847145
      IF(RPM(1).GE.5600..AND.RPM(1).LE.6000.) TORQUE(1)=
     1 -.00000025396B3*RPM(1)**3+.00038158730157*RPM(1)**2
     2 -1.9223492062348*RPM(1)+3380.66666645715304
      ACCO=ACC
      ACC=RAD*(X(I)*TORQUE(1)-FORCE*RAD)/(EI*X(I)**2+VI)
      IT=IT+1
      T=DT*IT
      V=V+(ACCO+ACC)/2.*DT/CON1
      IF(T.GT.TMAX) GO TO 311
      IF(V.GE.VMAX) GO TO 311
      GO TO 302
  300 F=TMAX
      RETURN
  305 I=I+1
      IF(T.EQ.0.) GO TO 301
      TT=T+TSHIFT
  306 ACC=-FORCE*RAD**2/VI
      IT=IT+1
      T=DT+IT
      V=V+ACC*DT/CON1
      IF(T.LT.TT) GO TO 307
      GO TO 302
  307 FORCE=.0293*V*V+31.2
      GO TO 306
  311 F=T/100.
      RETURN
      END
```

CONSTRAINT LISTING FOR PROBLEM #13

```
SUBROUTINE CONST(X,NPHI,PHI)
DIMENSION X(1),PHI(1)
PHI(1)=X(1)-X(2)
PHI(2)=X(2)-X(3)
PHI(3)=X(3)-X(4)
PHI(4)=X(4)-X(5)
RETURN
END
```

(14). Test problem #2 used by Colville [27].

GENERAL INFORMATION:

15 Variables

5 Functional inequality constraints

10 Variable bounds $x_j \geq 0$ $j=1,2,3,\ldots,15$

STARTING INFORMATION:

$x_{o_j} = .0001$ $j=1,2,\ldots,15,$ $j \neq 7;$ $x_{o_7} = 60$

$f(\bar{x}_o) = -2400.011$

$g_1(\bar{x}_o) = 45.00605$

$g_2(\bar{x}_o) = 33.00380$

$g_3(\bar{x}_o) = 23.99590$

$g_4(\bar{x}_o) = 42.00230$

$g_5(\bar{x}_o) = 48.00408$

SOLUTION:

$\bar{x}^* = [0, 0, 5.1736360, 0, 3.0612393, 11.8389158, 0, 0, .10376918, 0, .3000238, .33343802, .40002035, .4282753906, .223971045]$

$f(\bar{x}^*) = 32.3486790$

Constraints #1, 2, 3, 4 and 5 are active.

COMMENTS: This problem is the dual to problem #1 of the comparative study.

FUNCTION LISTING FOR PROBLEM #14

```
      FUNCTION F(X)
      DIMENSION X(1)
      COMMON/COF/A(10,5),B(10),C(5,5),D(5),E(5)
      DATA (C(I),I=1,25) /30.,-20.,-10.,32.,-10.,-20.,39.,-6.,
     1-31.,32.,-10.,-6.,10.,-6.,-10.,32.,-31.,-6.,39.,-20.,-10.,
     232.,-10.,-20.,30./
      DATA (B(I),I=1,10)/-40.,-2.,-.25,-4.,-4.,-1.,-40.,-60.,5.,1./
      DATA (D(I),I=1,5) /4.,8.,10.,6.,2./
      U1=0.0
      U2=0.0
      U3=0.0
      DO 100 I=1,10
      U1=B(I)*X(I)+U1
  100 CONTINUE
      DO 101 J=1,5
      DO 101 I=1,5
      U2=C(I,J)*X(10+I)*X(10+J)+U2
  101 CONTINUE
      DO 102 J=1,5
      U3=D(J)*X(10+J)**3+U3
  102 CONTINUE
      F=-(U1-U2-2.*U3)
      RETURN
      END
```

CONSTRAINT LISTING FOR PROBLEM #14

```
      SUBROUTINE CONST(X,NCONS,CON)
      DIMENSION X(1),CON(1)
      COMMON/COF/ A(10,5),B(10),C(5,5),D(5),E(5)
      DATA (A(I),I=1,50) /-16.,0.,-3.5,0.,0.,2.,-1.,-1.,1.,1.,
     12.,-2.,0.,-2.,-9.,0.,-1.,-2.,2.,1.,0.,0.,2.,0.,-2.,-4.,-1.,
     2-3.,3.,1.,1.,4.,0.,-4.,1.,0.,-1.,-2.,4.,1.,0.,2.,0.,-1.,
     3-2.8,0.,-1.,-1.,5.,1./
      DATA (E(I),I=1,5)/-15.,-27.,-36.,-18.,-12./
      DATA (D(I),I=1,5)/4.,8.,10.,6.,2./
      DATA (C(I),I=1,25) /30.,-20.,-10.,32.,-10.,-20.,39.,-6.,
     1-31.,32.,-10.,-6.,10.,-6.,-10.,32.,-31.,-6.,39.,-20.,-10.,
     232.,-10.,-20.,30./
      DO 100 J=1,5
      C1=0.0
      DO 101 I=1,5
      C1=C(I,J)*X(10+I)+C1
101   CONTINUE
      C2=0.0
      DO 102 I=1,10
      C2=A(I,J)*X(I)+C2
102   CONTINUE
      CON(J)=E(J)+2.*C1+3.*D(J)*X(10+J)**2-C2
100   CONTINUE
      RETURN
      END
```

(15). Test problem #7 used by Colville [27].

GENERAL INFORMATION:

16 Variables

8 Equality constraints

32 Variable bounds $0 \leq x_j \leq 5 \quad j=1,2,3, \ldots, 16$

STARTING INFORMATION:

$x_{o_j} = 0 \quad j=1,2,3, \ldots, 16$

$f(\bar{x}_o) = 46.0$

$h_1(\bar{x}_o) = -2.5$	$h_5(\bar{x}_o) = -1.3$
$h_2(\bar{x}_o) = -1.1$	$h_6(\bar{x}_o) = -2.1$
$h_3(\bar{x}_o) = 3.1$	$h_7(\bar{x}_o) = -2.3$
$h_4(\bar{x}_o) = 3.5$	$h_8(x_o) = 1.5$

SOLUTION:

$\bar{x}^* = [.03981344, .79197966, .2029019, .84431475,$
$1.26985898, .934787236, 1.68198142, .155235257,$
$1.5678912575, 0, 0, 0, .66016359, 0, .674293038, 0]$

$f(\bar{x}^*) = 244.89969778$

FUNCTION LISTING FOR PROBLEM #15

```
      FUNCTION F(X)
      DIMENSION X(1),A(16,16)
      DATA (A(I),I=1,256)/1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
     10.,0.,0.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
     20.,1.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
     31.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
     40.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
     50.,0.,0.,0.,1.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
     61.,1.,1.,1.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
     71.,0.,0.,0.,0.,1.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,
     80.,0.,1.,0.,0.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,
     90.,1.,1.,0.,1.,0.,0.,1.,0.,1.,0.,0.,0.,0.,0.,0.,
     90.,0.,0.,1.,0.,0.,1.,0.,0.,0.,1.,0.,0.,0.,0.,0.,
     10.,0.,0.,0.,1.,0.,0.,0.,1.,0.,0.,1.,0.,0.,0.,0.,
     20.,0.,0.,0.,0.,0.,1.,0.,0.,0.,1.,0.,1.,0.,0.,0.,
     30.,0.,1.,0.,0.,0.,0.,0.,0.,1.,0.,1.,1.,1.,0.,0.,
     40.,0.,0.,1.,0.,1.,0.,1.,0.,0.,0.,0.,0.,0.,1.,0.,
     51.,0.,0.,0.,1.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,1./
      F=0.0
      DO 100 I=1,16
      F1=X(I)*X(I)+X(I)+1.
      DO 100 J=1,16
      F2=X(J)*X(J)+X(J)+1.
      F=A(I,J)*F1*F2+F
  100 CONTINUE
      RETURN
      END
```

CONSTRAINT LISTING FOR PROBLEM #15

```
      SUBROUTINE EQUAL(X,CON,NEQUS)
      DIMENSION X(1),CON(1),B(8,16),C(8)

      DATA (B(I),I=1,128)/.22,-1.46,1.29,-1.1,0.,0.,1.12,0.,
     1.2,0.,-.89,-1.06,0.,-1.72,0.,.45,.19,-1.30,0.,.95,0.,-.33,
     20.,.26,.25,1.82,0.,-.54,-1.43,0.,.31,-1.1,.15,-1.15,
     3-1.16,0.,1.51,1.62,0.,.58,.11,0.,-.96,-1.78,.59,1.24,0.,0.,
     4.12,.8,0.,-.41,-.33,.21,1.12,-1.03,.13,0.,-.49,0.,-.43,-.26,
     50.,.1,1.,0.,0.,0.,0.,0.,-.36,0.,0.,1.,0.,0.,0.,0.,0.,0.,
     60.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,0.,0.,0.,
     70.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,1.,0.,0.,
     80.,0.,0.,0.,0.,0.,1.,0.,0.,0.,0.,0.,0.,0.,0.,1./
      DATA (C(I),I=1,8) /2.5,1.1,-3.1,-3.5,1.3,2.1,2.3,-1.5/

      DO 100 I=1,8
      CON(I)=0.0
      DO 101 J=1,16
      CON(I)=B(I,J)*X(J)+CON(I)
  101 CONTINUE
      CON(I)=CON(I)-C(I)
  100 CONTINUE
      RETURN
      END
```

(16). Test problem #8 used by Colville [27].

GENERAL INFORMATION:

3 Variables

14 Functional inequality constraints

6 Variable bounds $0 \le x_1 \le 2000$;

$0 \le x_2 \le 16{,}000$; $0 \le x_3 \le 120$

STARTING INFORMATION:

$\bar{x}_o = [1745, 12000, 110]$

$f(\bar{x}_o) = -868.645762$

$g_1(\bar{x}_o) = .390342104$	$g_8(\bar{x}_o) = .332668903$
$g_2(\bar{x}_o) = .013043417$	$g_9(\bar{x}_o) = 355.105637$
$g_3(\bar{x}_o) = .049385745$	$g_{10}(\bar{x}_o) = .109735908$
$g_4(\bar{x}_o) = .040883997$	$g_{11}(\bar{x}_o) = .002141772$
$g_5(\bar{x}_o) = .037798404$	$g_{12}(\bar{x}_o) = .103021254$
$g_6(\bar{x}_o) = .023471730$	$g_{13}(\bar{x}_o) = 3048.28948$
$g_7(\bar{x}_o) = 1.66932439$	$g_{14}(\bar{x}_o) = 1973.91317$

SOLUTION:

$\bar{x}^* = [1728.37078, 16000, 98.1323813]$

$f(\bar{x}^*) = -1162.0363352$

Constraints #2 is active.

COMMENTS: This is a process optimization problem developed by Colville. The constraints were scaled to allow for better convergence to the solution.

FUNCTION AND AUXILIARY SUBROUTINE LISTING FOR PROBLEM #16

```
      FUNCTION F(X)
      DIMENSION X(1),Y(8)
      CALL SUPPLE(X,Y)
      F=0.063*Y(2)*Y(5)-5.04*X(1)-3.36*Y(3)-0.035*X(2)-10.*X(3)
      RETURN
      END
```

```
      SUBROUTINE SUPPLE(X,Y)
      DIMENSION X(1),Y(1)
      Y(2)=1.6*X(1)
   10 Y(3)=1.22*Y(2)-X(1)
      Y(6)=(X(2)+Y(3))/X(1)
      Y2CALC=X(1)*(112.+13.167*Y(6)-0.6667*Y(6)**2)/100.
      IF(ABS(Y2CALC-Y(2))-0.001) 30,30,20
   20 Y(2)=Y2CALC
      GO TO 10
   30 CONTINUE
      Y(4)=93.0
  100 Y(5)=86.35+1.098*Y(6)-0.038*Y(6)**2+0.325*(Y(4)-89.)
      Y(8)=-133.+3.*Y(5)
      Y(7)=35.82-.222*Y(8)
      Y4CALC=98000.*X(3)/(Y(2)*Y(7)+X(3)*1000.)
      IF(ABS(Y4CALC-Y(4))-0.0001) 300,300,200
  200 Y(4)=Y4CALC
      GO TO 100
  300 CONTINUE
      RETURN
      END
```

CONSTRAINT LISTING FOR PROBLEM #16

```
SUBROUTINE CONST(X,NCONS,CON)
DIMENSION X(1),CON(1),Y(8)
CALL SUPPLE(X,Y)
CON(1)=(5000.-Y(2))/5000.
CON(2)=(2000.-Y(3))/2000.
CON(3)=(Y(4)-85.)/85.
CON(4)=(93.-Y(4))/93.
CON(5)=(Y(5)-90.)/90.
CON(6)=(95.-Y(5))/95.
CON(7)=(Y(6)-3.)/3.
CON(8)=(12.-Y(6))/12.
CON(9)=(Y(7)-0.01)/0.01
CON(10)=(4.-Y(7))/4.
CON(11)=(Y(8)-145.)/145.
CON(12)=(162.-Y(8))/162.
CON(13)=Y(2)
CON(14)=Y(3)
RETURN
END
```

(17). Test problem #1 used by Dembo [46].

GENERAL INFORMATION:

12 Variables

3 Functional inequality constraints

24 Variable bounds $.1 \le x_j \le 100$ $j = 1,2,3,\ldots,12$

STARTING INFORMATION:

$x_{o_j} = 4.0$ $j = 1,2,3,\ldots,12$

$f(\overline{x}_o) = .227682649$

$g_1(x_o) = .199810679$

$g_2(x_o) = -.757076016$

$g_3(\overline{x}_o) = -.754318463$

SOLUTION:

$\overline{x}^* = [2.6631947068, 4.517277762, 7.133802907, 2.237268448, 4.07840382657, 1.31827569, 4.125187034, 2.856195978, 1.6765929748, 2.1789111052, 5.12343515, 6.659338016]$

$f(\overline{x}^*) = 3.16859000$

Constraints #1, 2, and 3 are active.

COMMENTS: The problems involves a multiphase chemical equilibrium calculation. The scaled version of the problem was used.

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #17

```
      FUNCTION F(X)
      DIMENSION X(1)
      DIMENSION A(11)
      DATA A/-.00133172,-.002270927,-.00248546,-4.67,-4.671973,
     1 -.008140,-.008092,-.005,-.000909,-.00088,-.00119/
      F=1.00E+05
      DO 10 I=1,11
      TEMP=X(I)
      IF(X(I).LT.1.0E-15) TEMP=1.0E-15
   10 F=F*TEMP**A(I)
      RETURN
      END
```

```
      SUBROUTINE CONST(X,NPHI,PHI)
      DIMENSION X(1),PHI(1),C(30)
      DATA C/5.367373E-02,2.1863746E-02,9.7733533E-02,6.6940803E-03,
     1 1.0E-06,1.0E-05,1.0E-06,1.0E-10,1.0E-08,1.0E-02,1.0E-04,
     2 1.0898645E-01,1.6108052E-04,1.0E-23,1.9304541E-06,1.0E-03,
     3 1.0E-06,1.0E-05,1.0E-06,1.0E-09,1.0E-09,1.0E-03,1.0E-03,
     4 1.0898645E-01,1.6108052E-05,1.0E-23,1.9304541E-08,1.0E-05,
     5 1.1184059E-04,1.0E-04/
      PHI(1)=1.0-C(1)*X(1)-C(2)*X(2)-C(3)*X(3)-C(4)*X(4)*X(5)
      PHI(2)=1.0-C(5)*X(1)-C(6)*X(2)-C(7)*X(3)-C(8)*X(4)*X(12)-C(9)*
     1 X(5)/X(12)-C(10)*X(6)/X(12)-C(11)*X(7)*X(12)-C(12)*X(4)*X(5)-
     2 C(13)*X(2)*X(5)/X(12)-C(14)*X(2)*X(4)*X(5)-C(15)*X(2)/X(4)*X(5)
     3 /X(12)**2-C(16)*X(10)/X(12)
      PHI(3)=1.0-C(17)*X(1)-C(18)*X(2)-C(19)*X(3)-C(20)*X(4)-C(21)*X(5)
     1-C(22)*X(6)-C(23)*X(8)-C(24)*X(4)*X(5)-C(25)*X(2)*X(5)-
     2 C(26)*X(2)*X(4)*X(5)-C(27)*X(2)*X(5)/X(4)-C(28)*X(9)-
     3 C(29)*X(1)*X(9)-C(30)*X(11)
      RETURN
      END
```

(18). Test problem #3 used by Dembo [46].

GENERAL INFORMATION:

7 Variables

14 Functional inequality constraints

14 Variables bounds

$$1 \le x_1 \le 2000$$
$$1 \le x_2 \le 120$$
$$1 \le x_3 \le 5000$$
$$85 \le x_4 \le 93$$
$$90 \le x_5 \le 95$$
$$3 \le x_6 \le 12$$
$$145 \le x_7 \le 162$$

STARTING INFORMATION:

$\bar{x}_o = [1745, 110, 3048, 89, 92.8, 8, 145]$

$f(\bar{x}_o) = 2125.6598$

$g_1(\bar{x}_o) = .0153948067$	$g_8(\bar{x}_o) = .0080782000$
$g_2(\bar{x}_o) = .0101274528$	$g_9(\bar{x}_o) = .0132200000$
$g_3(\bar{x}_o) = .0144536405$	$g_{10}(\bar{x}_o) = .530463480$
$g_4(\bar{x}_o) = .0110454741$	$g_{11}(\bar{x}_o) = .087631602$
$g_5(\bar{x}_o) = .0095946752$	$g_{12}(\bar{x}_o) = .061639448$
$g_6(\bar{x}_o) = .0128187200$	$g_{13}(\bar{x}_0) = .250847500$
$g_7(\bar{x}_o) = .0066451185$	$g_{14}(\bar{x}_o) = 6.86844699$

SOLUTION:

$\bar{x}^* = [1698.10594, 53.7010394, 3031.2343436, 90.1147228, 95, 10.49405556, 153.53535167]$

$f(\bar{x}^*) = 1227.2272509$

Constraints #1, 3, 6, 7 and 9 are active.

COMMENTS: The problem involves an alkylation process optimization.

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #18

```
      FUNCTION F(X)
      DIMENSION C(6),X(1)
      DATA C/1.715,.035,4.0565,10.0,3000.0,-.063/
      F=C(1)*X(1)+C(2)*X(1)*X(6)+C(3)*X(3)+C(4)*X(2)+C(5)+C(6)*X(3)*X(5)
      RETURN
      END
```

```
      SUBROUTINE CONST(X,NPHI,PHI)
      DIMENSION X(1),PHI(1),C(38)
      DATA C/.59553571E-02,.88392857,-.11756250,1.1088,.1303533,-.006603
     1 3,.66173269E-03,.17239878E-01,-.56595559E-02,-.19120592E-01,
     2 .5685075E+02,1.08702,.32175,-.03762,.006198,.24623121E+04,
     3 -.25125634E+02,.16118996E+03,5000.,-.48951E+06,.44333333E+02,
     4 .33,.022556,-.007595,.00061,-.0005,.819672,.819672,24500.,-250.,
     5 .10204082E-01,.12244898E-04,.00006250,.00006250,-.00007625,1.22,
     6 1.0,-1.0/
      PHI(1)=1.0-C(1)*X(6)*X(6)-C(2)*X(3)/X(1)-C(3)*X(6)
      PHI(2)=1.0-C(4)*X(1)/X(3)-C(5)*X(1)/X(3)*X(6)-C(6)*X(1)/X(3)*X(6)
     1 *X(6)
      PHI(3)=1.0-C(7)*X(6)*X(6)-C(8)*X(5)-C(9)*X(4)-C(10)*X(6)
      PHI(4)=1.0-C(11)/X(5)-C(12)/X(5)*X(6)-C(13)*X(4)/X(5)-C(14)/X(5)
     1 *X(6)*X(6)
      PHI(5)=1.0-C(15)*X(7)-C(16)*X(2)/X(3)/X(4)-C(17)*X(2)/X(3)
      PHI(6)=1.0-C(18)/X(7)-C(19)*X(2)/X(3)/X(7)-C(20)*X(2)/X(3)/X(4)
     1 /X(7)
      PHI(7)=1.0-C(21)/X(5)-C(22)*X(7)/X(5)
      PHI(8)=1.0-C(23)*X(5)-C(24)*X(7)
      PHI(9)=1.0-C(25)*X(3)-C(26)*X(1)
      PHI(10)=1.0-C(27)*X(1)/X(3)-C(28)/X(3)
      PHI(11)=1.0-C(29)*X(2)/X(3)/X(4)-C(30)*X(2)/X(3)
      PHI(12)=1.0-C(31)*X(4)-C(32)/X(2)*X(3)*X(4)
      PHI(13)=1.0-C(33)*X(1)*X(6)-C(34)*X(1)-C(35)*X(3)
      PHI(14)=1.0-C(36)/X(1)*X(3)-C(37)/X(1)-C(38)*X(6)
      RETURN
      END
```

(19). Test problem #4 used by Dembo [46].

GENERAL INFORMATION:

8 Variables

4 Functional inequality constraints

16 Variable bounds $.1 \le x_j \le 10 \quad j = 1,2,3, \ldots, 8$

STARTING INFORMATION:

$\bar{x}_o = [6, 3, .4, .2, 6, 6, 1, .5]$

$f(\bar{x}_o) = 3.65736570$

$g_1(\bar{x}_o) = .04720000$ $g_3(\bar{x}_o) = -.099050230$

$g_2(\bar{x}_o) = -.07640000$ $g_4(\bar{x}_o) = -.416644828$

SOLUTION:

$\bar{x}^* = [6.465036554, 2.23275840, .6674155016, .5957723857,$
$5.932688789, 5.52724000, 1.0133420, .400676365]$

$f(\bar{x}^*) = 3.951163444$

Constraints #1, 2, 3 and 4 are active.

COMMENTS: The problem involves the design of a reactor.

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #19

```
      FUNCTION F(X)
      DIMENSION X(1)
      A=1.0E-15
      IF(X(1).GT.A.AND.X(2).GT.A.AND.X(7).GT.A.AND.X(8).GT.A)
     1 GO TO 10
      F=1.0E5
      RETURN
   10 F=0.4*X(1)**.67*X(7)**(-.67)+0.4*X(2)**.67*X(8)**(-.67)+10.0-X(1)
     1 -X(2)
      RETURN
      END
```

```
      SUBROUTINE CONST(X,NPHI,PHI)
      DIMENSION X(1),PHI(1)
      PHI(1)=1.0-0.0588*X(5)*X(7)-0.1*X(1)
      PHI(2)=1.0-0.0588*X(6)*X(8)-0.1*X(1)-0.1*X(2)
      IF(X(3).GT.1.0E-15) GO TO 4
      PHI(3)=-1.0E5
      GO TO 5
    4 PHI(3)=1.0-4.0*X(3)/X(5)-2.0*X(3)**(-.71)/X(5)-.0588*X(3)
     1 **(-1.3)*X(7)
    5 IF(X(4).GT.1.0E-15) GO TO 6
      PHI(4)=-1.0E5
      GO TO 7
    6 PHI(4)=1.0-4.0*X(4)/X(6)-2.0*X(4)**(-.71)/X(6)-.0588*X(4)
     1 **(-1.3)*X(8)
    7 RETURN
      END
```

(20). Test problem #5 used by Dembo [46].

GENERAL INFORMATION:

8 Variables

6 Functional inequality constraints

16 Variable bounds

$10000 \le x_1 \le 100$	$1000 \le x_5 \le 10$
$10000 \le x_2 \le 1000$	$1000 \le x_6 \le 10$
$10000 \le x_3 \le 1000$	$1000 \le x_7 \le 10$
$1000 \le x_4 \le 10$	$1000 \le x_8 \le 10$

STARTING INFORMATION:

$\bar{x}_o$ = [5000, 5000, 5000, 200, 350, 150, 225, 425]

$f(\bar{x}_o) = 15000.0$

$g_1(\bar{x}_o) = .222222439$	$g_4(\bar{x}_o) = .125$
$g_2(\bar{x}_o) = -.05555556$	$g_5(\bar{x}_o) = .0625$
$g_3(\bar{x}_o) = 3.5527 \times 10^{-15}$	$g_6(\bar{x}_o) = .250$

SOLUTION:

$\bar{x}^*$ = [579.179816, 1359.9511, 5110.11713, 182.00710, 295.595315, 217.992897, 286,411787, 395.595315]

$f(\bar{x}^*) = 7049.248049$

Constraints #1, 2, 3, 4, 5 and 6 are active.

COMMENTS: The problem involves the design of a heat exchanger.

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #20

```
      FUNCTION F(X)
      DIMENSION X(1)
      F=X(1)+X(2)+X(3)
      RETURN
      END
```

```
      SUBROUTINE CONST(X,NPHI,PHI)
      DIMENSION X(1),PHI(1),C(16)
      DATA C/833.33252,100.0,-83333.333,1250.0,1.0,-1250.0,1250000.0
     1 ,1.0,-2500.0,.0025,.0025,.0025,.0025,-.0025,.01,-.01/
      PHI(1)=1.0-C(1)/X(1)*X(4)/X(6)-C(2)/X(6)-C(3)/X(1)/X(6)
      PHI(2)=1.0-C(4)/X(2)*X(5)/X(7)-C(5)*X(4)/X(7)-C(6)/X(2)*X(4)/X(7)
      PHI(3)=1.0-C(7)/X(3)/X(8)-C(8)*X(5)/X(8)-C(9)/X(3)*X(5)/X(8)
      PHI(4)=1.0-C(10)*X(4)-C(11)*X(6)
      PHI(5)=1.0-C(12)*X(5)-C(13)*X(7)-C(14)*X(4)
      PHI(6)=1.0-C(15)*X(8)-C(16)*X(5)
      RETURN
      END
```

(21). Test problem #6 used by Dembo [46].

GENERAL INFORMATION:

13 Variables

13 Functional inequality constraints

26 Variable bounds

$1 \le x_1 \le .1$	$1000 \le x_8 \le .1$
$1 \le x_2 \le .1$	$1000 \le x_9 \le 500$
$1 \le x_3 \le .9$	$500 \le x_{10} \le .1$
$.1 \le x_4 \le .0001$	$150 \le x_{11} \le 1$
$.9 \le x_5 \le .1$	$150 \le x_{12} \le .0001$
$.9 \le x_6 \le .1$	$150 \le x_{13} \le .0001$
$1000 \le x_7 \le .1$	

STARTING INFORMATION:

$\bar{x}_o = [.5, .8, .9, .1, .14, .50, 489, 80, 650, 450, 150, 150, 150]$

$f(\bar{x}_o) = 450.0$

$g_1(\bar{x}) = .654881867$	$g_8(\bar{x}) = 1.421 \times 10^{-14}$
$g_2(\bar{x}) = -.20370813$	$g_9(\bar{x}) = -.012500000$
$g_3(\bar{x}) = .535981300$	$g_{10}(\bar{x}) = .111111111$
$g_4(\bar{x}) = .066000000$	$g_{11}(\bar{x}) = .375000000$
$g_5(\bar{x}) = .075745000$	$g_{12}(\bar{x}) = .182000000$
$g_6(\bar{x}) = -.181711949$	$g_{13}(\bar{x}) = .363125000$
$g_7(\bar{x}) = .027600000$	

SOLUTION:

$\bar{x}^* = [.80377316, .9, .944411275, .1, .190821994, .304576338, 574.085832, 74.0858325, 500.00746, .1, 20.2353465, 77.3430724, .01]$

(21). (cont'd)

$f(\overline{x}^*) = 97.5884189$

Constraints #1, 2, 3, 4, 5, 6, 7, 8, 9, 12 and 13 are active.

COMMENTS: The problem is a mathematical programming model of a three-stage membrane separation process.

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #21

```
      FUNCTION F(X)
      DIMENSION X(1)
      F=X(11)+X(12)+X(13)
      RETURN
      END
```

```
      SUBROUTINE CONST(X,NPHI,PHI)
      DIMENSION X(1),PHI(1),C(36)
      DATA C/1.262626,-1.231059,1.262626,-1.231059,1.262626,-1.231059,
     1 .034750,.975,-.00975,.034750,.975,-.009750,1.0,1.0,-1.0,.002,
     2 .002,1.0,1.0,-.002,-.002,1.0,1.0,500.,-1.0,-500.,.9,.002,-.002,
     3 1.0,1.0,.002,-.002,.03475,.975,-.00975/
      PHI(1)=1.0-C(1)*X(8)/X(11)-C(2)*X(1)*X(8)/X(11)
      PHI(2)=1.0-C(3)*X(9)/X(12)-C(4)*X(2)*X(9)/X(12)
      PHI(3)=1.0-C(5)*X(10)/X(13)-C(6)*X(3)*X(10)/X(13)
      PHI(4)=1.0-C(7)*X(2)/X(5)-C(8)*X(2)-C(9)*X(2)*X(2)/X(5)
      PHI(5)=1.0-C(10)*X(3)/X(6)-C(11)*X(3)-C(12)*X(3)*X(3)/X(6)
      PHI(6)=1.0-C(13)*X(1)/X(5)/X(7)*X(8)-C(14)*X(4)/X(5)
     1 -C(15)*X(4)/X(5)/X(7)*X(8)
      PHI(7)=1.0-C(16)*X(2)*X(9)-C(17)*X(5)*X(8)-C(18)*X(6)
     1 -C(19)*X(5)-C(20)*X(1)*X(8)-C(21)*X(6)*X(9)
      PHI(8)=1.0-C(22)/X(2)*X(3)/X(9)*X(10)-C(23)*X(6)/X(2)
     1 -C(24)/X(9)-C(25)/X(9)*X(10)-C(26)/X(2)*X(6)/X(9)
      PHI(9)=1.0-C(27)/X(2)-C(28)*X(10)-C(29)/X(2)*X(3)*X(10)
      PHI(10)=1.0-C(30)*X(2)/X(3)
      PHI(11)=1.0-C(31)*X(1)/X(2)
      PHI(12)=1.0-C(32)*X(7)-C(33)*X(8)
      PHI(13)=1.0-C(34)*X(1)/X(4)-C(35)*X(1)-C(36)*X(1)*X(1)/X(4)
      RETURN
      END
```

(22). Test problem #7 used by Dembo [46].

GENERAL INFORMATION:

16 Variables

19 Functional inequality constraints

32 Variable bounds

$.1 \le x_1 \le .9$	$.1 \le x_9 \le .9$
$.1 \le x_2 \le .9$	$.1 \le x_{10} \le .9$
$.1 \le x_3 \le .9$	$1 \le x_{11} \le 1000$
$.1 \le x_4 \le .9$	$.000001 \le x_{12} \le 500$
$.9 \le x_5 \le 1$	$1 \le x_{13} \le 500$
$.0001 \le x_6 \le .1$	$500 \le x_{14} \le 1000$
$.1 \le x_7 \le .9$	$500 \le x_{15} \le 1000$
$.1 \le x_8 \le .9$	$.000001 \le x_{16} \le 500$

STARTING INFORMATION:

$\bar{x}_o$ = [.8, .83, .85, .87, .90, .10, .12, .19, .25, .29, 512, 13.1, 71.8, 640, 650, 5.7]

$f(\bar{x}_o) = 284.739749$

$g_1(\bar{x}_o) = .004400000$	$g_{11}(\bar{x}_o) = -.0340896550$
$g_2(\bar{x}_o) = .006368958$	$g_{12}(\bar{x}_o) = .0022000000$
$g_3(\bar{x}_o) = .052865132$	$g_{13}(\bar{x}_o) = .9744140620$
$g_4(\bar{x}_o) = .060339100$	$g_{14}(\bar{x}_o) = .0333333333$
$g_5(\bar{x}_o) = .041887931$	$g_{15}(\bar{x}_o) = .0229885700$
$g_6(\bar{x}_o) = .017415364$	$g_{16}(\bar{x}_o) = .0235294148$
$g_7(\bar{x}_o) = -.021515790$	$g_{17}(\bar{x}_o) = .0361445783$
$g_8(\bar{x}_o) = -.11609600$	$g_{18}(\bar{x}_o) = .1379310340$
$g_9(\bar{x}_o) = -.14338235$	$g_{19}(\bar{x}_o) = .2400000000$
$g_{10}(\bar{x}_o) = .15354377$	

(22). (cont'd)

SOLUTION:

$\bar{x}^* = [.80377316, .81611713, .9, .9, .9, .1, .10703686, .19083674, .19083674, .19083674, 505.04987, 5.0498694, 72.636801, 500, 500, .00001]$

$f(\bar{x}^*) = 174.786995$

Constraints #1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 18 and 19 are active

COMMENTS: The problem is a mathematical programming model of a five-stage membrane separation process.

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #22

```
      FUNCTION F(X)
      DIMENSION X(1)
      DATA A,B/1.262626,-1.231060/
      F=A*(X(12)+X(13)+X(14)+X(15)+X(16))+B*(X(1)*X(12)+
     1 X(2)*X(13)+X(3)*X(14)+X(4)*X(15)+X(5)*X(16))
      RETURN
      END
```

```
      SUBROUTINE CONST(X,NPHI,PHI)
      DIMENSION X(1),PHI(1),C(51)
      DATA C/.03475,.975,-.00975,.03475,.975,-.00975,.03475,
     1 .975,-.00975,.03475,.975,-.00975,.03475,.975,-.00975,1.0,
     2 1.0,-1.0,1.0,.002,.002,-.002,-.002,1.0,.002,.002,1.0,-.002,
     3 -.002,1.0,1.0,500.0,-500.0,-1.0,1.0,1.0,500.0,-1.0,-500.0,
     4 .9,.002,-.002,.002,-.002,1.0,1.0,1.0,1.0,1.0,1.0,1.0/
      PHI(1)=1.0-C(1)*X(1)/X(6)-C(2)*X(1)-C(3)*X(1)*X(1)/X(6)
      PHI(2)=1.0-C(4)*X(2)/X(7)-C(5)*X(2)-C(6)*X(2)*X(2)/X(7)
      PHI(3)=1.0-C(7)*X(3)/X(8)-C(8)*X(3)-C(9)*X(3)*X(3)/X(8)
      PHI(4)=1.0-C(10)*X(4)/X(9)-C(11)*X(4)-C(12)*X(4)*X(4)/X(9)
      PHI(5)=1.0-C(13)*X(5)/X(10)-C(14)*X(5)-C(15)*X(5)*X(5)/X(10)
      PHI(6)=1.0-C(16)*X(6)/X(7)-C(17)*X(1)/X(7)/X(11)*X(12)-C(18)*
     1 X(6)/X(7)/X(11)*X(12)
      PHI(7)=1.0-C(19)*X(7)/X(8)-C(20)*X(7)/X(8)*X(12)-C(21)*X(2)/
     1 X(8)*X(13)-C(22)*X(13)-C(23)*X(1)/X(8)*X(12)
      PHI(8)=1.0-C(24)*X(8)-C(25)*X(8)*X(13)-C(26)*X(3)*X(14)-C(27)*
     2 X(9)-C(28)*X(2)*X(13)-C(29)*X(9)*X(14)
      PHI(9)=1.0-C(30)*X(9)/X(3)-C(31)/X(3)*X(4)/X(14)*X(15)-C(32)/
     1 X(3)*X(10)/X(14)-C(33)/X(3)*X(9)/X(14)-C(34)/X(3)*X(8)/X(14)
     2 *X(15)
      PHI(10)=1.0-C(35)/X(4)*X(5)/X(15)*X(16)-C(36)/X(4)*X(10)-
     1 C(37)/X(15)-C(38)/X(15)*X(16)-C(39)/X(4)*X(10)/X(15)
      PHI(11)=1.0-C(40)/X(4)-C(41)*X(16)-C(42)/X(4)*X(5)*X(16)
      PHI(12)=1.0-C(43)*X(11)-C(44)*X(12)
      PHI(13)=1.0-C(45)/X(11)*X(12)
      PHI(14)=1.0-C(46)*X(4)/X(5)
      PHI(15)=1.0-C(47)*X(3)/X(4)
      PHI(16)=1.0-C(48)*X(2)/X(3)
      PHI(17)=1.0-C(49)*X(1)/X(2)
      PHI(18)=1.0-C(50)*X(9)/X(10)
      PHI(19)=1.0-C(51)*X(8)/X(9)
      RETURN
      END
```

(23). Test problem #8A used by Dembo [46].

GENERAL INFORMATION:

7 Variables

4 Functional constraints

14 Variable bounds $.1 \leq x_j \leq 10 \quad j=1,2,3,\ldots,6$

$.01 \leq x_7 \leq 10$

STARTING INFORMATION:

$x_{o_j} = 6 \quad j=1,2,3,\ldots,7$

$f(\bar{x}_o) = 2205.86837$

$g_1(\bar{x}_o) = -369.81882$ $g_3(\bar{x}_o) = -15.9306112$

$g_2(\bar{x}_o) = -4.3413695$ $g_4(\bar{x}_o) = -137.947340$

SOLUTION:

$\bar{x}^* = [2.8560239, .6108117965, 2.150810, 4.71196656, .99941464, 1.34732658, .0316508066]$

$f(\bar{x}^*) = 1809.764787$

Constraints #1, 2 and 3 are active.

COMMENTS: This is a prototype geometric programming problem.

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #23

```
      FUNCTION F(X)
      DIMENSION X(1)
      DATA C1,C2,C3,C4/10.,15.,20.,25./
      ALPHA=-.250
      F=C1*X(1)/X(2)*X(4)**2/X(6)**3*X(7)**ALPHA+C2/X(1)/
     1 X(2)**2*X(3)*X(4)/X(5)*X(7)**(-.50)+C3*X(1)**(-2)*X(2)/X(4)
     2 *X(5)**(-2)*X(6)+C4*X(1)**2*X(2)**2/X(3)*X(5)**.50/X(6)**2
     3 *X(7)
       RETURN
       END
```

```
      SUBROUTINE CONST(X,NPHI,PHI)
      DIMENSION X(1),PHI(1),C(14)
      DATA C/.5,.7,.2,1.3,.8,3.1,2.,.1,1.,.65,.2,.3,.4,.5/
      PHI(1)=1.0-C(1)*X(1)**.5/X(3)/X(6)**2*X(7)-C(2)*X(1)**3*X(2)/
     1 X(3)**2*X(6)*X(7)**.5-C(3)/X(2)*X(3)/X(4)**.5*X(6)**(2./3.)*
     2 X(7)**.25
      PHI(2)=1.0-C(4)/X(1)**.5*X(2)/X(3)/X(5)*X(6)-C(5)*X(3)/X(4)/
     1 X(5)*X(6)**2-C(6)/X(1)*X(2)**.5/X(4)**2/X(5)*X(6)**(1./3.)
      PHI(3)=1.0-C(7)*X(1)/X(3)**(1.5)*X(5)/X(6)*X(7)**(1./3.)-C(8)*
     1 X(2)/X(3)**.5*X(5)/X(6)/X(7)**.5-C(9)/X(1)*X(2)*X(3)**.5*X(5)
     2 -C(10)/X(2)**2*X(3)*X(5)/X(6)*X(7)
      PHI(4)=-C(11)/X(1)**2*X(2)/X(4)*X(5)**.5*X(7)**(1./3.)-C(12)*X(1)
     1 **.5*X(2)**2*X(3)*X(4)**(1./3.)/X(5)**(2./3.)*X(7)**.25-C(13)/
     2 X(1)**3/X(2)**2*X(3)*X(5)*X(7)**.75-C(14)/X(3)**2*X(4)*
     3 X(7)**.50
      PHI(4)=1.0+PHI(4)
      RETURN
      END
```

(24). The welded beam problem.

GENERAL INFORMATION:

4 Variables

5 Functional inequality constraints

3 Variable bounds $x_1 \geq .125$; $x_2 \geq 0$; $x_3 \geq 0$

STARTING INFORMATION:

$\bar{x}_o = [1, 7, 4, 2]$

$f(\bar{x}_o) = 15.81545$

$g_1(\bar{x}_o) = .867852506$

$g_2(\bar{x}_o) = 1.42500000$

$g_3(\bar{x}_o) = 1.00000000$

$g_4(\bar{x}_o) = 183.187852$

$g_5(\bar{x}_o) = .232850000$

SOLUTION:

$\bar{x}^* = [.24436897, 6.2187934158, 8.29147139, .24436897]$

$f(\bar{x}^*) = 2.38116476461$

Constraints #1, 2, 3 and 4 are active.

COMMENTS: This problem involves the design of a welded beam structure to produce the minimum cost. The design variables include the weld thickness, the weld length, the bar thickness and the breadth of the bar. A complete description may be found in reference [66].

FUNCTION AND CONSTRAINT LISTING FOR PROBLEM #24

```
FUNCTION F(X)
DIMENSION X(1)
F=1.10471*X(1)*X(1)*X(2)+.04811*X(3)*X(4)*(14.+X(2))
RETURN
END
```

```
SUBROUTINE CONST(X,NPHI,PHI)
DIMENSION X(1),PHI(1)
REAL L,LOAD
REAL M,J
L=14. $ LOAD=6000. $ TD=13600. $ SIGD=30000.
F=LOAD
T1=F/(1.414*X(1)*X(2))
M=F*(L+(X(2)/2.))
R=SQRT((X(2)*X(2)/4.)+((X(3)+X(1))/2.)**2)
J=2.*(.707*X(1)*X(2)*((X(2)*X(2)/12.)+((X(3)+X(1))/2.)**2))
T2=M*R/J
COSA=X(2)/(2.*R)
T=SQRT(T1*T1+2.*T1*T2*COSA+T2*T2)
SIG=6.*F*L/(X(4)*X(3)*X(3))
PHI(1)=(TD-T)/10000.
PHI(2)=(SIGD-SIG)/10000.
PHI(3)=X(4)-X(1)
E=30.E6
EI=E*X(3)*X(4)*X(4)*X(4)/12.
G=12.E6
GJ=G*X(3)*X(4)*X(4)*X(4)/3.
EITC=EI*GJ
EIDC=EI/GJ
REITC=SQRT(EITC)
REIDC=SQRT(EIDC)
PC=4.013*REITC*(1.-(X(3)/(2.*L))*REIDC)/(L*L)
PHI(4)=(PC-F)/10000.
DEL=4.*F*L*L*L/(E*X(4)*X(3)*X(3)*X(3))
PHI(5)=.25-DEL
RETURN
END
```

(25). Coupler curve problem.

GENERAL INFORMATION:

6 Variables

4 Functional inequality constraints

6 Variable bounds $.5 \le x_1 \le 3$

$x_2 \ge 0$; $x_3 \ge 0$

$2 \le x_4 \le 10$

STARTING INFORMATION:

$\bar{x}_o^* = [1, 4.5, 4, 5, 3, 3]$

$f(x_o) = 2.3088037$

$g_1(\bar{x}_o) = 2.5$ $g_3(\bar{x}_o) = 5.20584413$

$g_2(\bar{x}_o) = 3.5$ $g_4(\bar{x}_o) = 25.7058441$

SOLUTION:

$\bar{x}^* = [.996387714, 4.19429134, 2.97449996, 3.962212286, 1.65300, 1.2551]$

$f(\bar{x})^* = .06060082755$

Constraint #3 is active.

COMMENTS: This problem involves the design of a four bar linkage to approximate a required curve. The objective function represents the mean square error of the actual points generated to the desired points on the curve. A complete description of the problem is given by Tomás [67].

AUXILIARY SUBROUTINE AND FUNCTION LISTING FOR PROBLEM #25

```
      SUBROUTINE DES(PHI,X1,Y1)
      DIMENSION PHI(31),X1(31),Y1(31)
      PI=3.141592654
      DO 10 I=1,31
      X1(I)=.40+SIN((2.0*PI)*((PI-PHI(I))/(2.0*PI)-.16))
   10 Y1(I)=2.0+.90*SIN(PI-PHI(I))
      RETURN
      END
```

```
      FUNCTION F(X)
      DIMENSION X(1),X1A(31),Y1A(31)
      DIMENSION PHI(31),X1(31),Y1(31)
      DATA IA/0/
      IF(IA.EQ.1) GO TO 2
      IA=1
      PI=3.14159
      XINC=(2.0*PI)/30.0
      DO 1 I=1,31
      PHI(I)=XINC*FLOAT(I-1)
    1 CALL DES(PHI,X1,Y1)
    2 F=0.0
      DO 10 I=1,31
      CALL POS(X,PHI(I),COSS)
      SINS=SQRT(ABS(1.0-COSS*COSS))
      COSG=(X(4)+X(3)*COSS-X(1)*COS(PHI(I)))/X(2)
      SING=(X(3)*SINS-X(1)*SIN(PHI(I)))/X(2)
      X1A(I)=X(1)*COS(PHI(I))+X(5)*COSG-X(6)*SING
      Y1A(I)=X(1)*SIN(PHI(I))+X(5)*SING+X(6)*COSG
   10 F=F+(X1A(I)-X1(I))**2+(Y1A(I)-Y1(I))**2
      F=SQRT(F/31.0)
      RETURN
      END
```

CONSTRAINT AND AUXILIARY SUBROUTINE LISTING FOR PROBLEM #25

```
      SUBROUTINE CONST(X,NCONS,CON)
      DIMENSION X(1),CON(1)
      COMMON/A/ XMU1,XMU2
      DATA XMU1,XMU2/.7853981633,2.356194491/
      CON(1)=-X(1)+X(2)+X(3)-X(4)
      CON(2)=-X(1)-X(2)+X(3)+X(4)
      CON(3)=-X(2)*X(2)-X(3)*X(3)+(X(4)-X(1))*(X(4)-X(1))
     1 +2.0*X(2)*X(3)*COS(XMU1)
      CON(4)=X(2)*X(2)+X(3)*X(3)-(X(4)+X(1))*(X(4)+X(1))
     1 -2.0*X(2)*X(3)*COS(XMU2)
      RETURN
      END
```

```
      SUBROUTINE POS(X,PHI,W)
      DIMENSION X(1)
      REAL K,L,M
      PI=3.141592654
      M=2.0*X(1)*X(3)*SIN(PHI)
      L=2.0*X(3)*X(4)-2.0*X(1)*X(3)*COS(PHI)
      K=X(1)*X(1)-X(2)*X(2)+X(3)*X(3)+X(4)*X(4)-2.0*X(4)*X(1)*COS(PH
      A=L*L+M*M
      B=2.0*K*L
      C=K*K-M*M
      TERM=SQRT(ABS(B*B-4.0*A*C))
      IF(PI-PHI.LT.0.0) TERM=-TERM
      W=(-B+TERM)/(2.0*A)
      RETURN
      END
```

(26). Whirlpool design problem [68].

GENERAL INFORMATION:

3 Variables

1 Functional equality constraint

4 Variable bounds $x_1 \leq .044$; $13.13 \leq x_2 \leq 24$

$x_3 \leq 600$

STARTING INFORMATION:

$\bar{x}_o = [.1, 18, 144]$

$f(\bar{x}_o) = 30.9860722$

$h_1(\bar{x}_o) = -211.4309$

SOLUTION:

$\bar{x}^* = [.122063682, 24, 108.5052434]$

$f(\bar{x}^*) = 27.305651561$

FUNCTION LISTING FOR PROBLEM #26

```
 FUNCTION F(X)
 DIMENSION X(1)
 DATA RHO,XMU,CP,PR,PI,D,TIN,TSURF,H,W,RHOC,RHOA/.0747,
1.0443,.240,.709,3.14159,.525,75.0,45.0,13.13,3.166,559.,169./
 AF=X(2)/X(1)*2.0*(W*H-30.0*PI*D**2/4.)/144.0
 AT=30.0*PI*D*X(2)/144.0
 AC=(H*X(2)-10.0*D*X(2)-X(2)/X(1)*.006*H)/144.0
 G=(RHO*X(3)*(H*X(2))/(AC*144.0))*60.0
 RE=G*1.083/(12.0*XMU)
 IF(RE.LT.1.0E-10) RE=1.0E-10
 HO=(.195*G*CP)/(PR**.67*RE**.35)
 XMDOT=RHO*X(3)*H*X(2)/144.0*60.0
 DELP=1.833E-06/RHO*G**2*3.0*(AF/AC*RE**(-.5)+.1*AT/AC)
 IF(HO.LT.1.0E-10) HO=1.0E-10
 XVAL=.0732*SQRT(HO)
 ETAF=TANH(XVAL)/XVAL
 ETAS=1.0-AF/(AF+AT)*(1.0-ETAF)
 HEF=1.0-EXP(-ETAS*HO*(AF+AT)/(XMDOT*CP))
 Q=HEF*(TIN-TSURF)*XMDOT*CP
 H1=DELP/RHO*XMDOT/1.98E+06
 IF(H1.LT.1.0E-10) H1=1.0E-10
 COSTM=SQRT(H1)/.0718+4.0
 COSTT=1.01*30.0*X(2)*PI/4.0*(D**2-(D-.036)**2)
 COSTF=.47*H*W*.006*RHOA/1728.*X(2)/X(1)
 COSTT=COSTT*RHOC/1728.
 F=COSTM+COSTT+COSTF
 RETURN
 END
```

CONSTRAINT LISTING FOR PROBLEM #26

```
      SUBROUTINE EQUAL(X,PSI,NPSI)
      DIMENSION X(1),PSI(1)
      DATA RHO,XMU,CP,PR,PI,D,TIN,TSURF,H,W,RHOC,RHOA/.0747,
     1.0443,.240,.709,3.14159,.525,75.0,45.0,13.13,3.166,559.,169./
      AF=X(2)/X(1)*2.0*(W*H-30.0*PI*D**2/4.)/144.0
      AT=30.0*PI*D*X(2)/144.0
      AC=(H*X(2)-10.0*D*X(2)-X(2)/X(1)*.006*H)/144.0
      G=(RHO*X(3)*(H*X(2))/(AC*144.0))*60.0
      RE=G*1.083/(12.0*XMU)
      IF(RE.LT.1.0E-10) RE=1.0E-10
      HO=(.195*G*CP)/(PR**.67*RE**.35)
      XMDOT=RHO*X(3)*H*X(2)/144.0*60.0
      DELP=1.833E-06/RHO*G**2*3.0*(AF/AC*RE**(-.5)+.1*AT/AC)
      IF(HO.LT.1.0E-10) HO=1.0E-10
      XVAL=.0732*SQRT(HO)
      ETAF=TANH(XVAL)/XVAL
      ETAS=1.0-AF/(AF+AT)*(1.0-ETAF)
      HEF=1.0-EXP(-ETAS*HO*(AF+AT)/(XMDOT*CP))
      Q=HEF*(TIN-TSURF)*XMDOT*CP
      PSI(1)=6000.0-Q
      RETURN
      END
```

(27). Synthetic natural gas production problem [54].

GENERAL INFORMATION:

48 Variables

2 Equality constraints

1 Functional inequality constraint

72 Variable bounds $x_j \geq .002$ $j=1,2,3,\ldots,48$

$x_j \leq 2.0$ $j=1,2,3,\ldots,24$

STARTING INFORMATION:

$x_{o_j} = 1.0$ $j=1,2,3,\ldots,24$

$x_{o_j} = 1.3$ $j=25,26,\ldots,30$

$x_{o_j} = 1.0$ $j=31,32,\ldots,48$

$f(x_o) = 1.8623009$

$g_1(x_o) = .0319258906$

$h_1(x_o) = 0$

$h_2(x_o) = 0$

SOLUTION:

$\overline{x}^* =$ [2, .002, 2, .0339797, .01657455, 2, 1.8945347, .002, 2, .03424074, .016670308, 2, 2, .002, 2, .002, .002, 1.988000, 2, .002, 2, .002, .002, 2, 1.0159886, .002, 1.003163, .002, .002, .999691944, 1.11272844, .002, 1.1024463, .002, .002, 1.1030764, .92326572, .9343325, .92947437, .91383802, .90517162, .89452569, 1.174573, .002, 1.12080408, .002, .002, 1.1163321536]

Constraint is active at solution.

FUNCTION AND EQUALITY CONSTRAINT
LISTINGS FOR PROBLEM #27

```
      FUNCTION F(X)
      DIMENSION X(1)
      PEN(Z)=(.1+2.*Z*(Z+SQRT(.1+Z*Z)))/4.
      E=0.
      DO 100 I=1,12
      C=1.-X(I)
  100 E=E+10.*C*C
      DO 120 I=25,36
  120 E=E+1000.*PEN(X(I)-1.)
      DO 140 I=37,42
  140 E=E+2000.*PEN(X(I)-1.)
      DO 160 I=43,48
  160 E=E+100.*X(I)
      F=E/1000.
      RETURN
      END
```

```
      SUBROUTINE EQUAL(X,PSI,NPSI)
      DIMENSION X(1),PSI(1)
      PSI(1)=12.
      PSI(2)=12.
      DO 170 I=1,12
      PSI(1)=PSI(1)-X(I)
  170 PSI(2)=PSI(2)-X(I+12)
      RETURN
      END
```

INEQUALITY CONSTRAINT LISTING FOR PROBLEM #27

```
      SUBROUTINE CONST(X,NPHI,PHI)
      DIMENSION A(18),U(18),X(1),PHI(1)
      DATA (A(I),I=1,18)/.9,.8,1.1,1.,.7,1.1,1.,1.,1.1,
     1 .9,.8,1.2,.9,1.2,1.2,1.,1.,.9/
C     1ST TIER OF GASFIERS
      DO 20 I=1,6
      K1=I+24
      K2=I+42
      K3=I+12
      ALP=X(K1)*X(K1)*A(I)*2.*X(K2)/(1.+X(K2))*X(K3)
   20 U(I)=X(I)*X(I)/(X(I)+ALP)
C     2ND TIER OF GASFIERS
      DO 40 I=7,12
      K1=I+24
      K2=I+36
      K3=I+12
      ALP=X(K1)*X(K1)*A(I)*2.*X(K2)/(1.+X(K2))*X(K3)
      SUM=X(I)+U(I-6)
   40 U(I)=SUM*SUM/(SUM+ALP)
C     1ST TIER OF METHANATORS
      DO 60 I=13,15
      K1=2*(I-10)+1
      K2=I+24
      ALP=X(K2)*X(K2)*A(I)
      SUM=U(K1)+U(K1+1)
   60 U(I)=SUM*SUM/(SUM+ALP)
C     2ND TIER OF METHANATORS
      DO 80 I=16,18
      K1=I+24
      ALP=X(K1)*X(K1)*A(I)
      SUM=U(I-3)
   80 U(I)=SUM*SUM/(SUM+ALP)
      R=U(16)+U(17)+U(18)
      PHI(1)=1.5-R
      RETURN
      END
```

(28). Flywheel optimization problem.

GENERAL INFORMATION:

5 Variables

3 Functional inequality constraints

10 Variable bounds $x_j \le 10 \quad j=1,2,3,4,5$

$x_j \ge -10 \quad j=1,2,3$

$x_4 \ge .10$

$x_5 \ge 1.0$

STARTING INFORMATION:

$\bar{x}_o = [-.3359769, -1.432398, 0, 4, 9]$

$f(\bar{x}_o) = -1.631484$

$g_1(\bar{x}_o) = 16.40753$

$g_2(\bar{x}_o) = 1$

$g_3(\bar{x}_o) = 28.03172$

SOLUTION:

$\bar{x}^* = [.19852438, -3.01059794, -.0530266138,$
$2.83165094, 10]$

$f(\bar{x}^*) = -5.55840576$

Constraints #1, 2 and 3 are active.

COMMENTS: This problem involves the design of a flywheel to produce the maximum kinetic energy for a given amount of material rotating at a specified speed. The flywheel is to fit on a one inch shaft and the design variables specify the flywheel contour. The flywheel contour is generated using Fourier coefficients to allow the flywheel to take any arbitrary shape with a limitation imposed by the small number of terms used. The constraints include a constraint on the maximum allowable stress in the flywheel which involves the solution of a second order nonlinear boundary value problem. The evaluation of this constraint

(28). (cont'd)

alone required almost 3/4 second. The other constraints placed limits on the maximum thickness of the flywheel and the maximum value of the outer radius of the flywheel.

FUNCTION AND AUXILIARY SUBROUTINE FOR PROBLEM #28

```
      FUNCTION F(X)
      DIMENSION X(1),XC(100),THICK(100),DTHICK(100)
      COMMON/B/ XMU,RHO,THICK,W2,DTHICK,KKK
      COMMON/ST/ VALUE2
      DATA XMU,ALPHA,W,EPSI,RHO,YI,XI,II,KKK,EPS/.30,
     1 1000.0,628.0,.0001,7.263E-4,0.0,1.0,99,98,.10/
      W2=X(6)*100.0
      DR=X(5)-XI
      XC(1)=XI
      DO 60 I=1,KKK
   60 XC(I+1)=XC(I)+DR/FLOAT(KKK)
      CALL THFN(XC,THICK,DTHICK,X,KKK,X(11),XI)
      CALL ENERGY(XC,F)
      F=-F/1.0E+06
      RETURN
      END
```

```
      SUBROUTINE ENERGY(X,XKE)
      DIMENSION X(100),THICK(100),DTHICK(100)
      COMMON/B/ XMU,RHO,THICK,W,DTHICK,KKK
      CONST=3.1415927*RHO*W**2
      XKE=0.0
      DO 10 I=1,KKK
   10 XKE=XKE+X(I)**3*THICK(I)*(X(I+1)-X(I))
      XKE=XKE*CONST
      RETURN
      END
```

CONSTRAINT LISTING FOR PROBLEM #28

```
      SUBROUTINE CONST(C,NPHI,PHI)
      DIMENSION Y1(100),Y(100),X(100),RST(100),TST(100),THICK(100)
      DIMENSION  DTHICK(100),STOT(100),C(10),PHI(4)
      COMMON/B/  XMU,RHO,THICK,W2,DTHICK,KKK
      COMMON/TFN1/ TMAX
      COMMON/ST/ VALUE2
      DATA XMU,ALPHA,W,EPSI,RHO,YI,XI,II,KKK,EPS/.30,
     1 1000.0,628.0,.0001,7.263E-4,0.0,1.0,99,98,.10/
      ISTRT=0
      W2=C(6)*100.0
      DR=C(5)-XI
      X(1)=XI
      DO 60 I=1,KKK
   60 X(I+1)=X(I)+DR/KKK
      CALL THFN(X,THICK,DTHICK,C,KKK,C(4),XI)
    5 Y1I=100000.00
    2 CALL RUNKUT(Y1I,YI,XI,C(5),II,Y,Y1,X)
      FXL=-Y(II)
      XL=Y1I
      YI=0.0
      Y1I=2503000.00
      CALL RUNKUT(Y1I,YI,XI,C(5),II,Y,Y1,X)
      FXR=-Y(II)
      XR=Y1I
      CALL FALSE(XL,XR,FXL,FXR,EPS,Y,Y1,X,XI,C(5),YI,ROOT,II)
      SMAX=0.0
      VALUE2=0.0
      DO 300 NN=1,II
      RST(NN)=Y(NN)/(THICK(NN)*X(NN))
      TST(NN)=(Y1(NN)+THICK(NN)*RHO*W2**2*X(NN)**2)/THICK(NN)
      STOT(NN)=SQRT((RST(NN)-TST(NN))**2+RST(NN)**2+TST(NN)**2)
      IF(STOT(NN).GT.SMAX) SMAX=STOT(NN)
      VALUE2=VALUE2+(30000.0-STOT(NN))**2
  300 CONTINUE
      VALUE2=SQRT(VALUE2)
      IF(ISTRT.EQ.2) WRITE(6,310)
  310 FORMAT(1H1, * RADIAL STRESS*,2X, *TANGENTIAL STRESS*,2X,
     1 * EQUIVALENT STRESS*,1X,*  RADIUS *,4X, * THICKNESS*)
      IF(ISTRT.EQ.2) WRITE(6,320)(RST(I),TST(I),STOT(I),X(I),
     1 THICK(I),I=1,II)
  320 FORMAT(1H ,5F15.2)
      PHI(1)=(30000.0-SMAX)/1000.0
      PHI(2)=5.0-TMAX
      CALL VOLUME(C(4),C,V,THICK,KKK,X)
      PHI(3)=(625.0-V)/10.
  900 RETURN
      END
```

AUXILIARY SUBROUTINE FOR PROBLEM #28

```
      SUBROUTINE RUNKUT(Y1I,YI,XI,XF,II,Y,Y1,X)
      DIMENSION Y1(100),Y(100),X(100)
      REAL M0,M1,M2,M3
      X(1)=XI
      Y(1)=YI
      Y1(1)=Y1I
      H=(XF-XI)/(II-1)
      KK=II-1
      DO 10 J=1,KK
      LL=J
      XR=X(J)
      YR=Y(J)
      Y1R=Y1(J)
      M0=H*YRUN(XR,YR,Y1R,LL)
      XR=X(J)+H/2.0
      YR=Y(J)+H*Y1(J)/2.0
      Y1R=Y1(J)+M0/2.0
      M1=H*YRUN(XR,YR,Y1R,LL)
      YR=YR+H*M0/4.0
      Y1R=Y1(J)+M1/2.0
      M2=H*YRUN(XR,YR,Y1R,LL)
      XR=X(J)+H
      YR=Y(J)+H*Y1(J)+H*M1/2.0
      Y1R=Y1(J)+M2
      M3=H*YRUN(XR,YR,Y1R,LL)
      Y(J+1)=Y(J)+H*Y1(J)+H/6.0*(M0+M1+M2)
      Y1(J+1)=Y1(J)+(M0+2.0*M1+2.0*M2+M3)/6.0
   10 X(J+1)=X(J)+H
      RETURN
      END
```

AUXILIARY SUBROUTINES FOR PROBLEM #28

```
      SUBROUTINE FALSE(XL,XR,FXL,FXR,EPS,Y,Y1,X,XI,XF,YI,ROOT,II)
      DIMENSION Y1(100),Y(100),X(100)
  105 XAPP=XL+(FXL*(XR-XL)/(FXL-FXR))
      CALL RUNKUT(XAPP,YI,XI,XF,II,Y,Y1,X)
      FXAPP=-Y(II)
      IF(ABS((XAPP-XSAVE)/XAPP).LE.EPS) GO TO 250
      VALUE=FXAPP*FXL
      IF(VALUE.LT.0) GO TO 110
      XL=XAPP
      XSAVE=XL
      FXL=FXAPP
      GO TO 105
  110 XR=XAPP
      XSAVE=XR
      FXR=FXAPP
      GO TO 105
  250 ROOT=XAPP
      RETURN
      END
```

```
      FUNCTION YRUN(XR,YR,Y1R,I)
      DIMENSION THICK(100),DTHICK(100)
      COMMON/B/  XMU,RHO,THICK,W2,DTHICK,KKK
      YRUN=(1.0/THICK(I)*DTHICK(I)-1.0/XR)*Y1R+(1.0/(XR**2)-XMU/(XR*THI
     *K(I))*DTHICK(I))*YR-(3.0+XMU)*RHO*W2**2*THICK(I)*XR
      RETURN
      END
```

AUXILIARY SUBROUTINES FOR PROBLEM #28

```
      SUBROUTINE THFN(X,THICK,DTHICK,C,KKK,CO,XI)
      DIMENSION   X(100),THICK(100),DTHICK(100),C(10)
      COMMON/TFN1/ TMAX
      THICK(1)=CO
      XL=X(KKK+1)-X(1)
      NFST=2
      TMAX=THICK(1)
      DO 10 I=1,KKK
      THICK(I+1)=CO+C(1)*(X(I+1)-XI)
      DO 9 LM=1,NFST
      JKL=LM+1
    9 THICK(I+1)=THICK(I+1)+C(JKL)*SIN((2.0*JKL-3.0)*3.14159*(X(I)-X(1))
     1 /XL)
      IF(THICK(I+1).GT.TMAX) TMAX=THICK(I+1)
   10 DTHICK(I)=(THICK(I+1)-THICK(I))/(X(I+1)-X(I))
      RETURN
      END
```

```
      SUBROUTINE VOLUME(CO,C,V,THICK,KKK,X)
      DIMENSION X(100),THICK(100),C(10)
      V=0.0
      PI=3.14159
      DELTX=(X(KKK+1)-X(1))/KKK
      LMN=KKK-1
      DO 10 I=1,LMN,2
      R1=(X(I+1)+X(I))/2.0
      R2=(X(I+1)+X(I+2))/2.0
      R3=(R1+R2)/2.0
   10 V=V+2.0*PI*DELTX/3.0*(THICK(I)*R1+4.0*THICK(I+1)*R3+THICK(I+2)*R2)
      RETURN
      END
```

(29). Optimization of a multi-spindle automatic lathe [69].

GENERAL INFORMATION:

10 Variables

1 Equality constraint

14 Functional inequality constraints

20 Variable bounds

$0 \le x_1 \le 10$	$.00005 \le x_6 \le .0013$
$0 \le x_2 \le .1$	$.00005 \le x_7 \le .0027$
$.00005 \le x_3 \le .0081$	$.00005 \le x_8 \le .002$
$10 \le x_4 \le 1000$	$.00005 \le x_9 \le 1$
$.00005 \le x_5 \le .0017$	$.00005 \le x_{10} \le 1$

STARTING INFORMATION:

$\bar{x}_o$ = [10, .005, .0081, 100, .0017, .0013, .0027, .002, .15, .105]

$f(\bar{x}_o) = 2931.46961755$

$h_1(\bar{x}_o) = -1.77636 \times 10^{-15}$	$g_8(\bar{x}_o) = 49.99566$
$g_1(\bar{x}_o) = 9.074074$	$g_9(\bar{x}_o) = 49.99584$
$g_2(\bar{x}_o) = 9.117647$	$g_{10}(\bar{x}_o) = 49.97842$
$g_3(\bar{x}_o) = 9.092308$	$g_{11}(\bar{x}_o) = 49.97748$
$g_4(\bar{x}_o) = 9.196296$	$g_{12}(\bar{x}_o) = 49.98758$
$g_5(\bar{x}_o) = 9.375000$	$g_{13}(\bar{x}_o) = 49.98960$
$g_6(\bar{x}_o) = 49.99737$	$g_{14}(\bar{x}_o) = 25.19410$
$g_7(\bar{x}_o) = 49.99737$	

(29). (cont'd)

SOLUTION:

$\bar{x}^* = [.209162445, .000614223, .0081, 442.684799, .0017, .0013, .0027, .001457317, .155236846, .0997631534]$

$f(\bar{x}^*) = -1614.9381624$

Constraints #1, 9, and 12 are active.

COMMENTS: This problem concerns maximizing the profit rate for producing a component on an automatic spindle lathe.

FUNCTION AND CONSTRAINT LISTINGS FOR PROBLEM #29

```
      FUNCTION F(X)
      DIMENSION X(1)
      F=20.0E+03
     1 *(15.0E-2*X(1)+14.0E+0*X(2)-6000.0E-5)
     2 /(2.0E-3+X(1)+60.0E+0*X(2))
      RETURN
      END
```

```
      SUBROUTINE EQUAL(X,PSI,NPSI)
      DIMENSION X(1),PSI(1)
      PSI(1)=X(9)+X(10)-255.0E-3
      RETURN
      END
```

```
      SUBROUTINE CONST(X,NPHI,PHI)
      DIMENSION X(1),PHI(1)
      PHI(1)=X(1)-75.0E-2/X(3)/X(4)
      PHI(2)=X(1)-X(9)/X(5)/X(4)
      PHI(3)=X(1)-X(10)/X(6)/X(4)-10.0E+0/X(4)
      PHI(4)=X(1)-19.0E-2/X(7)/X(4)-10.0E+0/X(4)
      PHI(5)=X(1)-125.0E-3/X(8)/X(4)
      PHI(6)=1.0E+4*X(2)-131.0E-5*X(9)*X(5)**666.0E-3*X(4)**15.0E-1
      PHI(7)=1.0E+4*X(2)-1038.0E-6*X(10)*X(6)**160.0E-2*X(4)**3.0E+0
      PHI(8)=1.0E+4*X(2)-223.0E-6*X(7)**666.0E-3*X(4)**15.0E-1
      PHI(9)=1.0E+4*X(2)-76.0E-6*X(8)**355.0E-2*X(4)**566.0E-2
      PHI(10)=1.0E+4*X(2)-698.0E-6*X(3)**120.0E-2*X(4)**2.0E+0
      PHI(11)=1.0E+4*X(2)-50.0E-6*X(3)**160.0E-2*X(4)**3.0E+0
      PHI(12)=1.0E+4*X(2)-654.0E-8*X(3)**242.0E-2*X(4)**417.0E-2
      PHI(13)=1.0E+4*X(2)-257.0E-6*X(3)**666.0E-3*X(4)**15.0E-1
      PHI(14)=30.0E+0-2003.0E-3*X(5)*X(4)-1885.0E-3*X(6)*X(4)
     1 -184.0E-3*X(8)*X(4)-2.0E+0*X(3)**803.0E-3*X(4)
      RETURN
      END
```

(30). Waste water treatment problem.

GENERAL INFORMATION:

19 Variables

11 Equality constraints

1 Functional inequality constraint

38 Variable bounds $x_j \geq .00001 \quad j = 1,2,\ldots,19$

$x_j \leq 50 \quad j = 1,2,16,17; \quad x_j \leq 100 \quad j = 3,4,5,6$

$x_j \leq 1.0 \times 10^5 \quad j = 7,8,9,10,11,18,19;$

$x_j \leq 1 \quad j = 12,13,14,15;$

STARTING INFORMATION:

$\bar{x}_o = [2, 4, 100, 50, 5, 20, 20, 3000, 3000, 2000, 7000, .001, .3, .5, .001, 5, 1, 9000, .5]$

$f(\bar{x}_o) = 61.9274433203$

$g_1(\bar{x}_o) = .2000000000$	$h_6(\bar{x}_o) = -75.42857143$
$h_1(\bar{x}_o) = +11.8622220$	$h_7(\bar{x}_o) = +42.5281479$
$h_2(\bar{x}_o) = +5.05888904$	$h_8(\bar{x}_o) = -3211.294074$
$h_3(\bar{x}_o) = -43.52963014$	$h_9(\bar{x}_o) = +1729.00000$
$h_4(\bar{x}_o) = +1641.76482$	$h_{10}(\bar{x}_o) = -6.18698064$
$h_5(\bar{x}_o) = +42.8571428$	$h_{11}(\bar{x}_o) = +.104000000$

SOLUTION: (Vicinity)

$\bar{x}^* = [.004473667, 3.441565, 99.34824, 89.130035, 15.279316, 15.279316, 94.726127, 12304.197, 12313.263, 12313.263, 95905.631, .00001, .00001, .9999890, .00001, .00001, .1622235, 8305.1515, .0014797356]$

(30). (Cont'd)

$f(\overline{x}^*) = 24.4724654$

Constraint #1 is active.

COMMENTS: This problem involves the design of a waste water treatment plant for minimum cost. A complete description of the problem is given by Himmelblau [70]. The problem contains many local minima, the one listed here is the best one found from the reported starting point.

FUNCTION AND INEQUALITY CONSTRAINT LISTING FOR PROBLEM #30

```
      FUNCTION F(X)
      DIMENSION X(1)
      ZI1=25.0*(2268.0*X(16)*X(1))**0.827
      ZI2=1.75E+05*X(17)+3.65E+04*X(17)**.182
      ZI3=12.6*X(18)+5.35*10.**3.378/X(18)**.126
      F=1.4*(ZI1+ZI2+ZI3+1.095E+04+1.15E+03*(X(1)*(X(13)-X(14))
     1 +X(2)*(1.0+X(12))-3.0*(1.0-X(19))))
      F=F/1.0E+04
      DO 33 I=1,11
      F=F+1.0E+03*X(19+I)
   33 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CONST(X,NPHI,PHI)
      DIMENSION X(1),PHI(1)
      COMMON/ANS/ NF,NI,NE
      NI=NI+1
      PHI(1)=1.0-X(13)-X(14)
      RETURN
      END
```

EQUALITY CONSTRAINT LISTING FOR PROBLEM #30

```
      SUBROUTINE EQUAL(X,PSI,NPSI)
      DIMENSION X(1),PSI(11)
      AK=.0259*25.0/20.0**.656
      XZ4=X(3)*EXP(-AK*X(16))
      ZJ1=-(X(1)*X(13)*XZ4+300.0*X(19))
      PSI(1)=ZJ1+X(1)*X(3)-X(2)*X(5)*X(12)
      YZ4=X(7)+.50*(X(3)-XZ4)
      ZJ2=-X(13)*X(1)*YZ4
      PSI(2)=ZJ2+X(1)*X(7)-X(2)*X(9)*X(12)
      ZJ3=-300.0*(1.0-X(19))+3.0*X(6)*(1.0-X(19))-X(1)*X(14)*XZ4
      PSI(3)=ZJ3+X(2)*(X(4)-X(6))+X(1)*X(6)*X(14)
      ZJ4=3.0*X(11)*(1.0-X(19))+X(1)*X(14)*(X(11)-YZ4)
      PSI(4)=ZJ4+X(2)*(X(8)-X(11))
      ZJ5=X(17)*(.48*X(5)*X(9)/(100.0+X(5)))
      PSI(5)=-2.0*ZJ5+X(2)*(X(4)-X(5))
      PSI(6)=ZJ5+X(2)*(X(8)-X(9))-.048*X(9)*X(17)
      ZK7=X(1)*(1.0-X(13)-X(14))
      QZ12=X(1)*(1.0-X(13)-X(14))+X(2)*(1.0-X(12))
      PSI(7)=-ZK7*XZ4+X(6)*QZ12-X(2)*X(5)*(1.0-X(12))
      ZJ8=X(10)*QZ12-ZK7*YZ4
      PSI(8)=ZJ8-X(2)*X(9)*(1.0-X(12))
      PSI(9)=6.0*(1.0-X(15))*(20.0-X(6))+X(11)*(X(2)-3.0*(1.0-X(15)
     1 )-X(1)*X(14))+3.0*X(19)*X(11)-X(10)*QZ12
      CK=7.4*2.0*1.2**4/2.31E+04
      TEST=-CK*X(18)/QZ12
      IF(TEST.GT.99) ZJ10=-2.1*SQRT(ABS(X(10)))*EXP(99.0)
      IF(TEST.LT.99) ZJ10=-2.1*SQRT(ABS(X(10)))*EXP(-CK*X(18)/QZ12)
      PSI(10)=ZJ10+2.0*(20.0-X(6))
      PSI(11)=(1.0-X(13))*X(1)-X(12)*X(2)-3.0*X(19)
      RETURN
      END
```

Appendix C

Intermediate Data

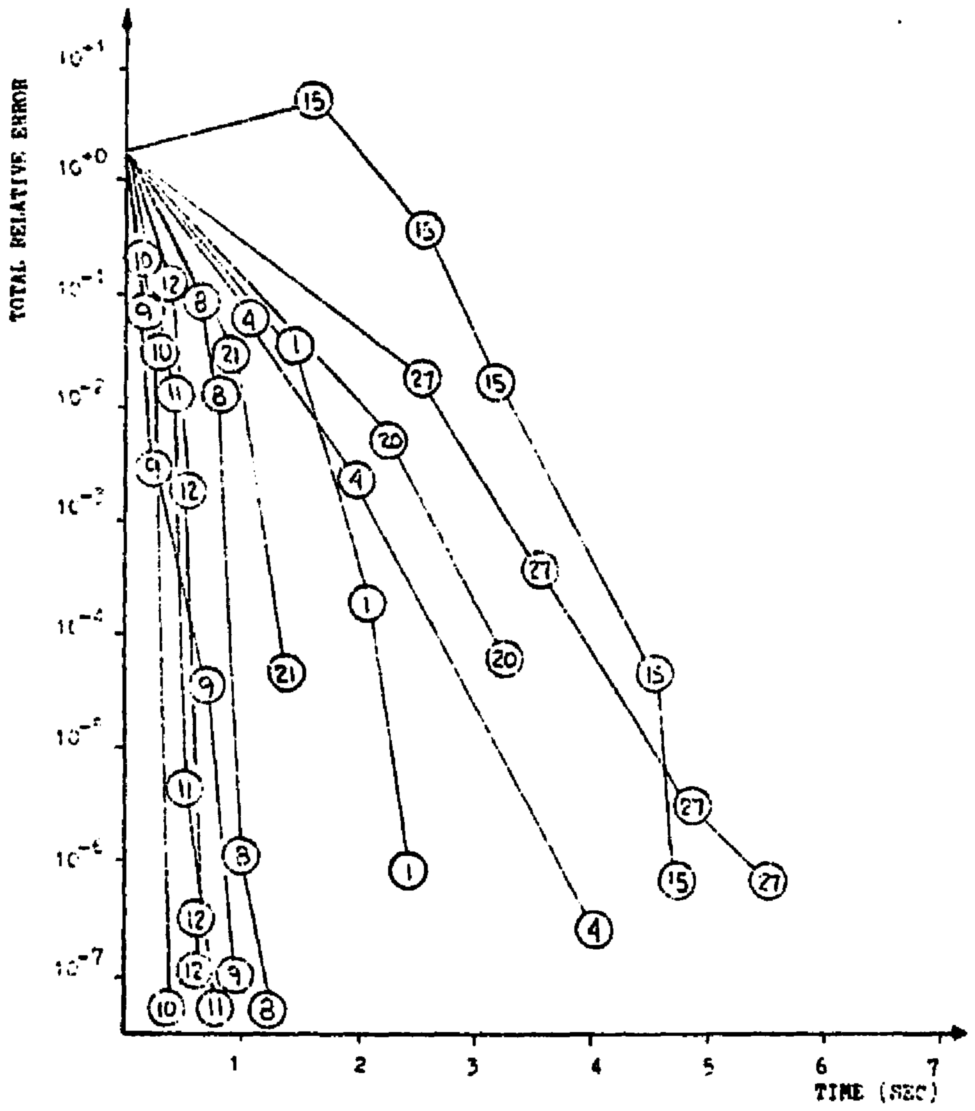

Figure C.1 Total Relative Error versus Time for Problem Number 1.

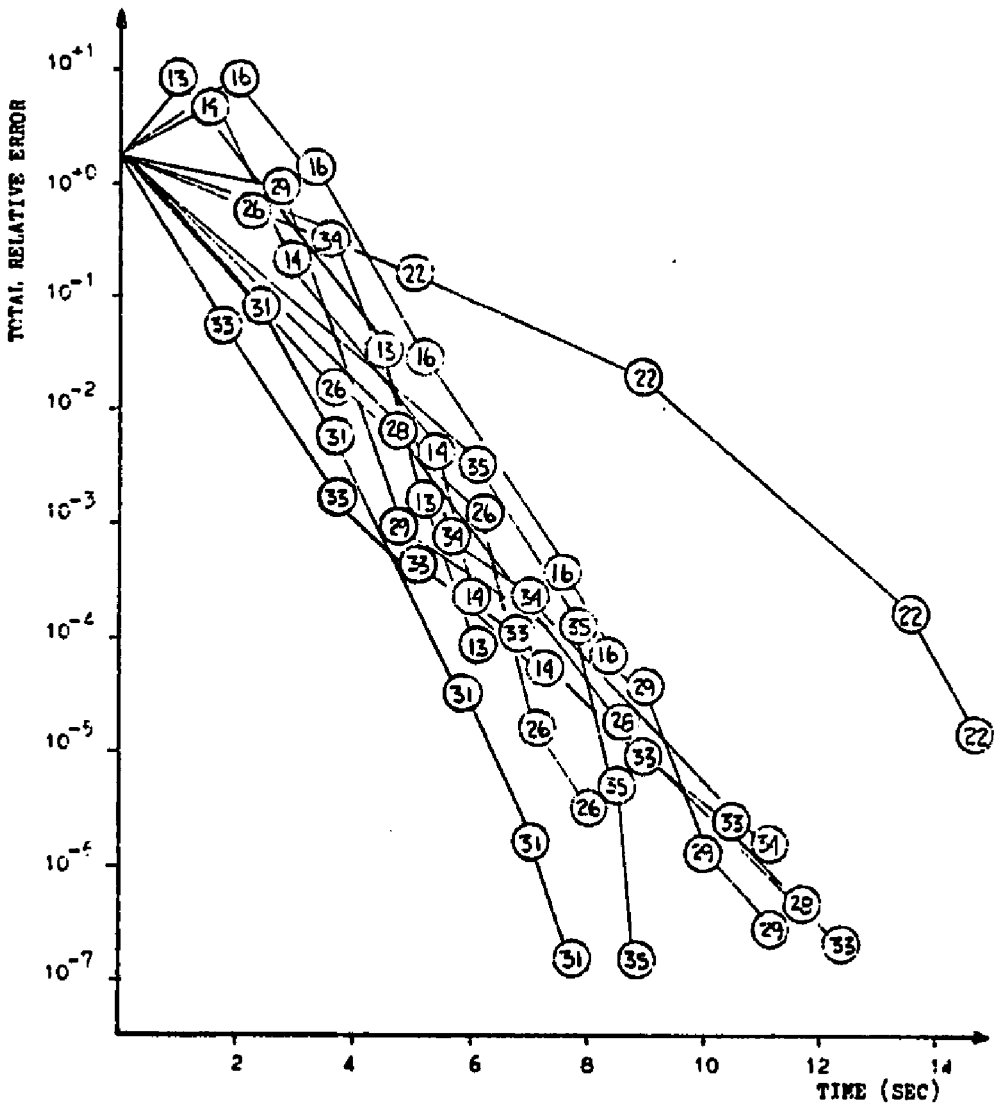

Figure C.1, (Cont'd)

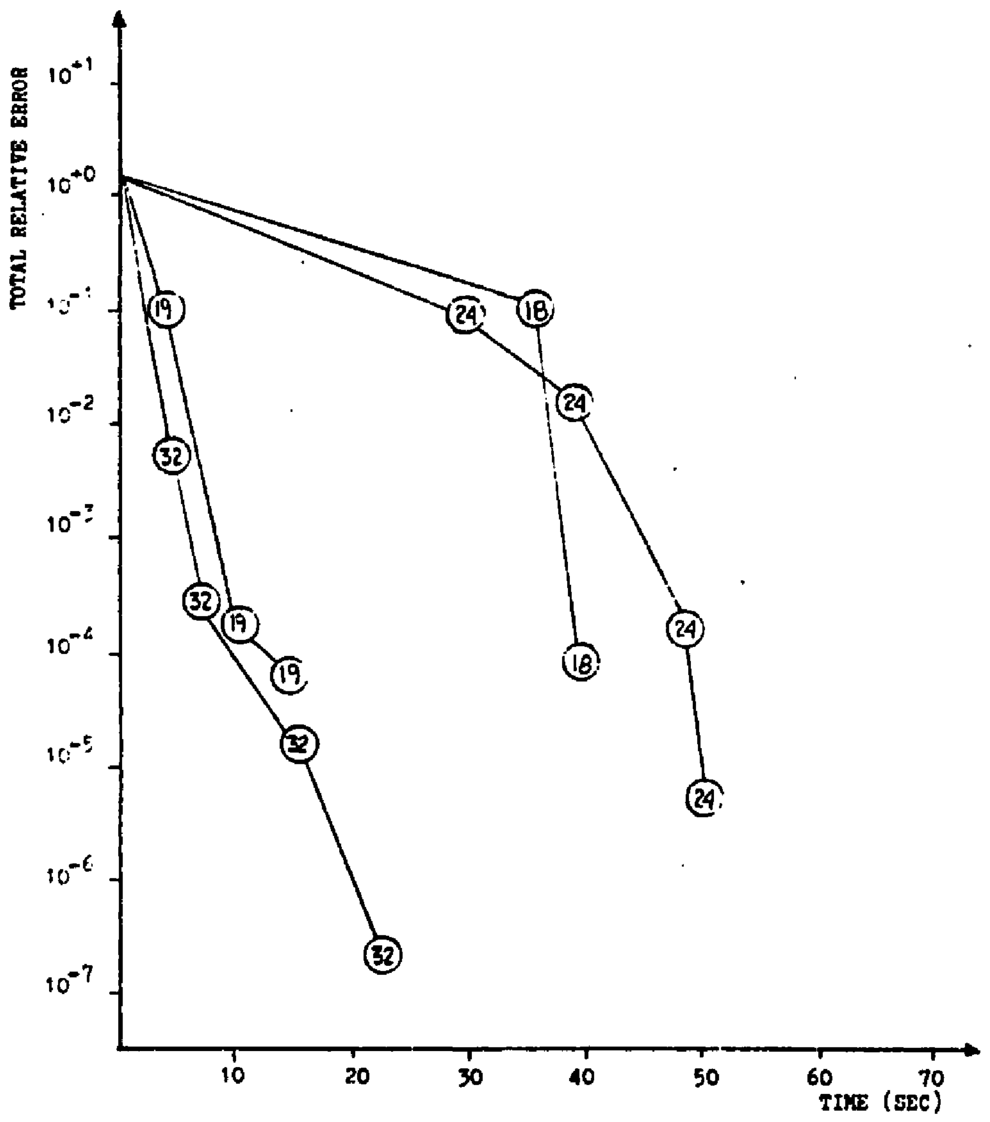

Figure C.1, (Cont'd)

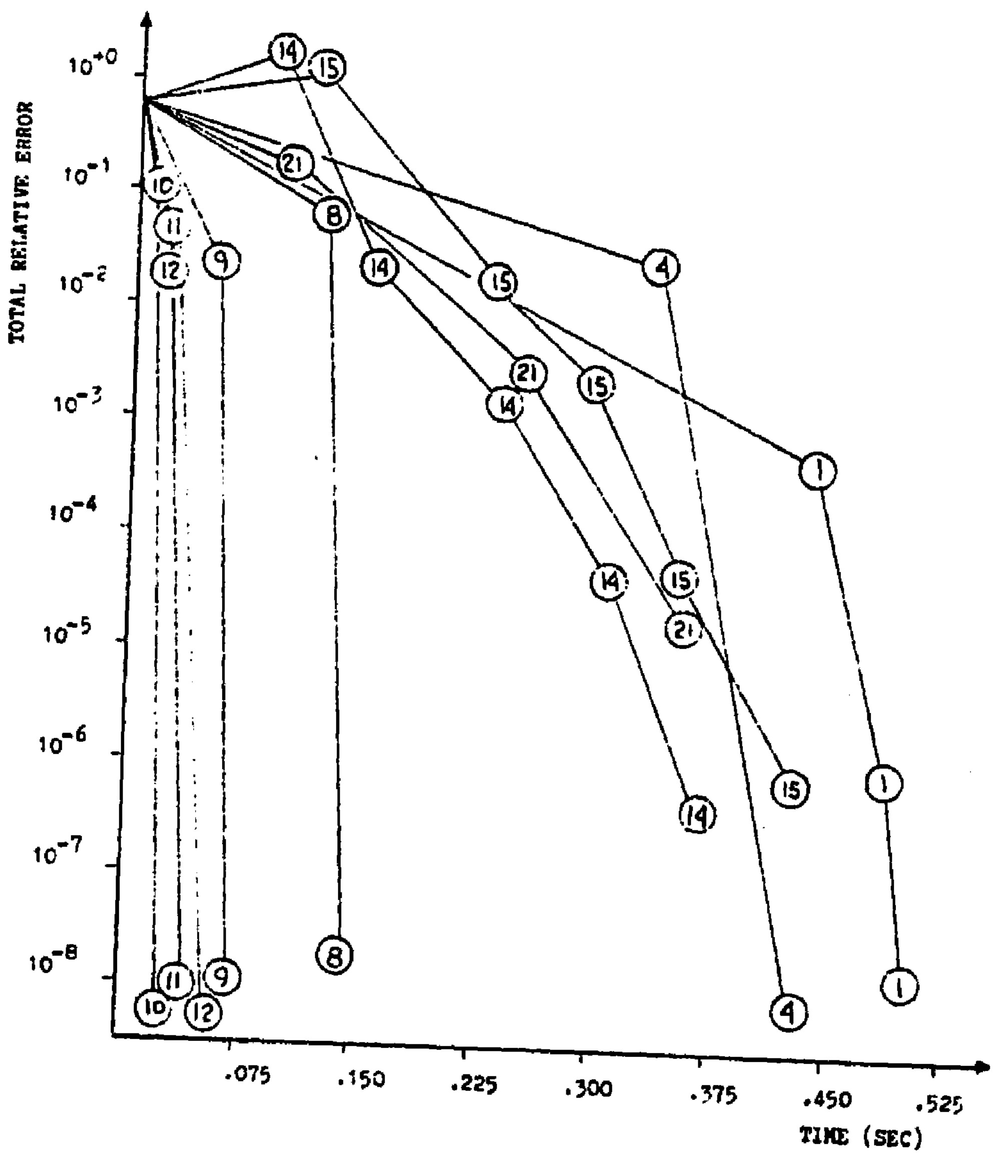

Figure C.2 Total Relative Error versus Time for Problem Problem Number 2.

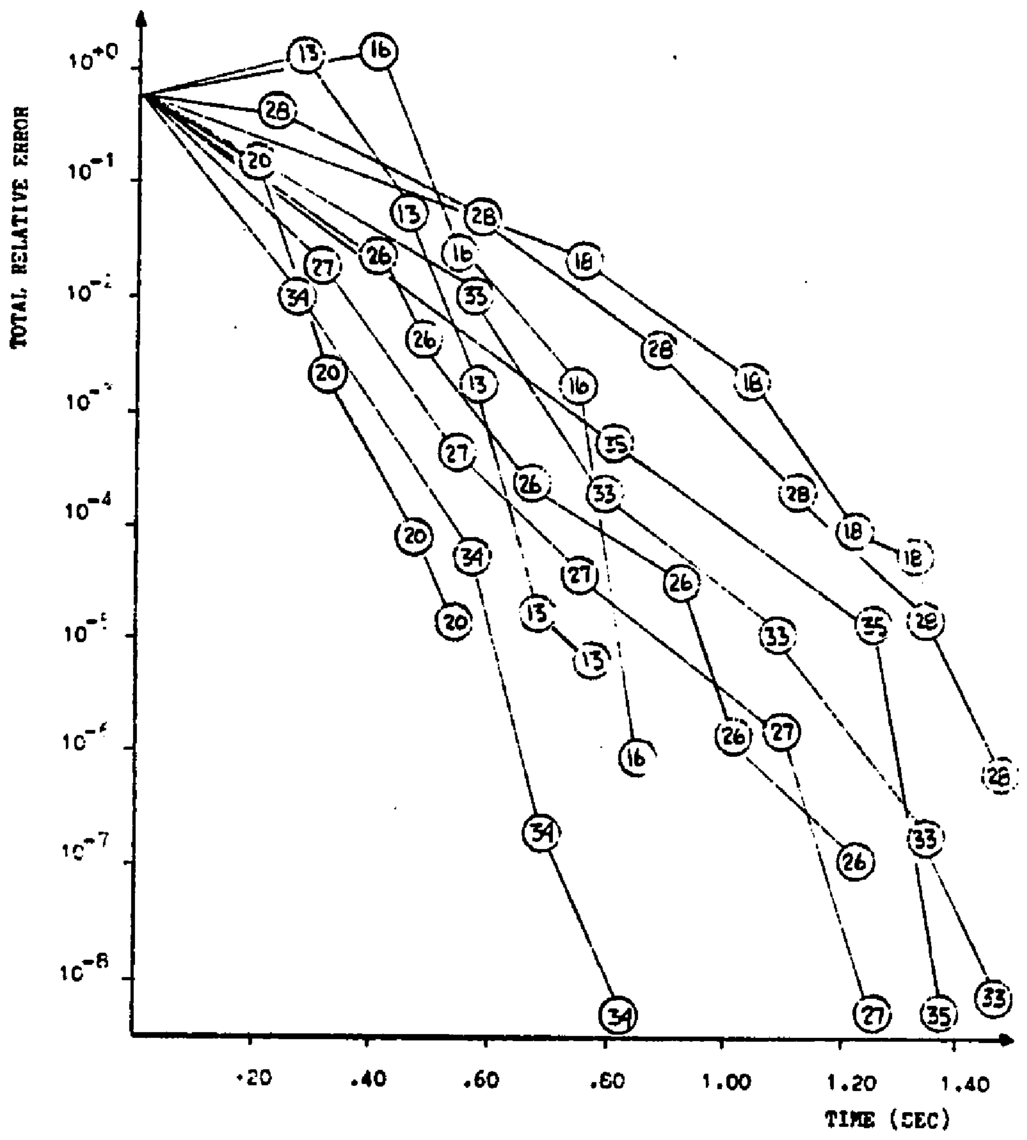

Figure C.2, (Cont'd)

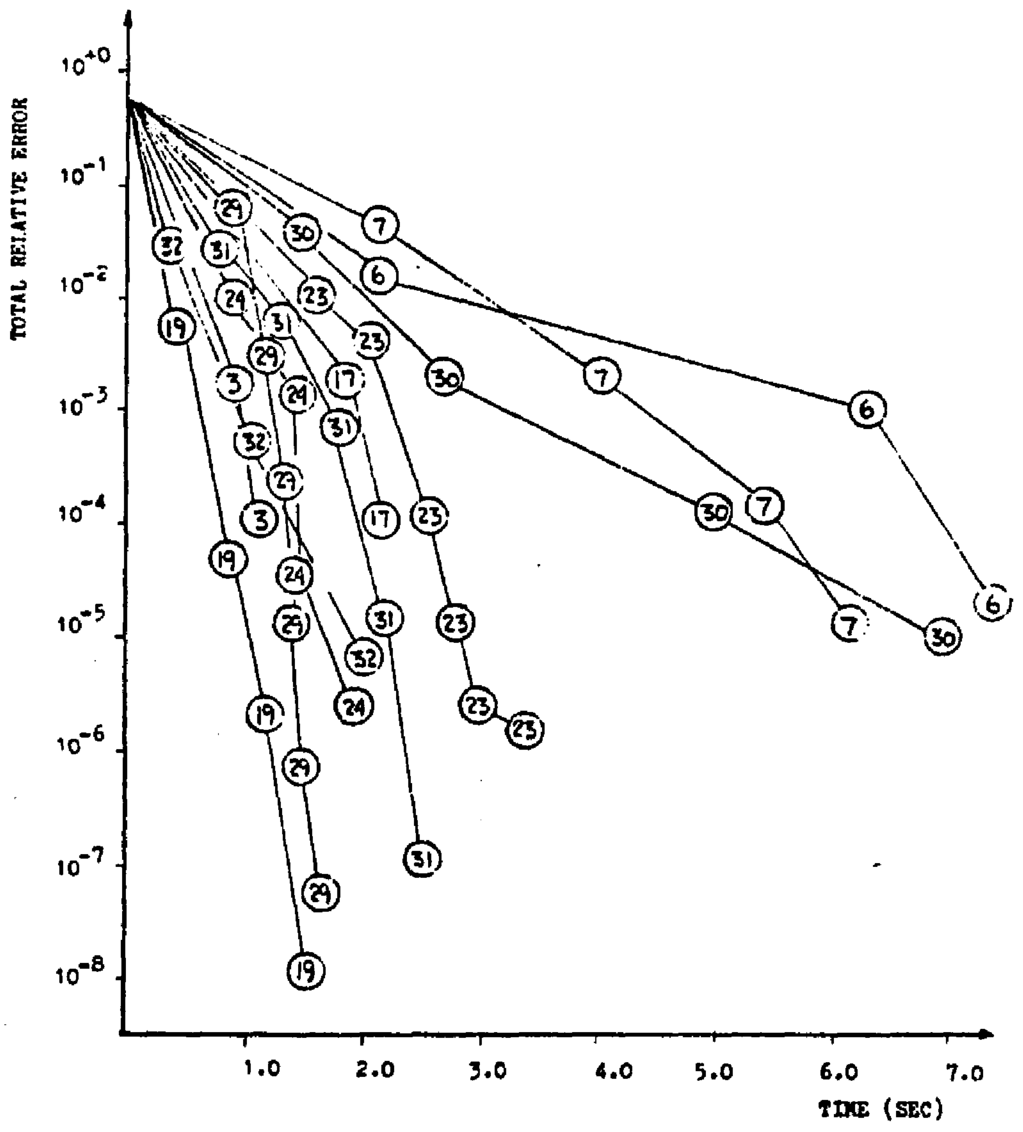

Figure C.2, (Cont'd)

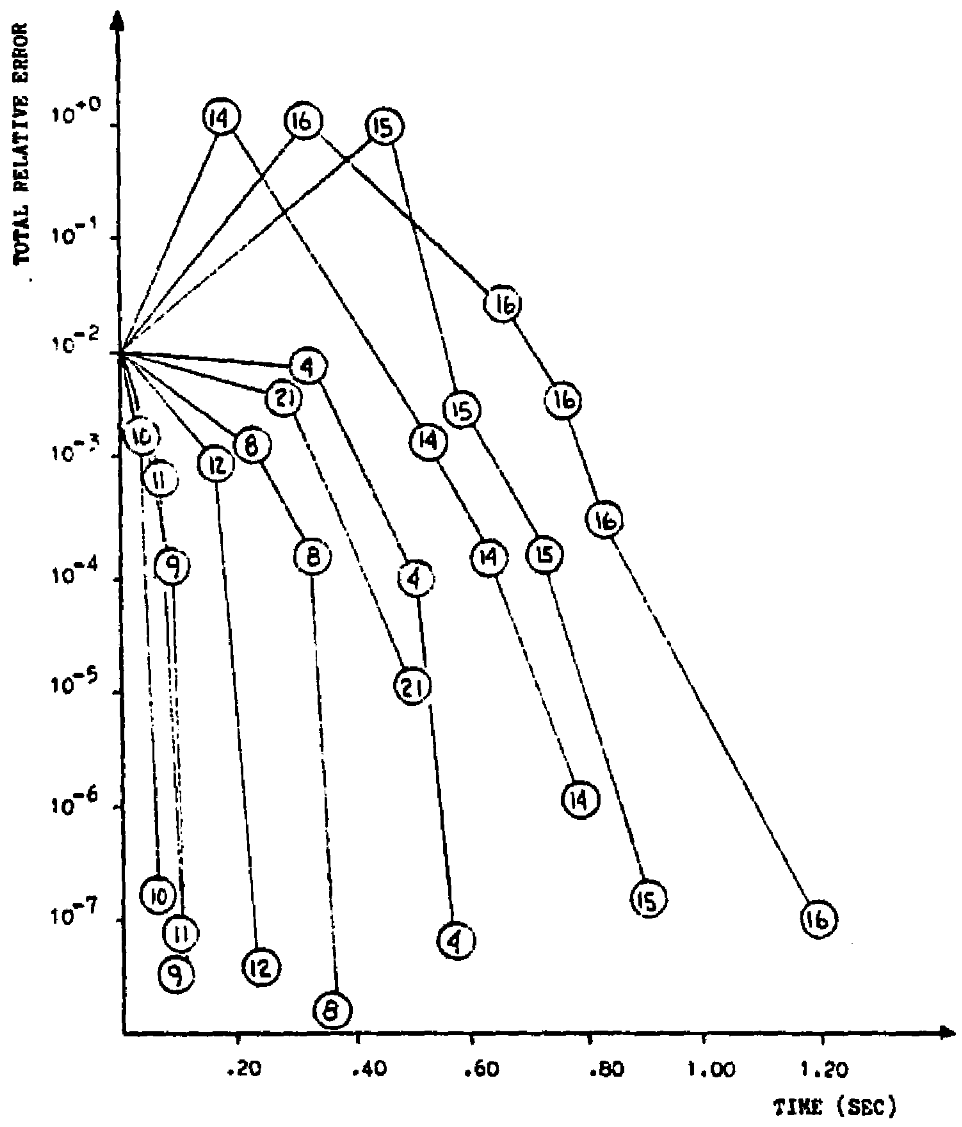

Figure C.3 Total Relative Error versus Time for Problem Number 3.

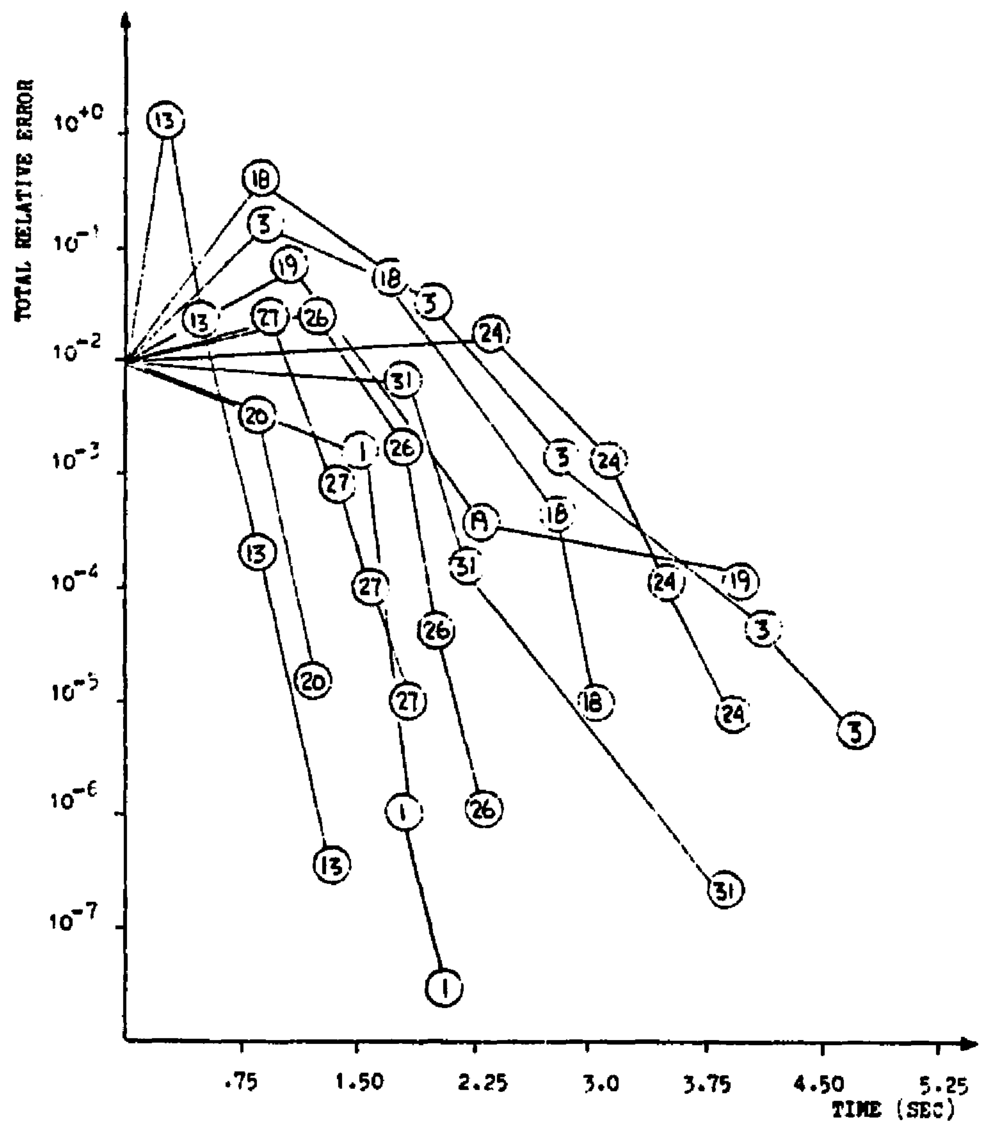

Figure C.3, (Cont'd)

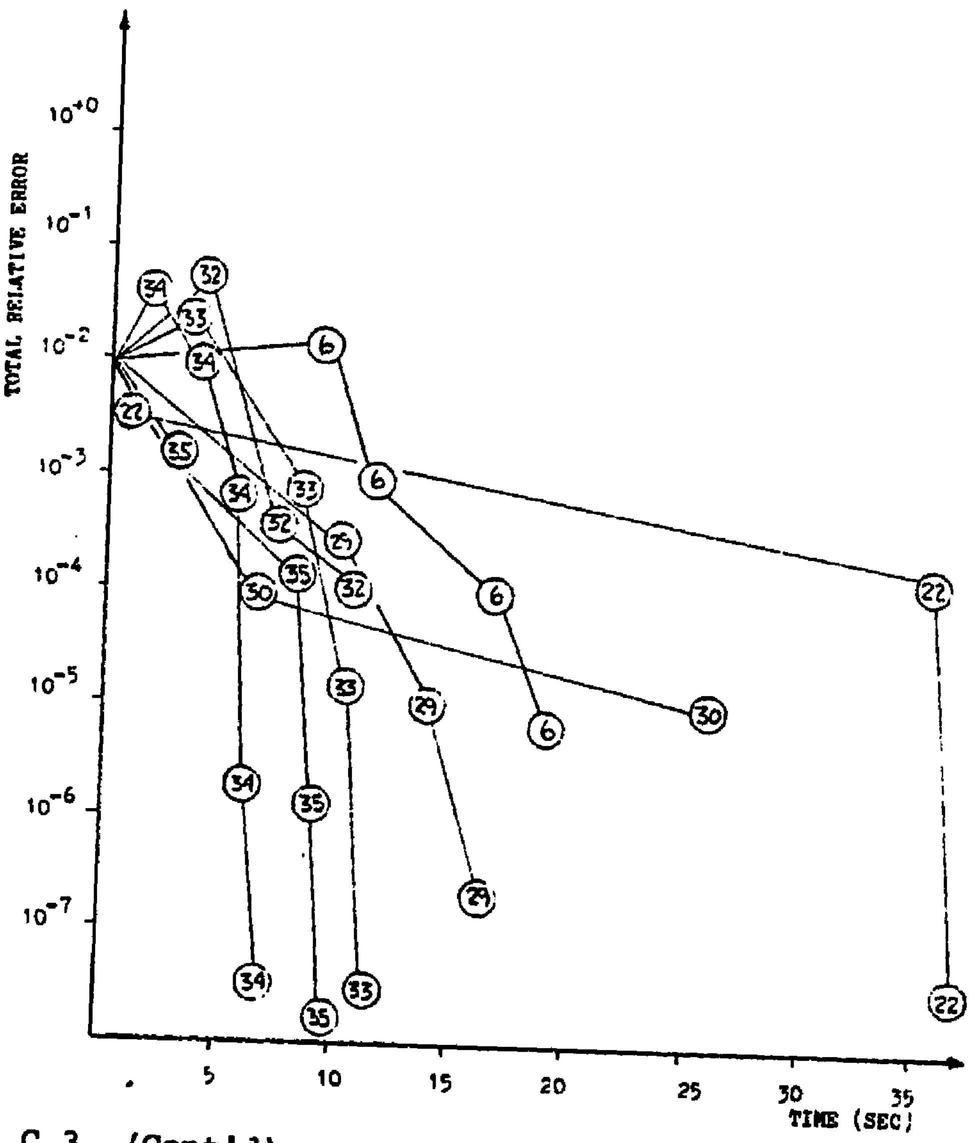

Figure C.3, (Cont'd)

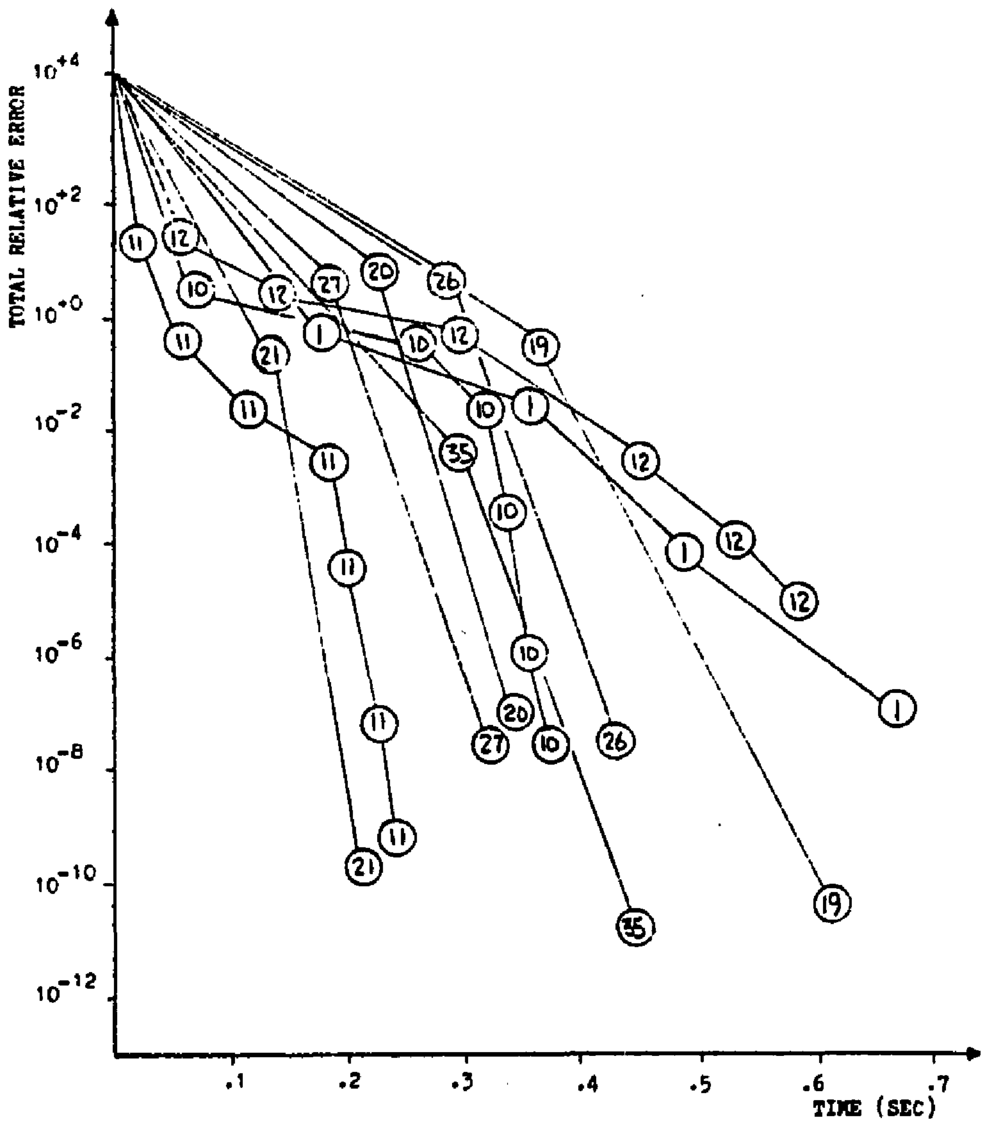

Figure C.4 Total Relative Error versus Time for Problem Number 4.

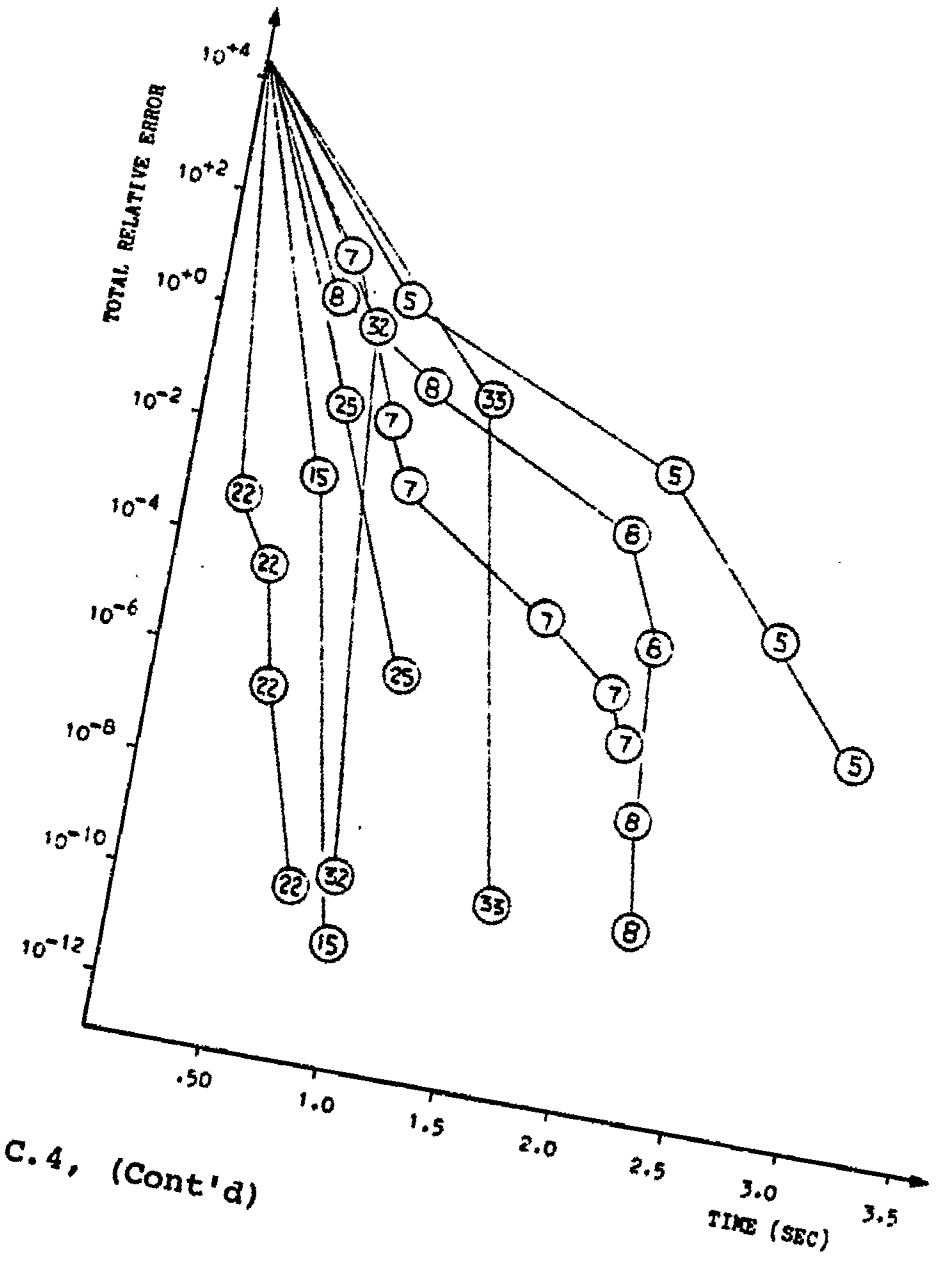

Figure C.4, (Cont'd)

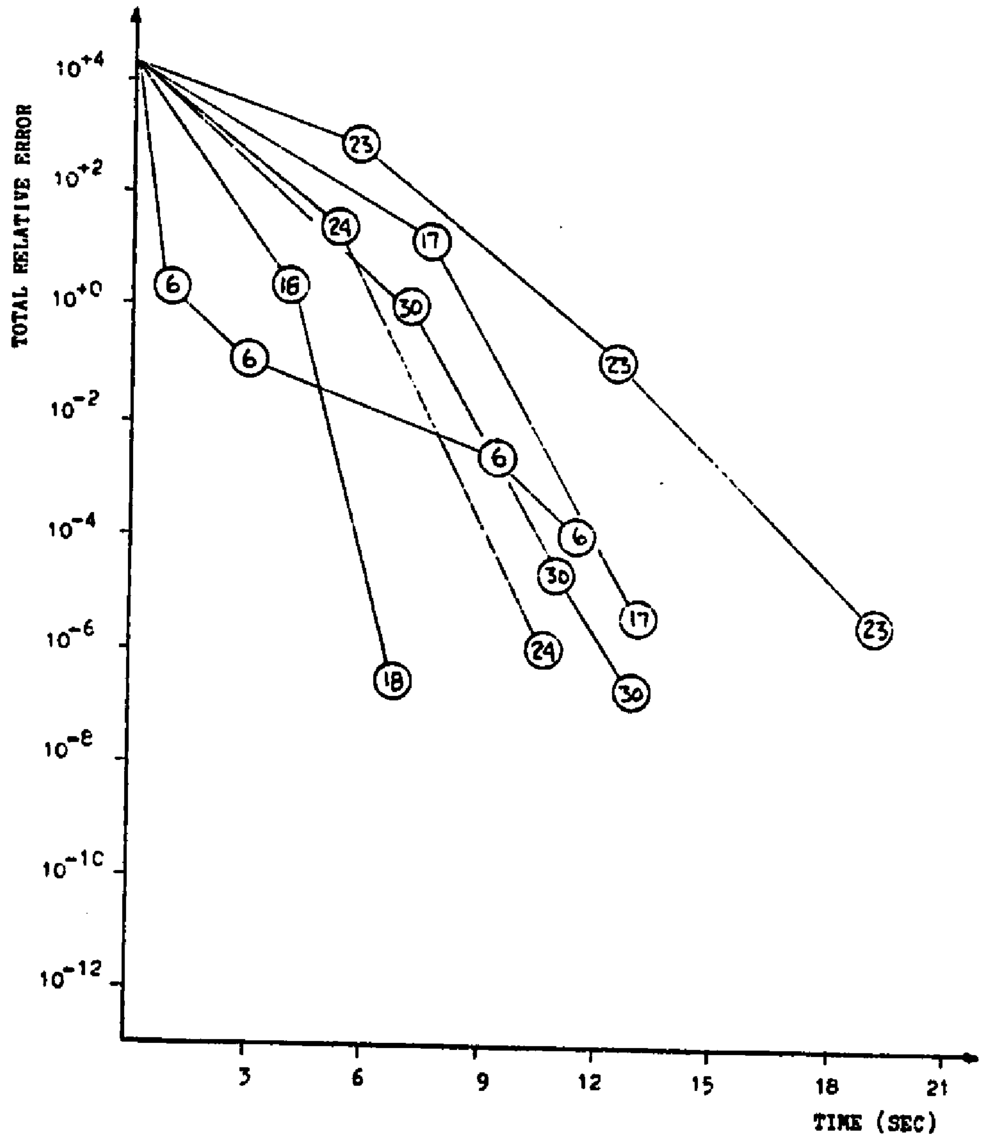

Figure C.4, (Cont'd)

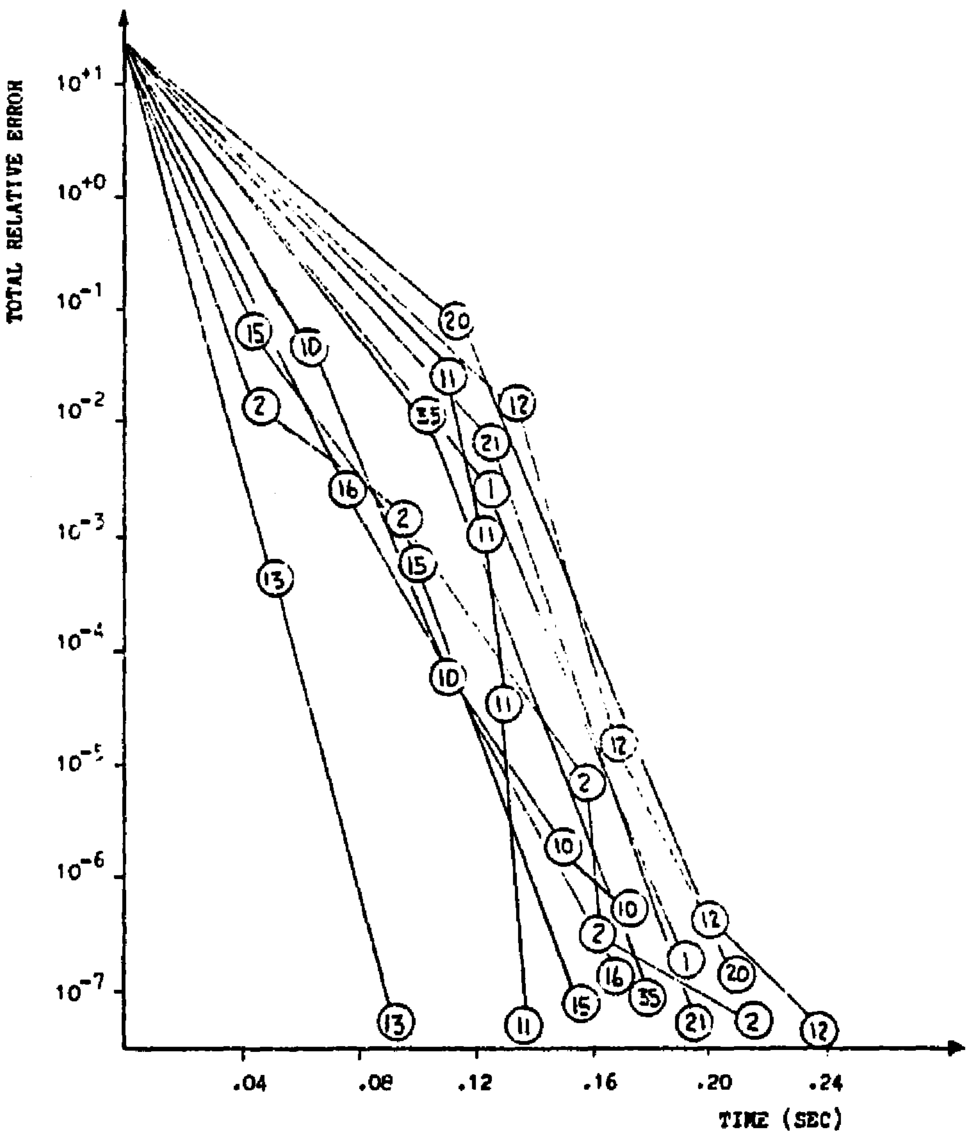

Figure C.5 Total Relative Error versus Time for Problem Number 5.

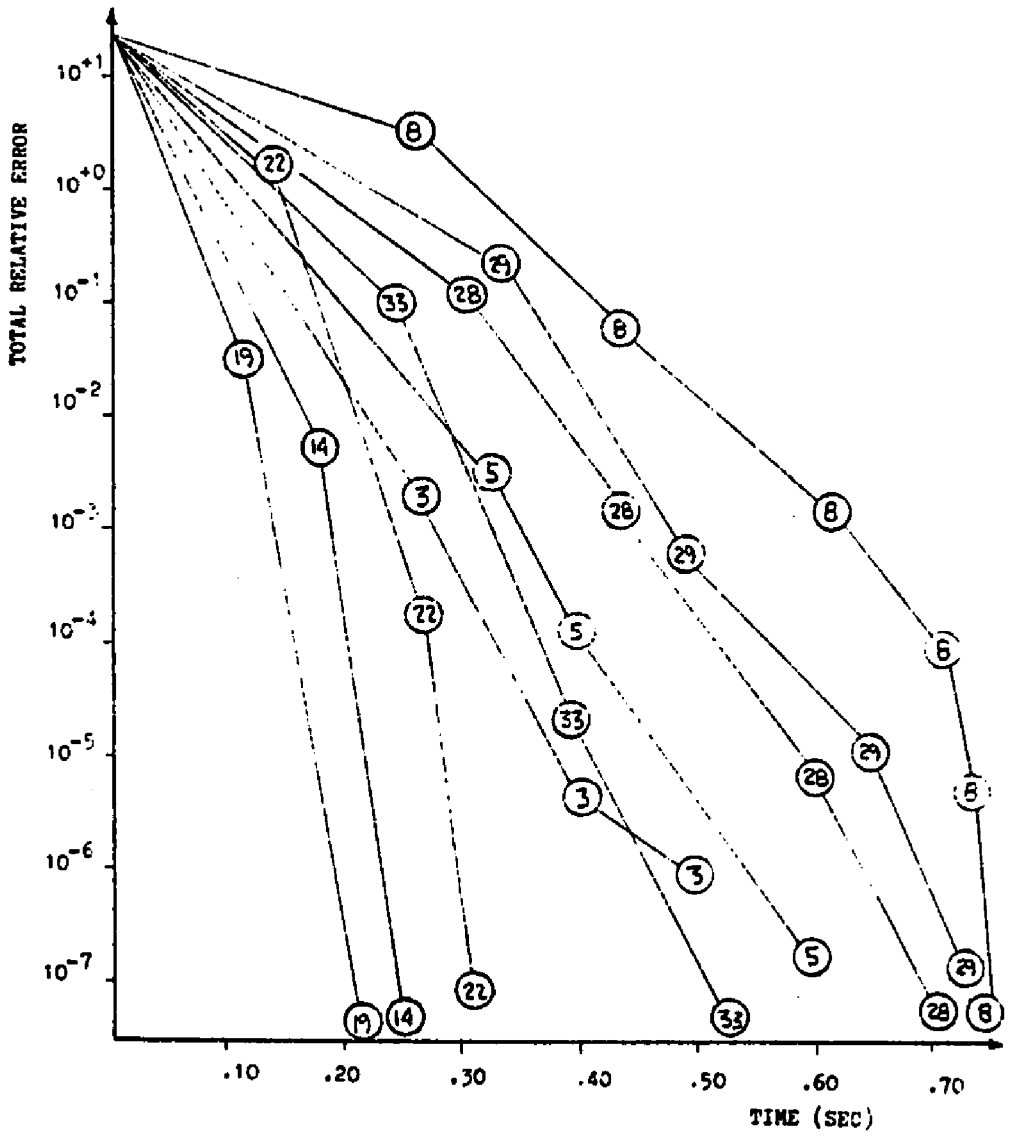

Figure C.5, (Cont'd)

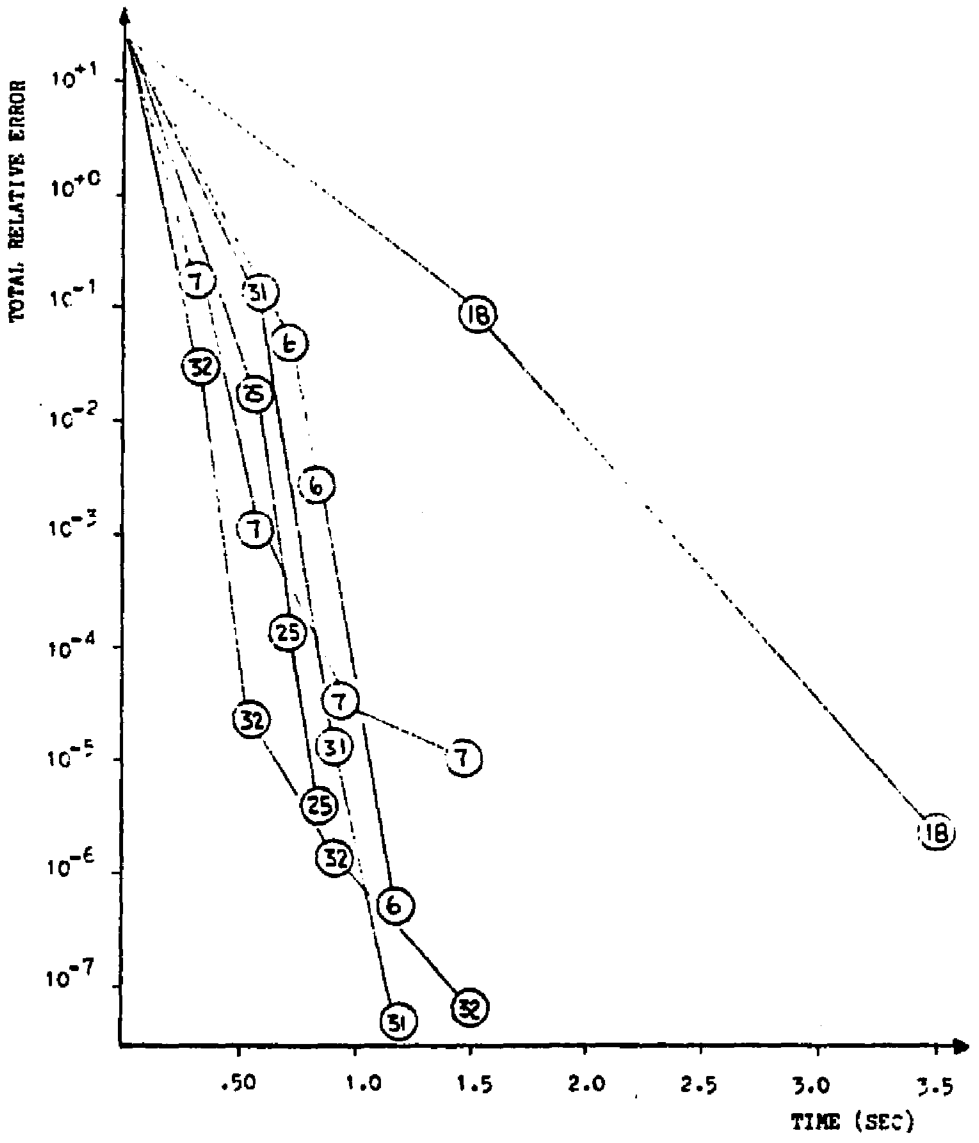

Figure C.5, (Cont'd)

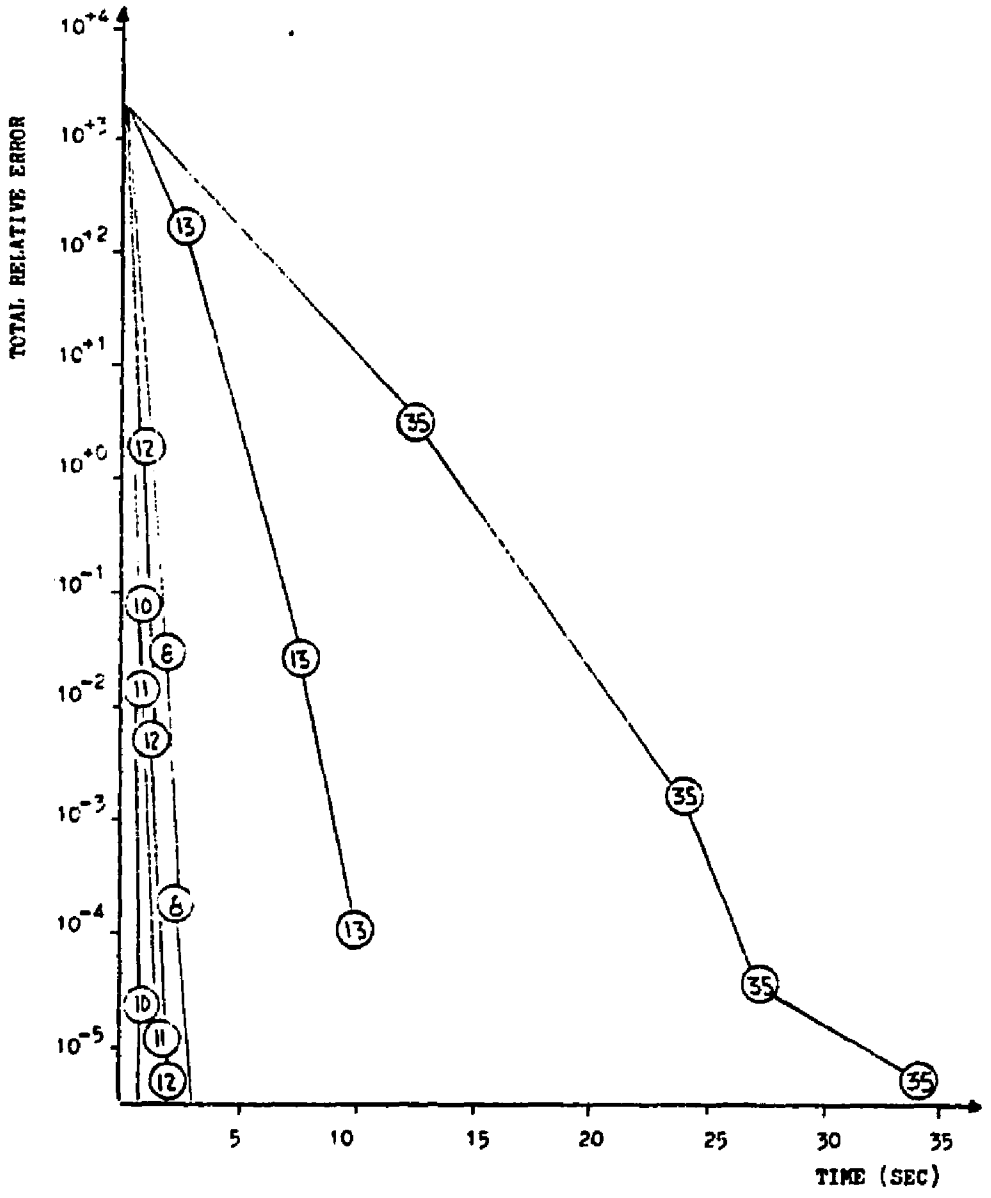

Figure C.6 Total Relative Error versus Time for Problem Number 6.

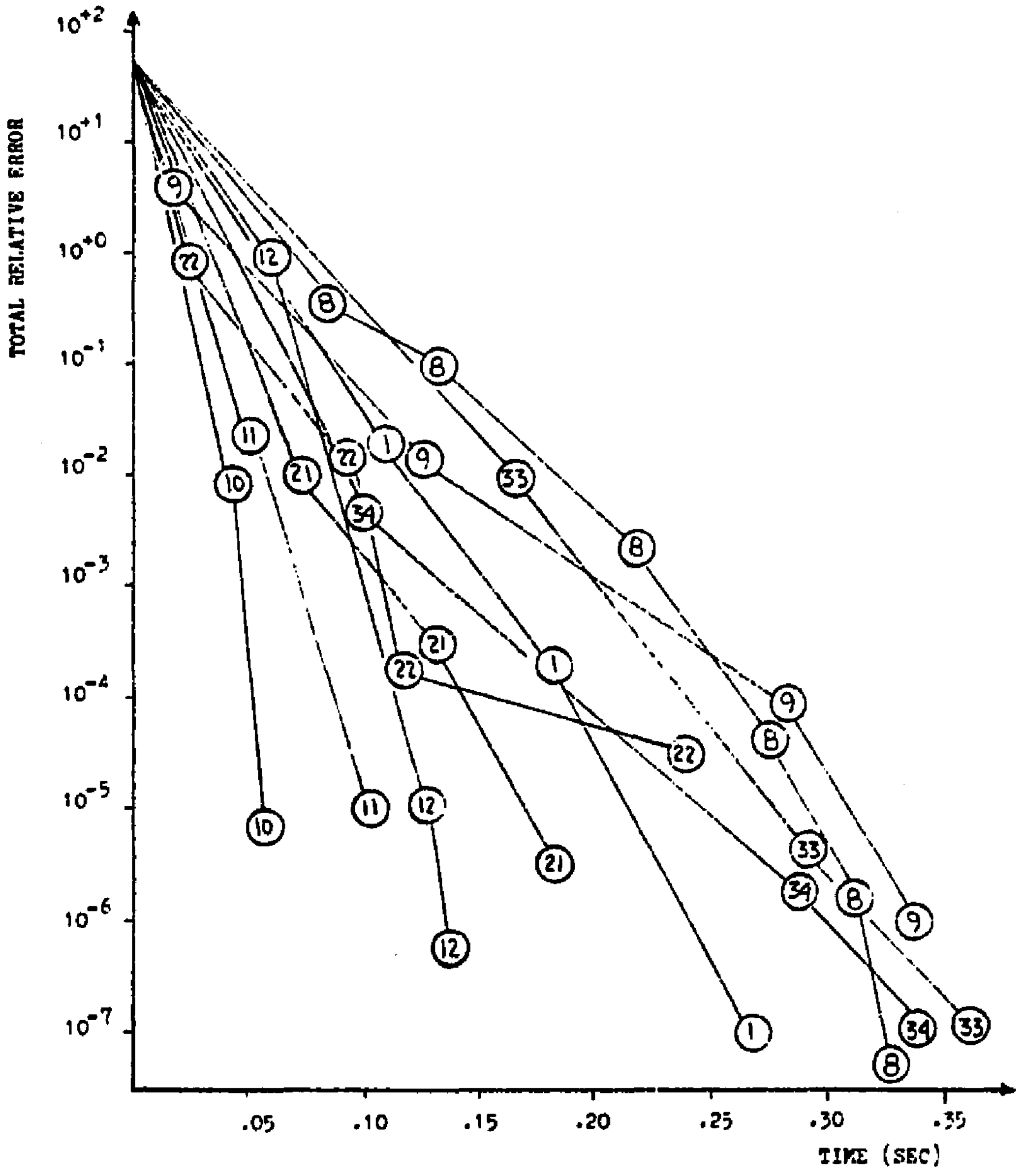

Figure C.7 Total Relative Error versus Time for Problem Number 7.

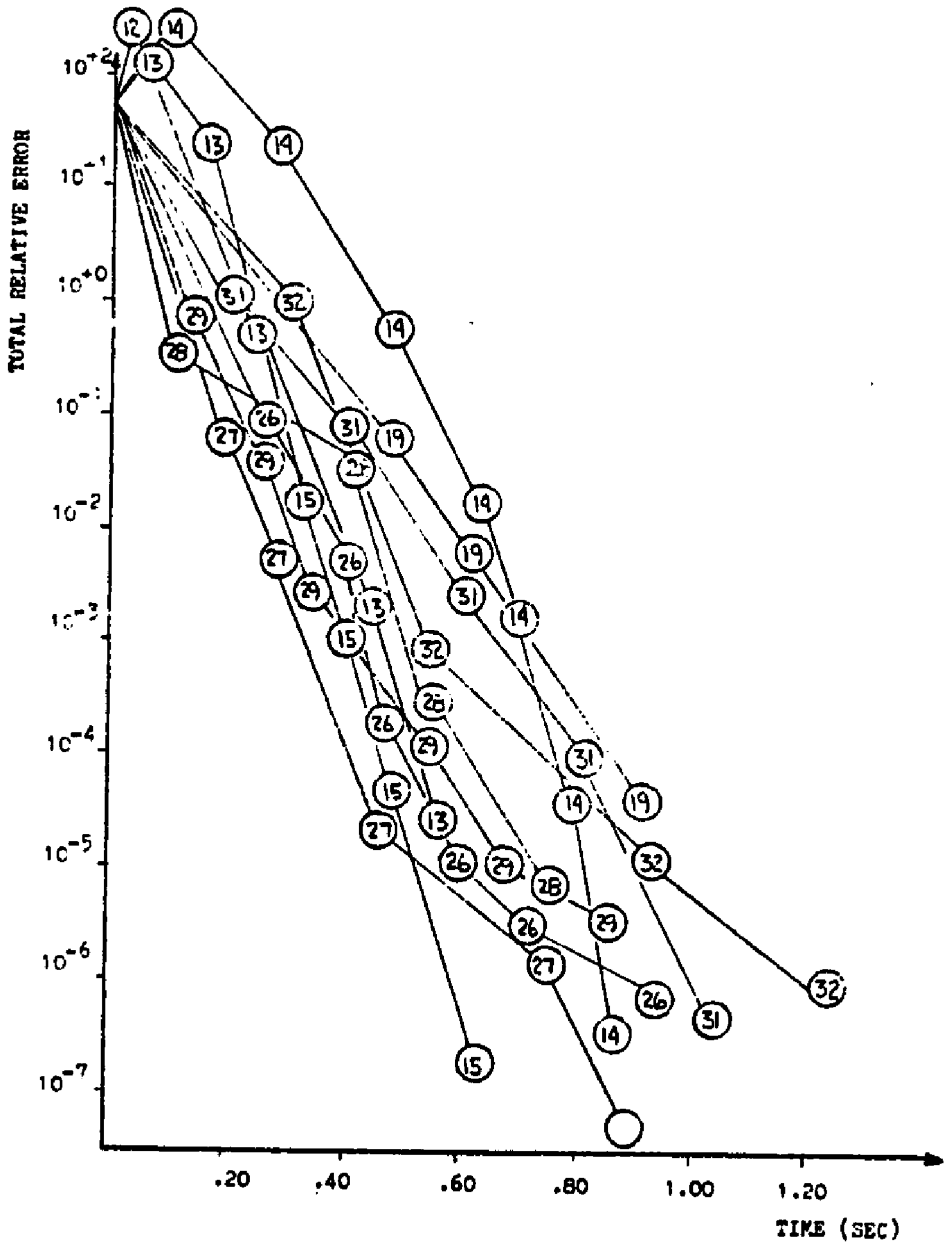

Figure C.7, (Cont'd)

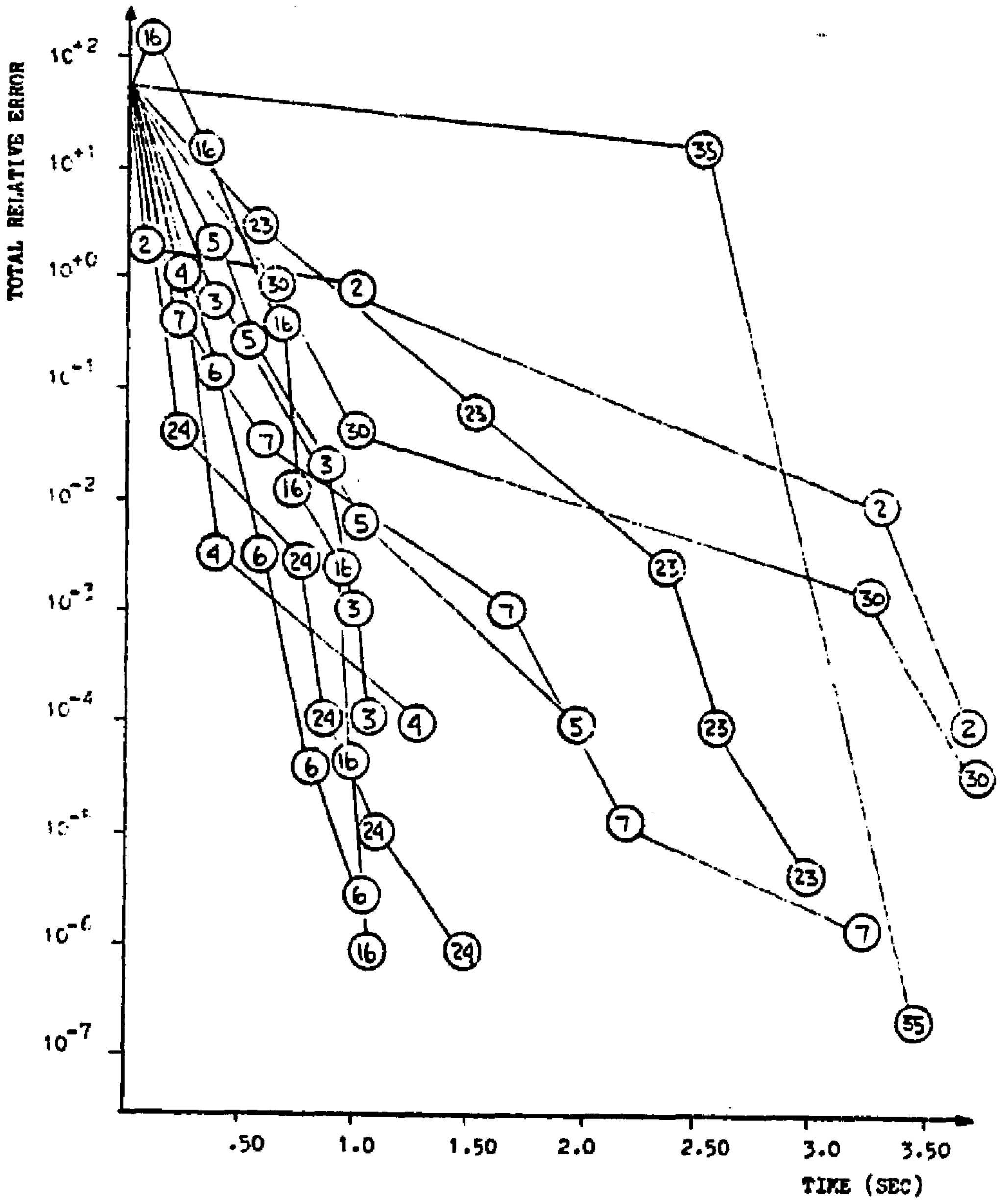

Figure C.7, (Cont'd)

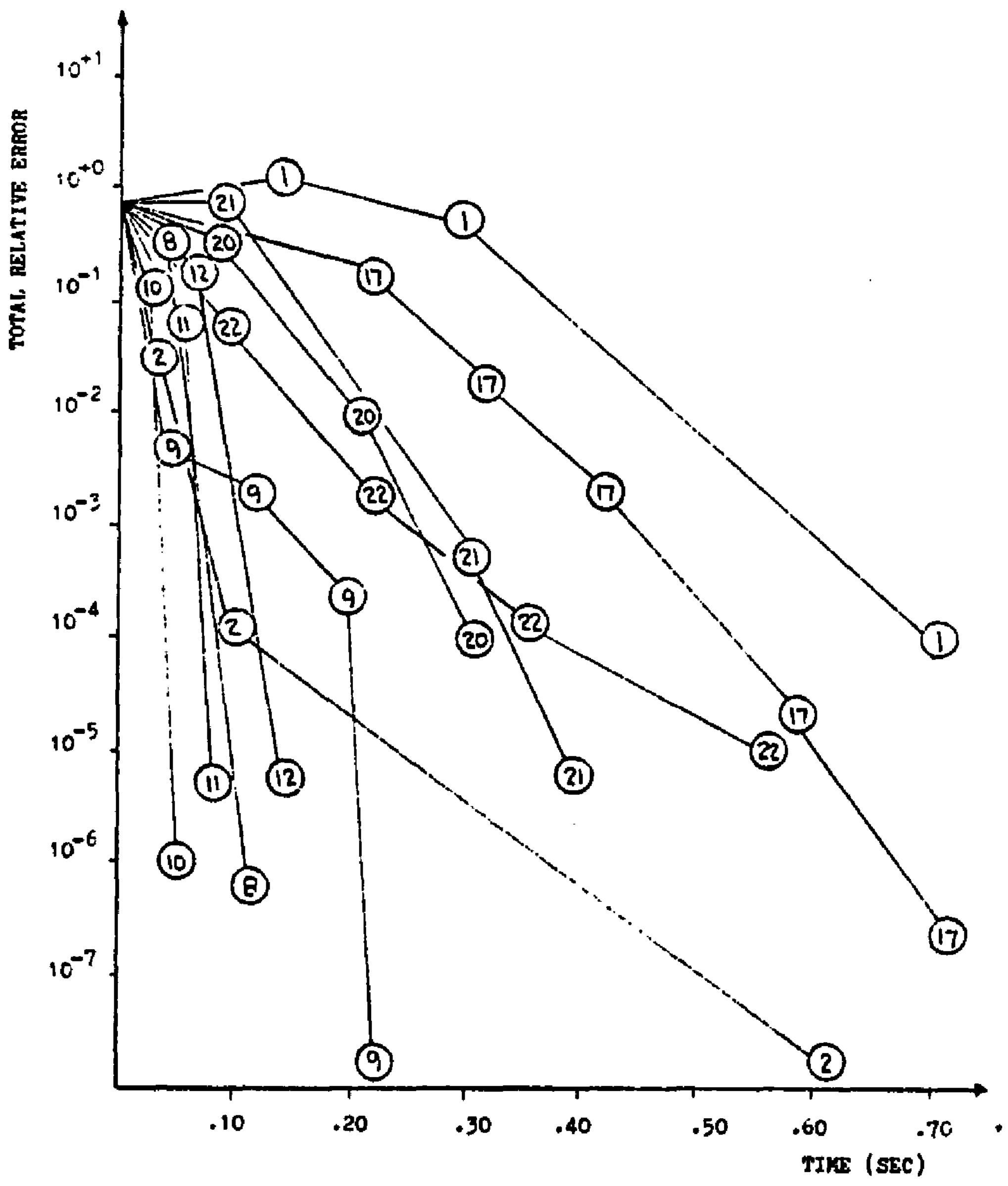

Figure C.8 Total Relative Error versus Time for Problem Number 8.

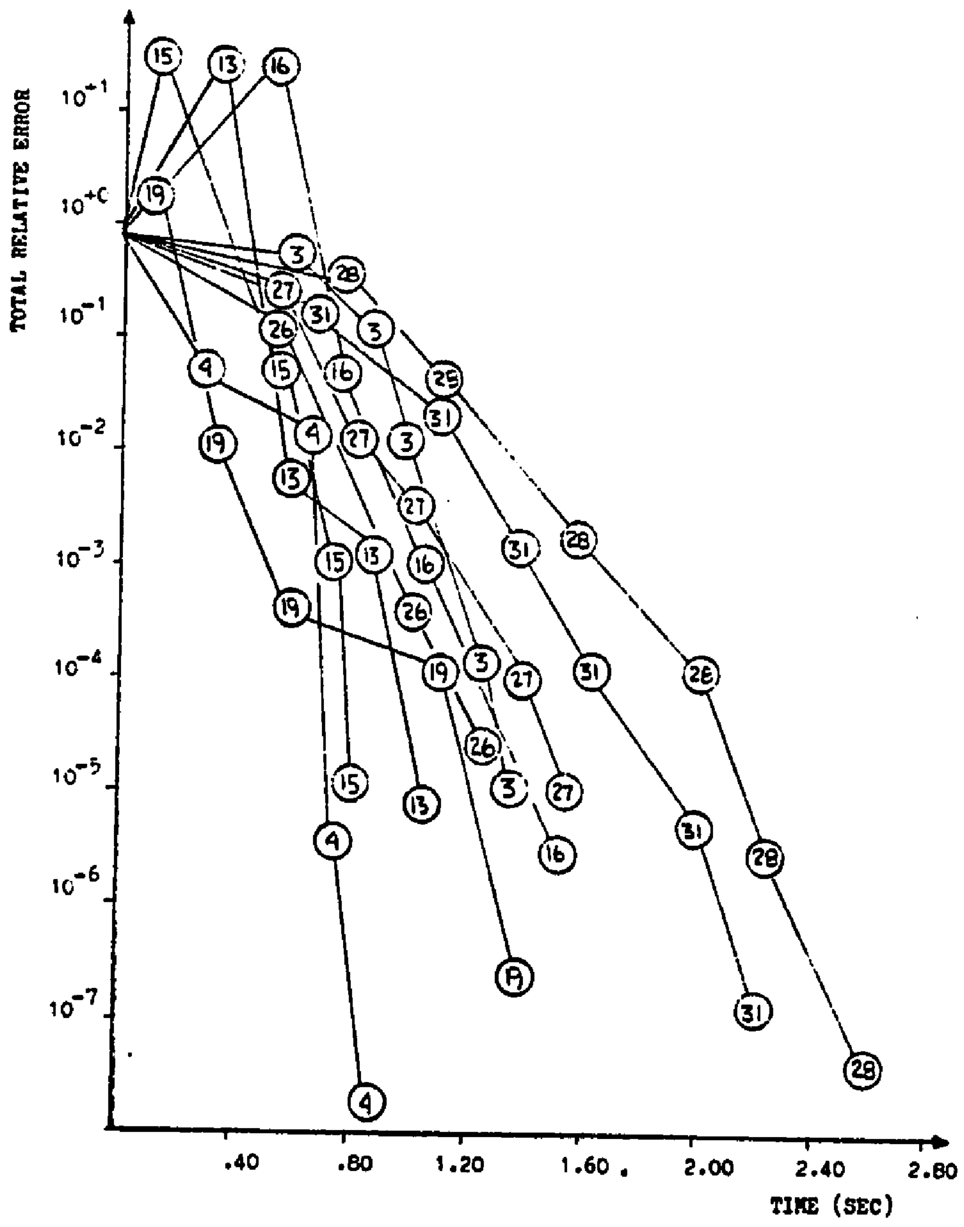

Figure C.8, (Cont'd)

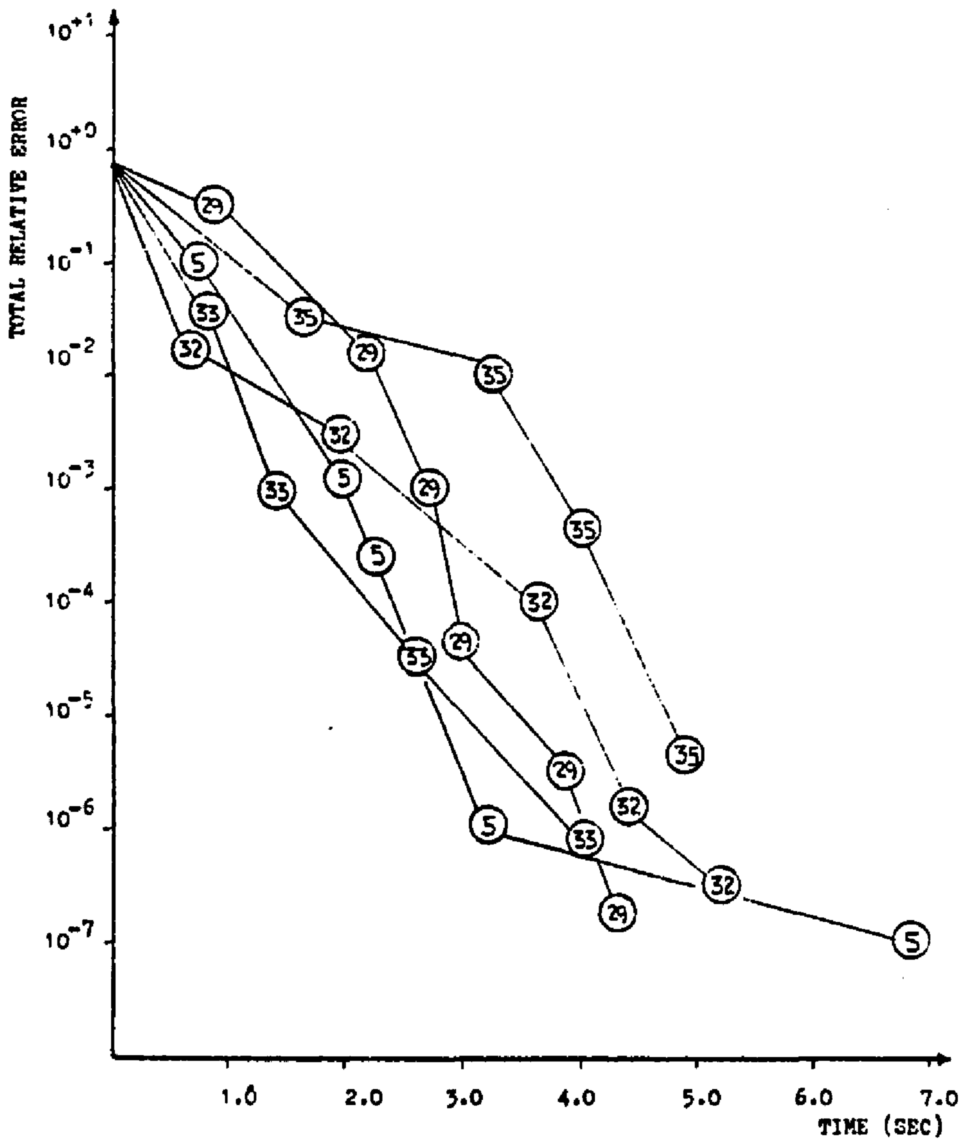

Figure C.8, (Cont'd)

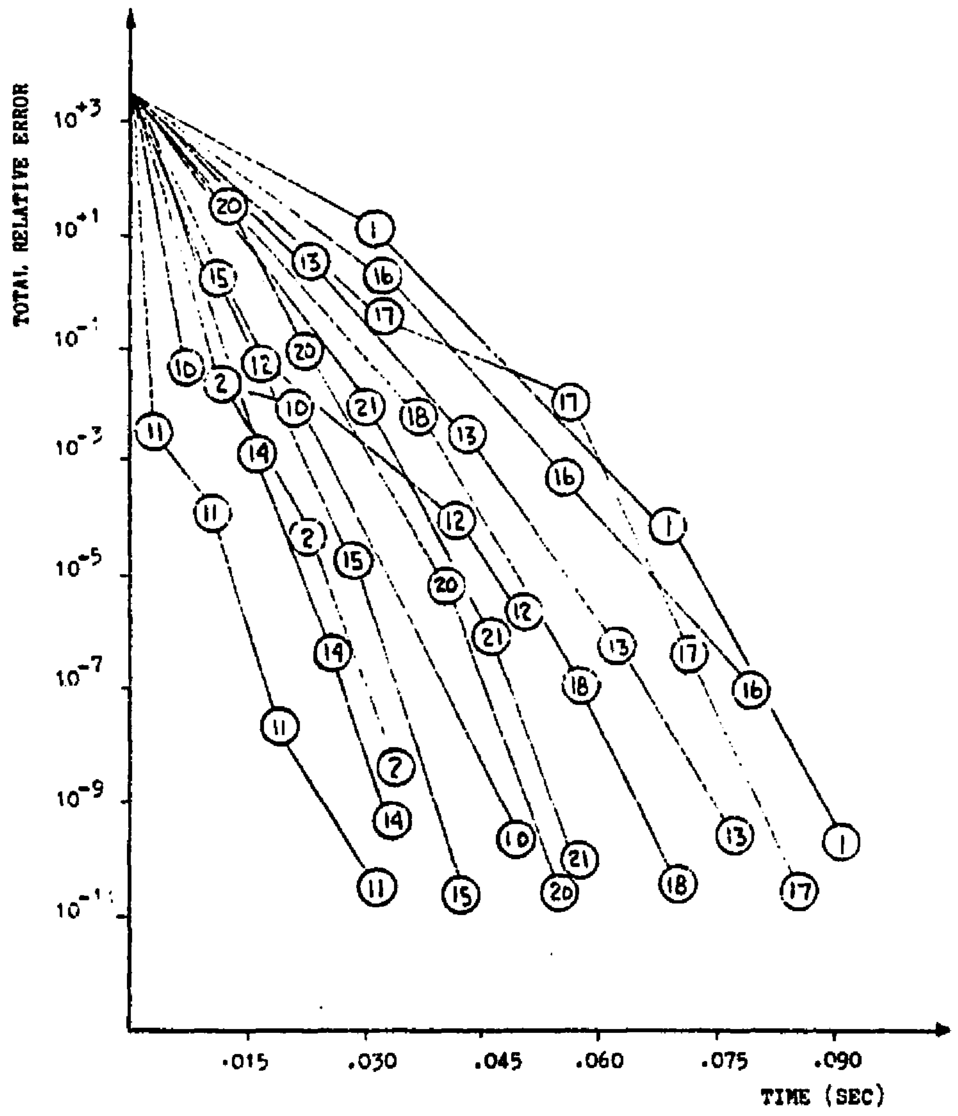

Figure C.9 Total Relative Error versus Time for Problem Number 10.

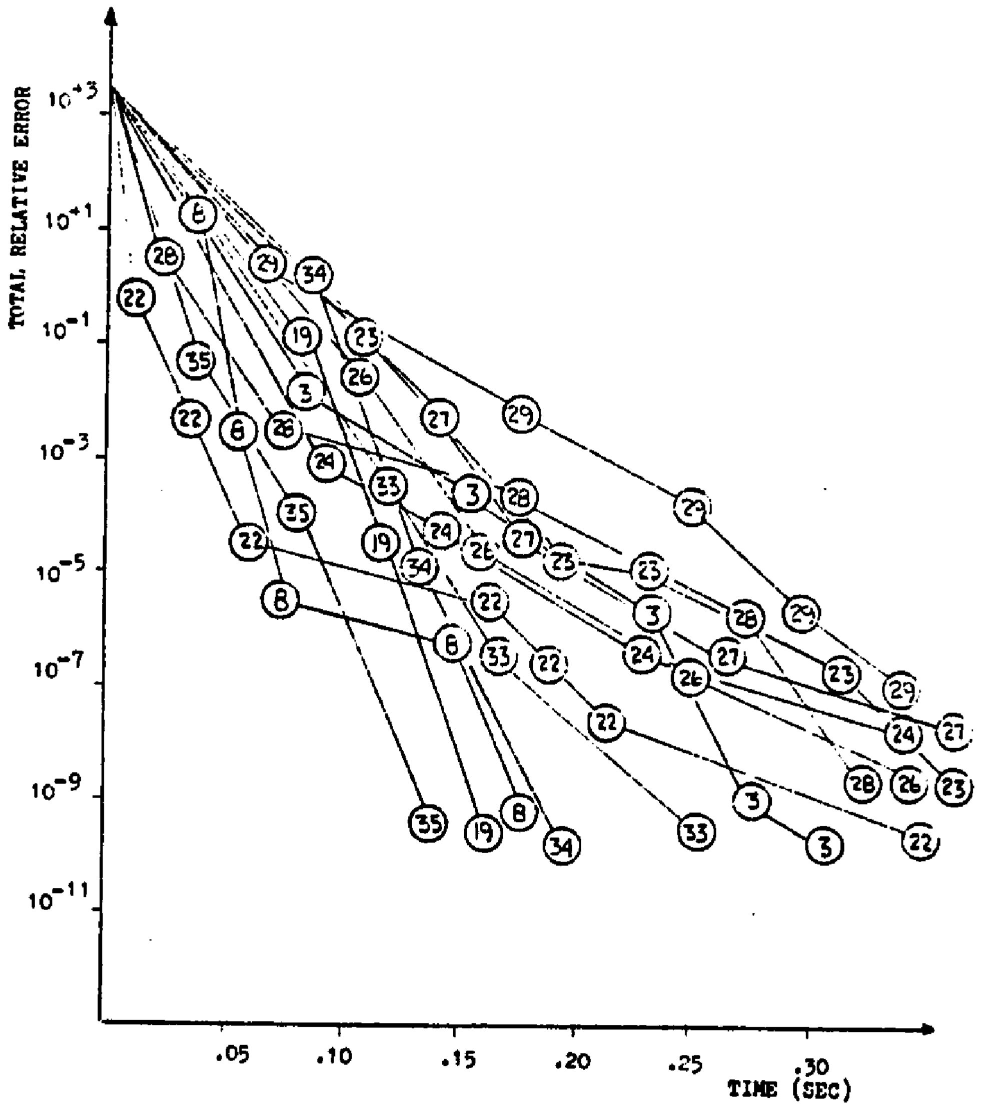

Figure C.9, (Cont'd)

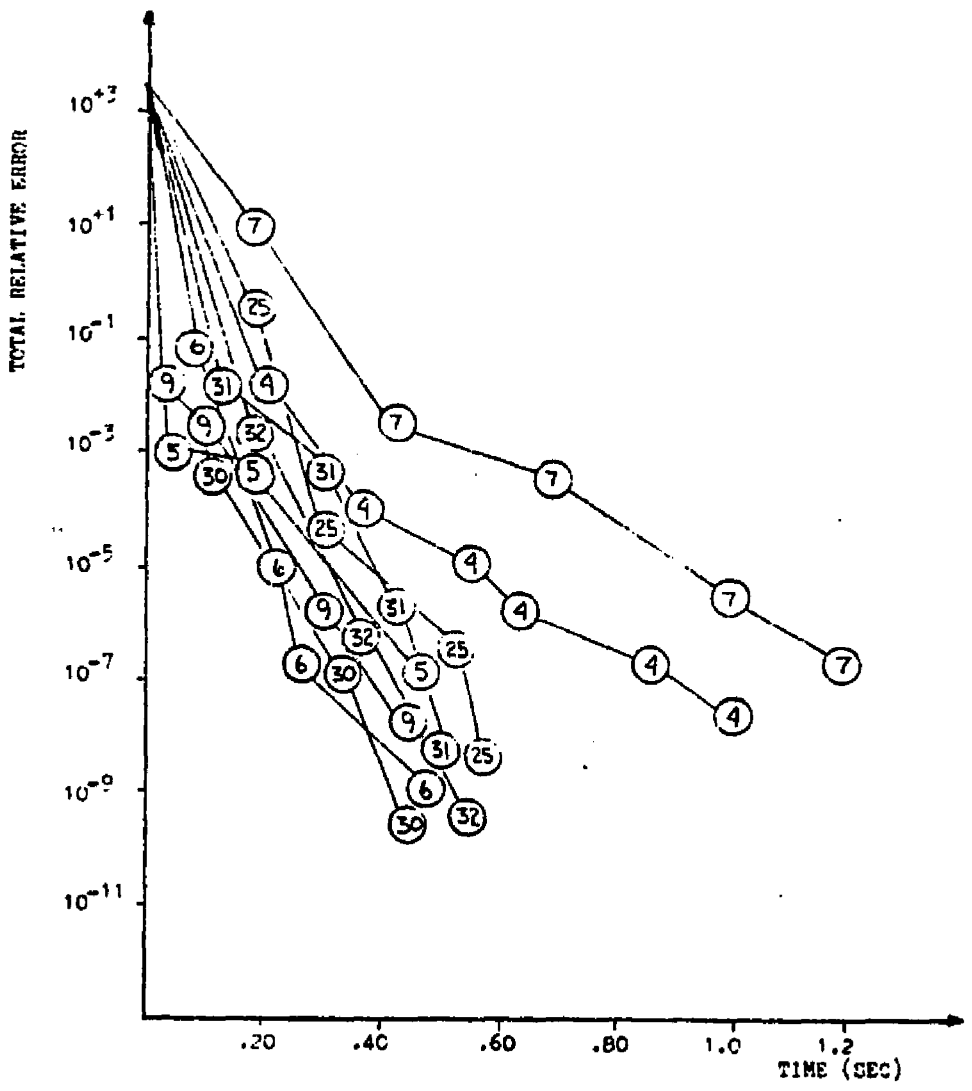

Figure C.9, (Cont'd)

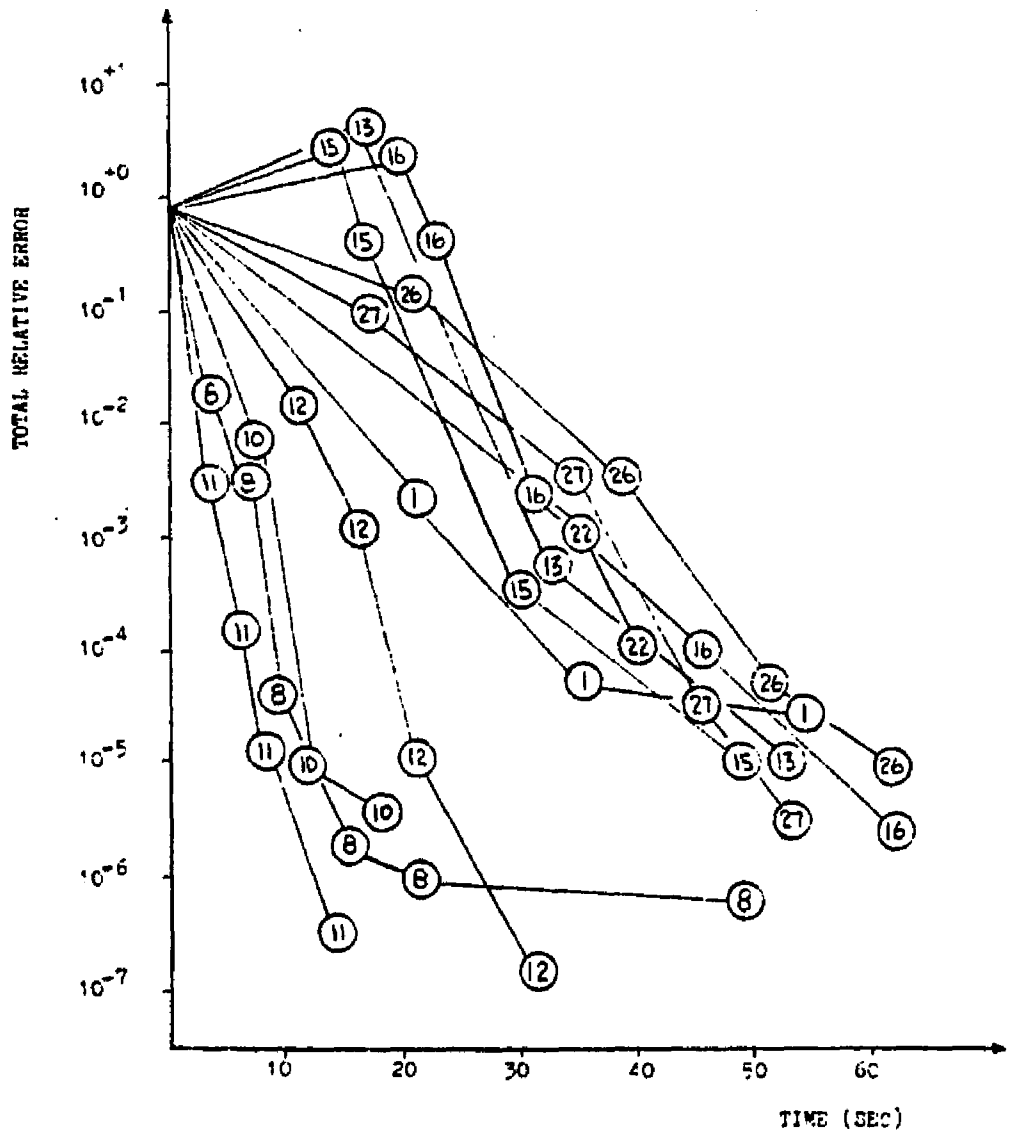

Figure C.10 Total Relative Error versus Time for Problem Number 11.

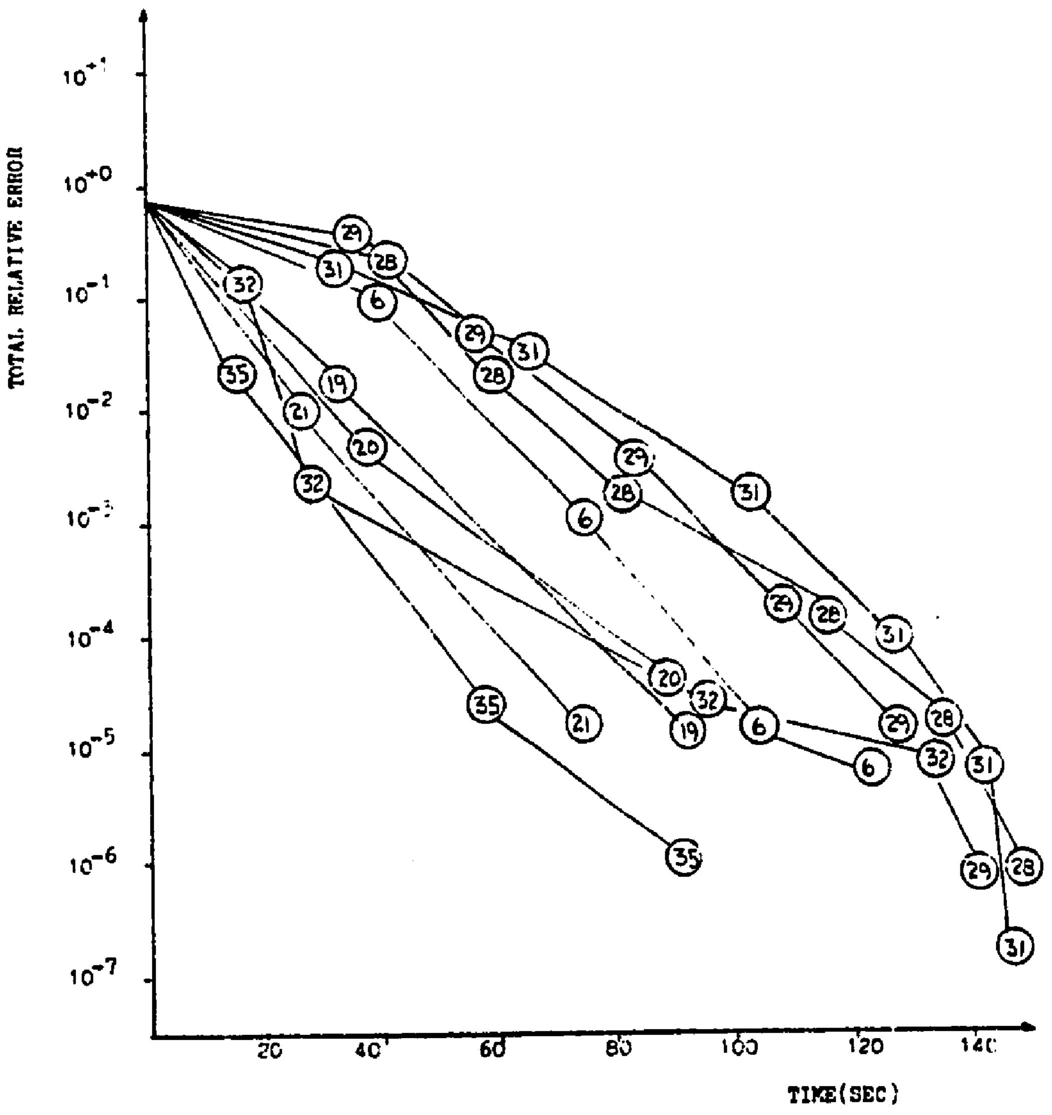

Figure C.10, (Cont'd)

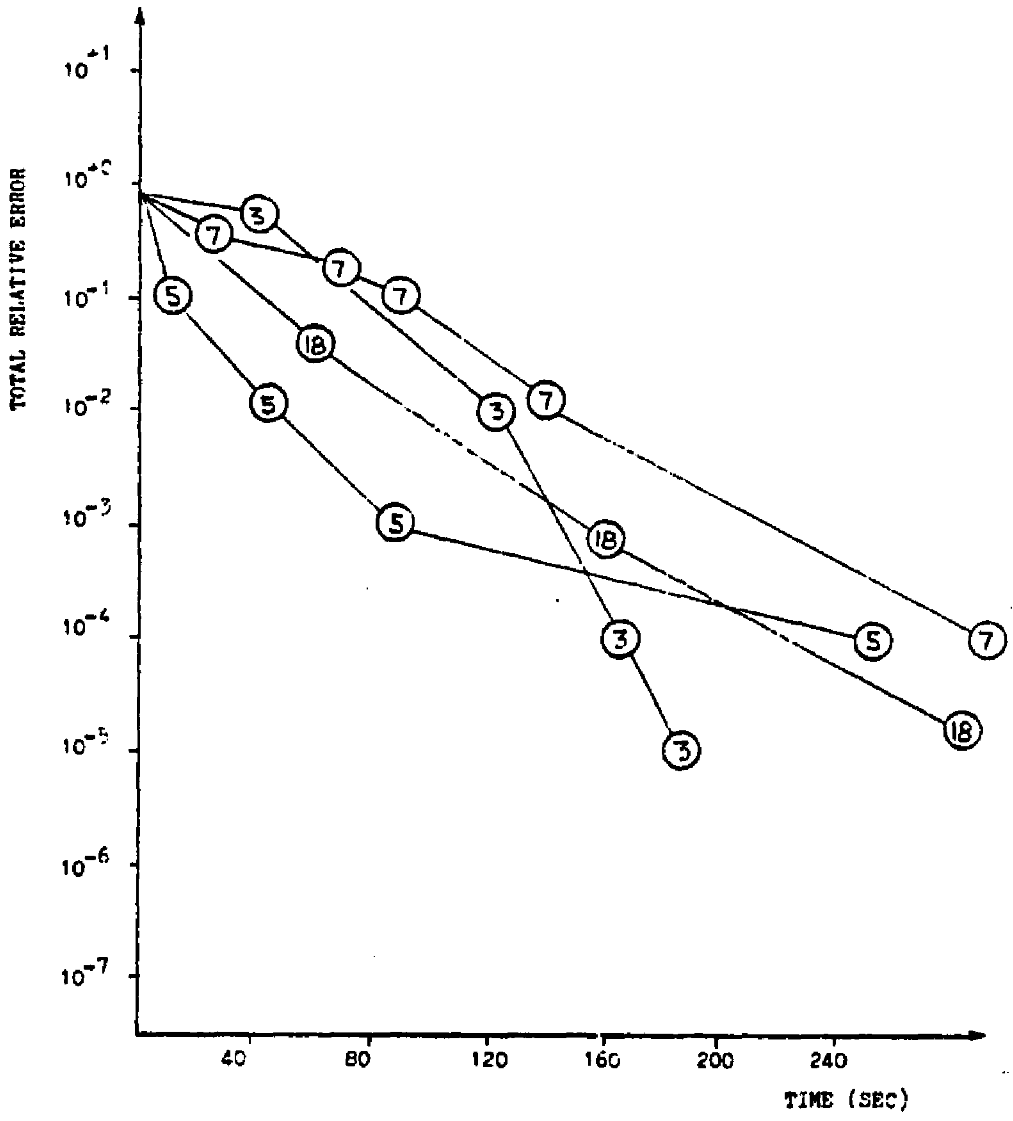

Figure C.10, (Cont'd)

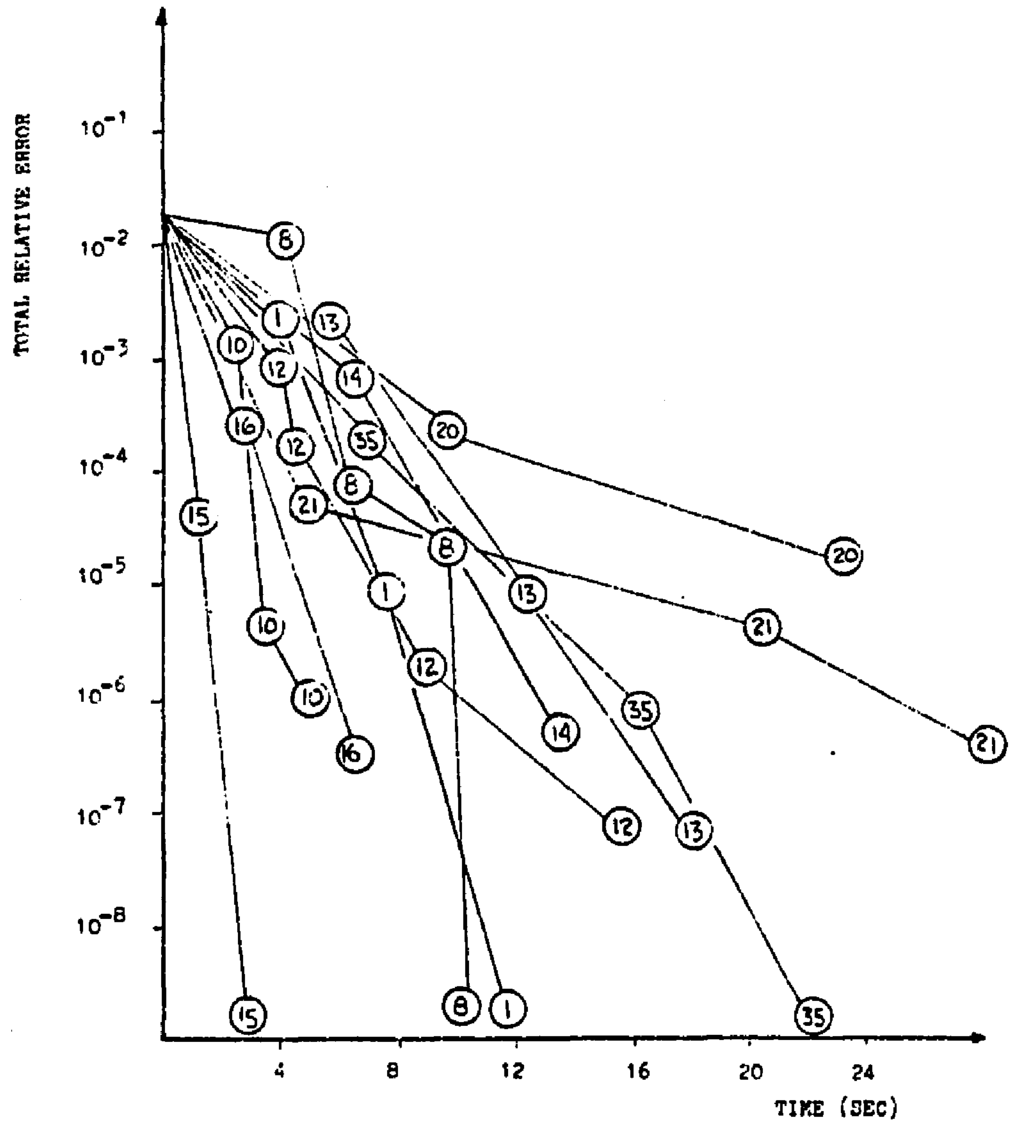

Figure C.11   Total Relative Error versus Time for Problem Number 12.

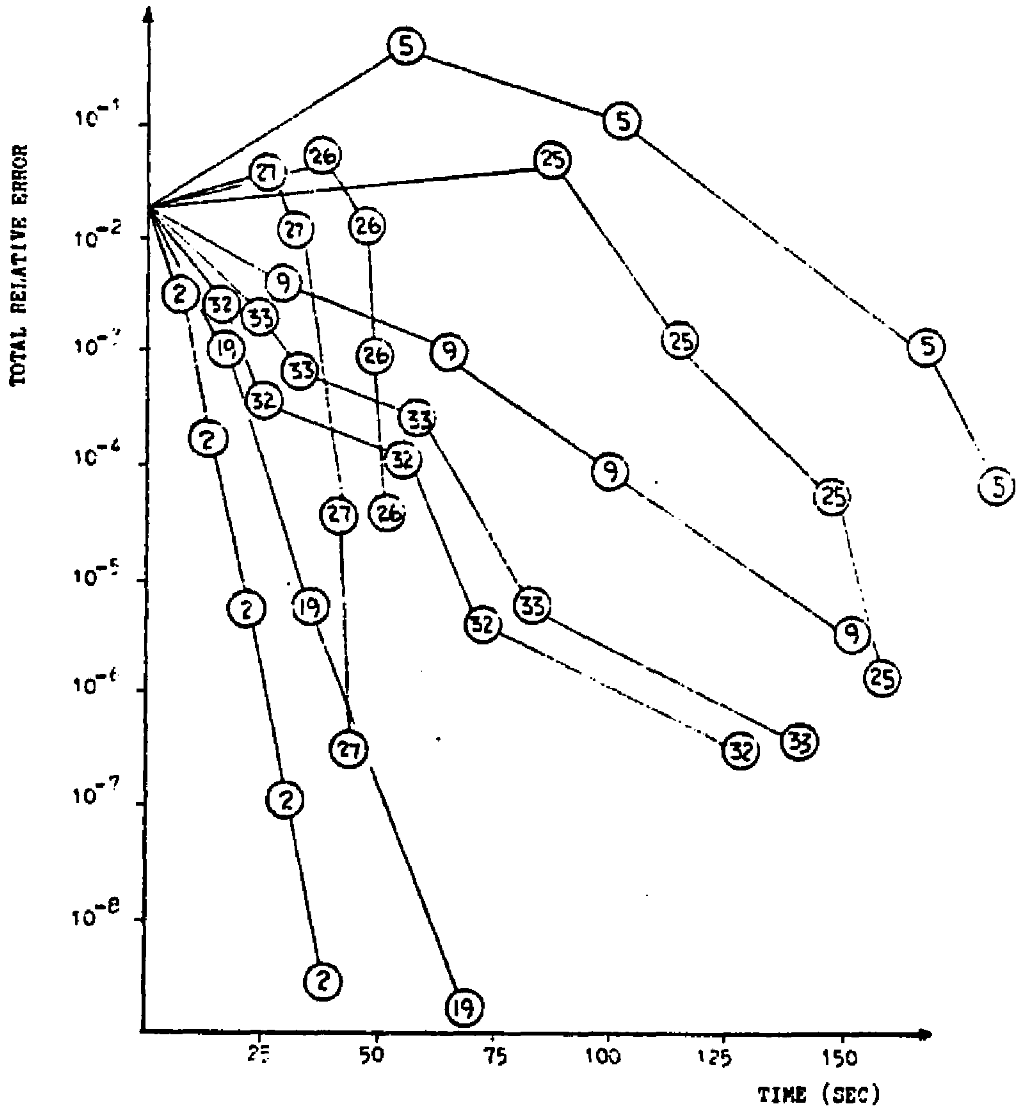

Figure C.11, (Cont'd)

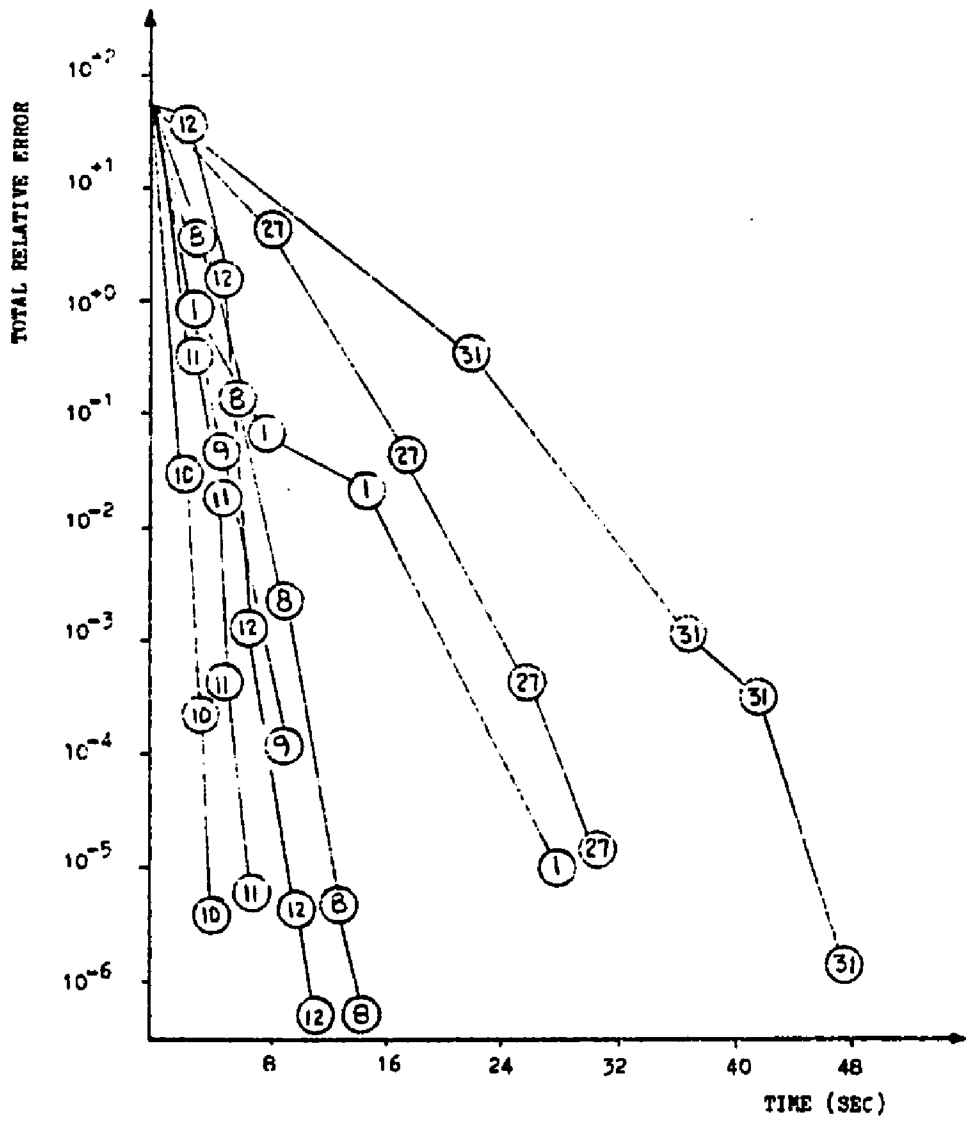

Figure C.12 Total Relative Error versus Time for Problem Number 14.

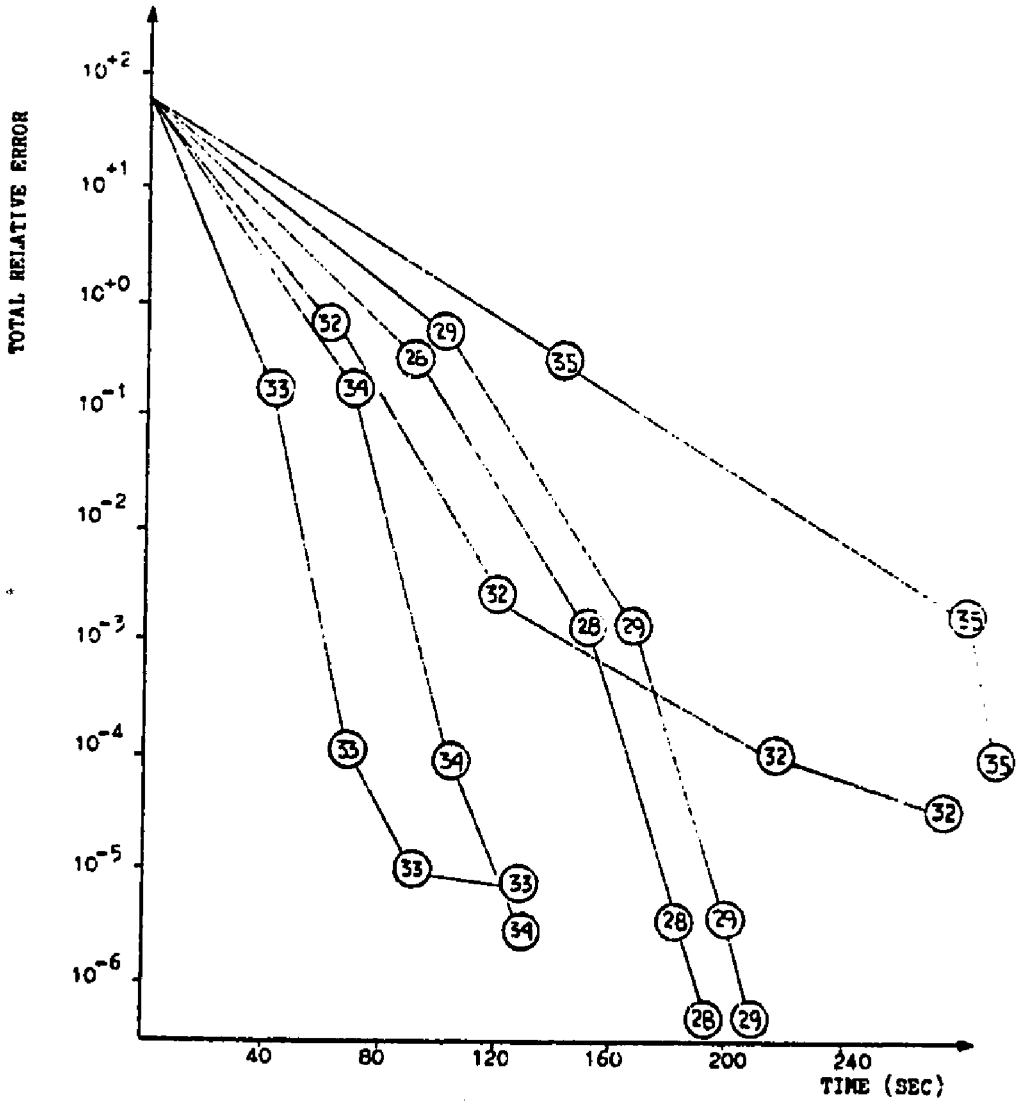

Figure C.12, (Cont'd)

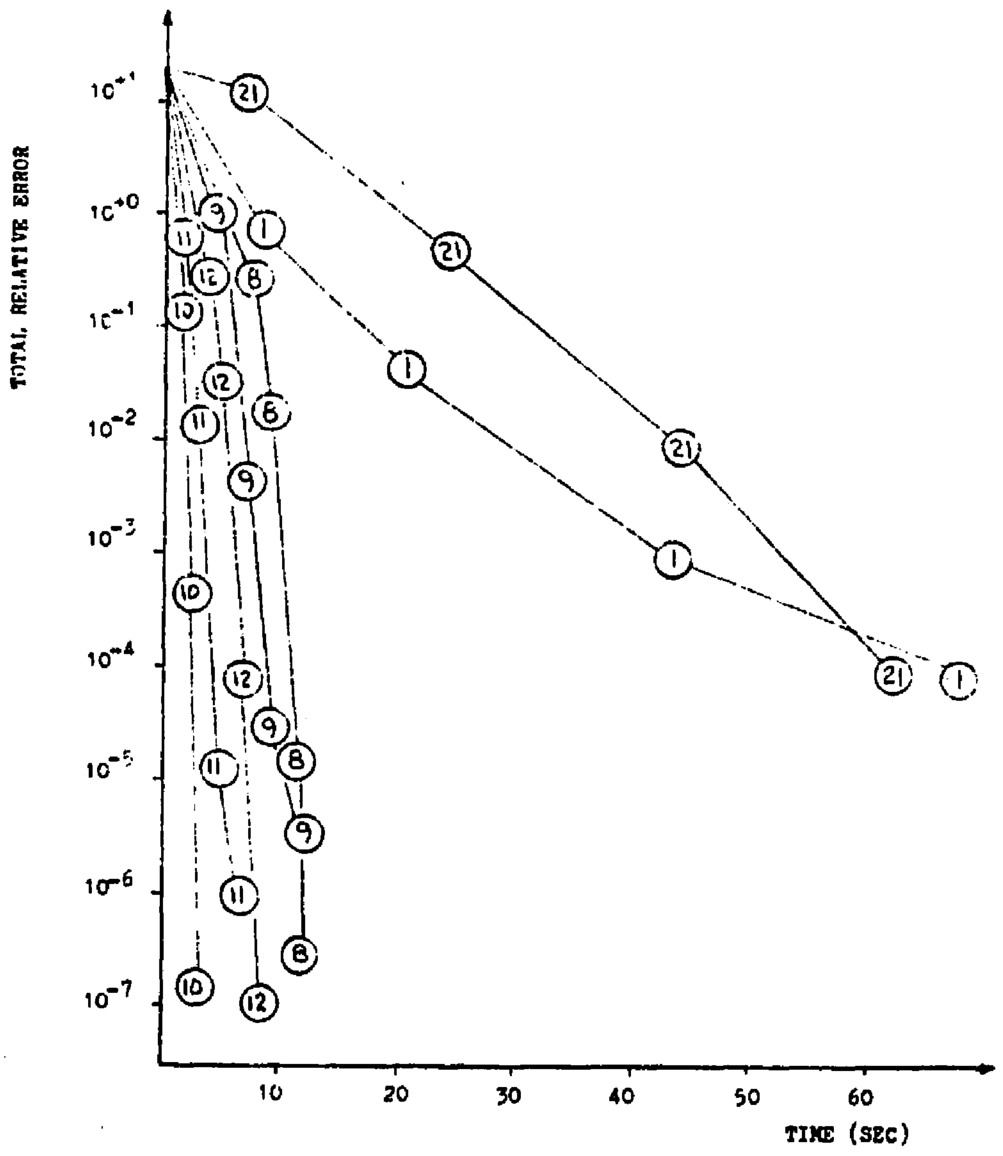

Figure C.13 Total Relative Error versus Time for Problem Number 15.

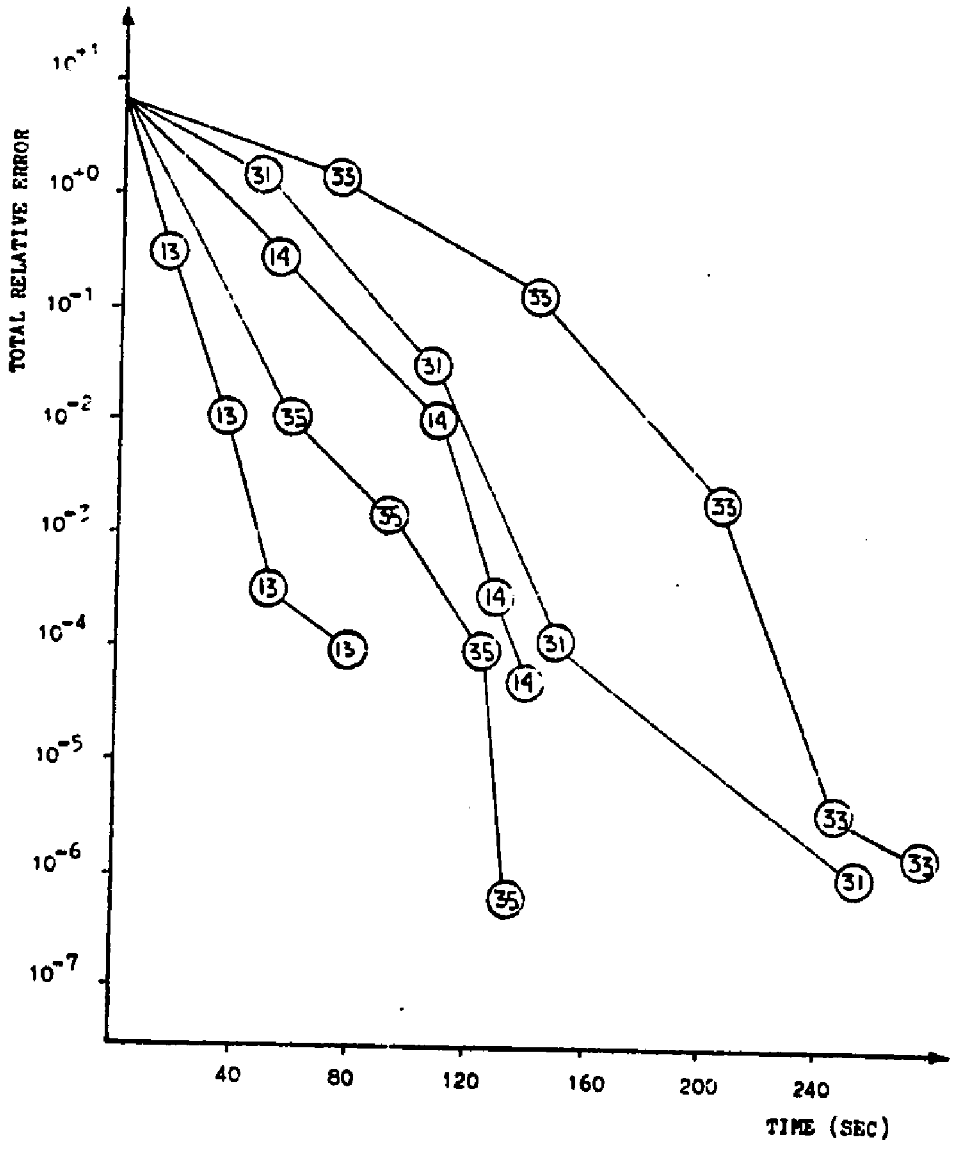

Figure C.13, (Cont'd)

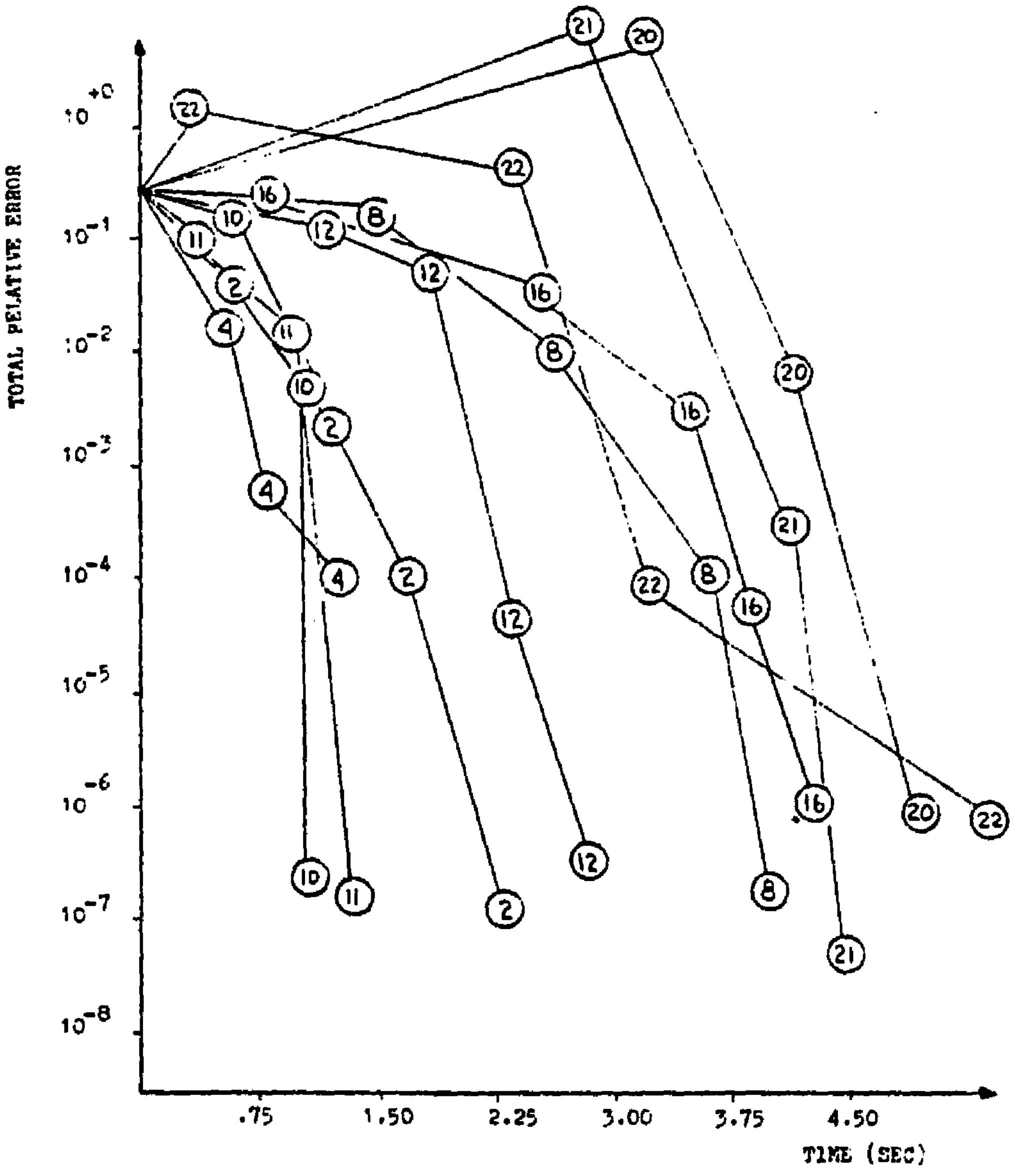

Figure C.14 Total Relative Error versus Time for Problem Number 16.

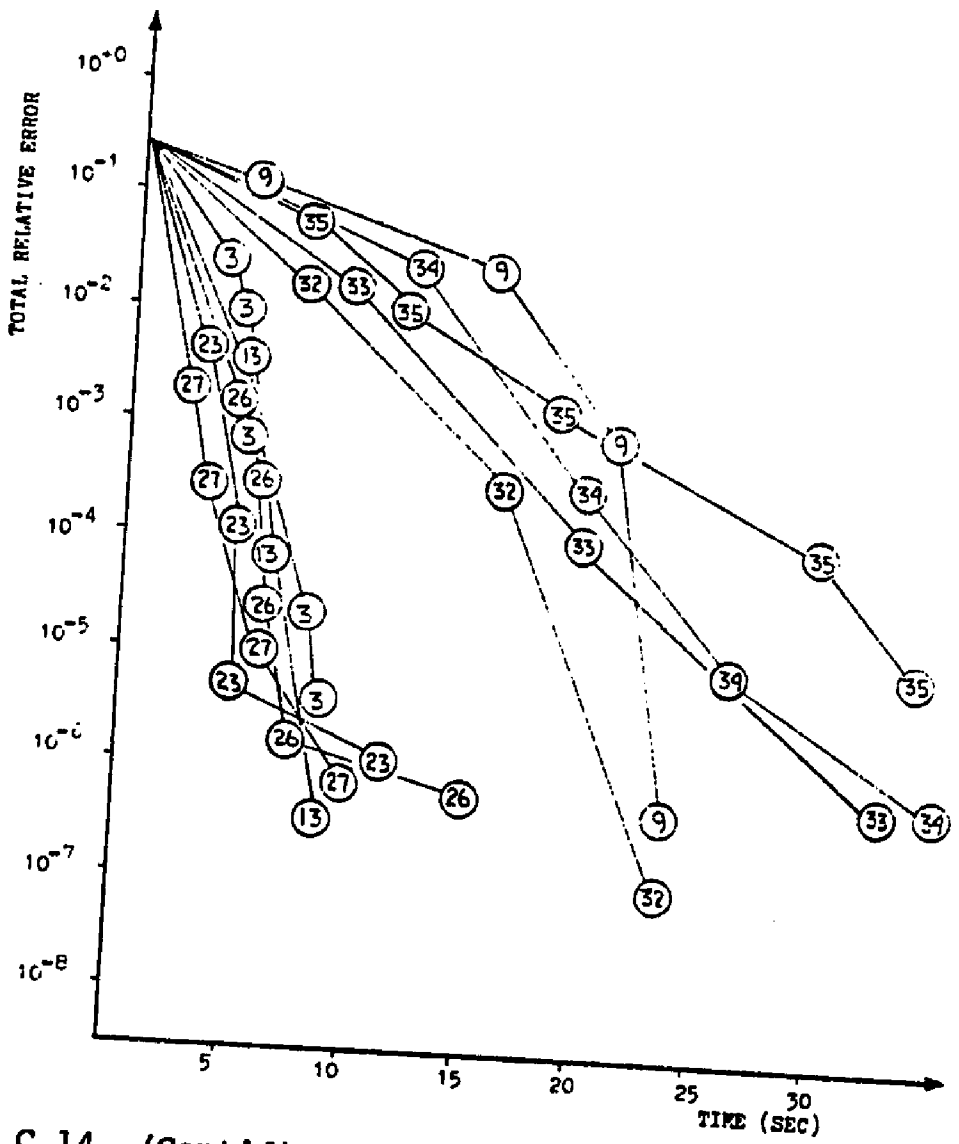

Figure C.14, (Cont'd)

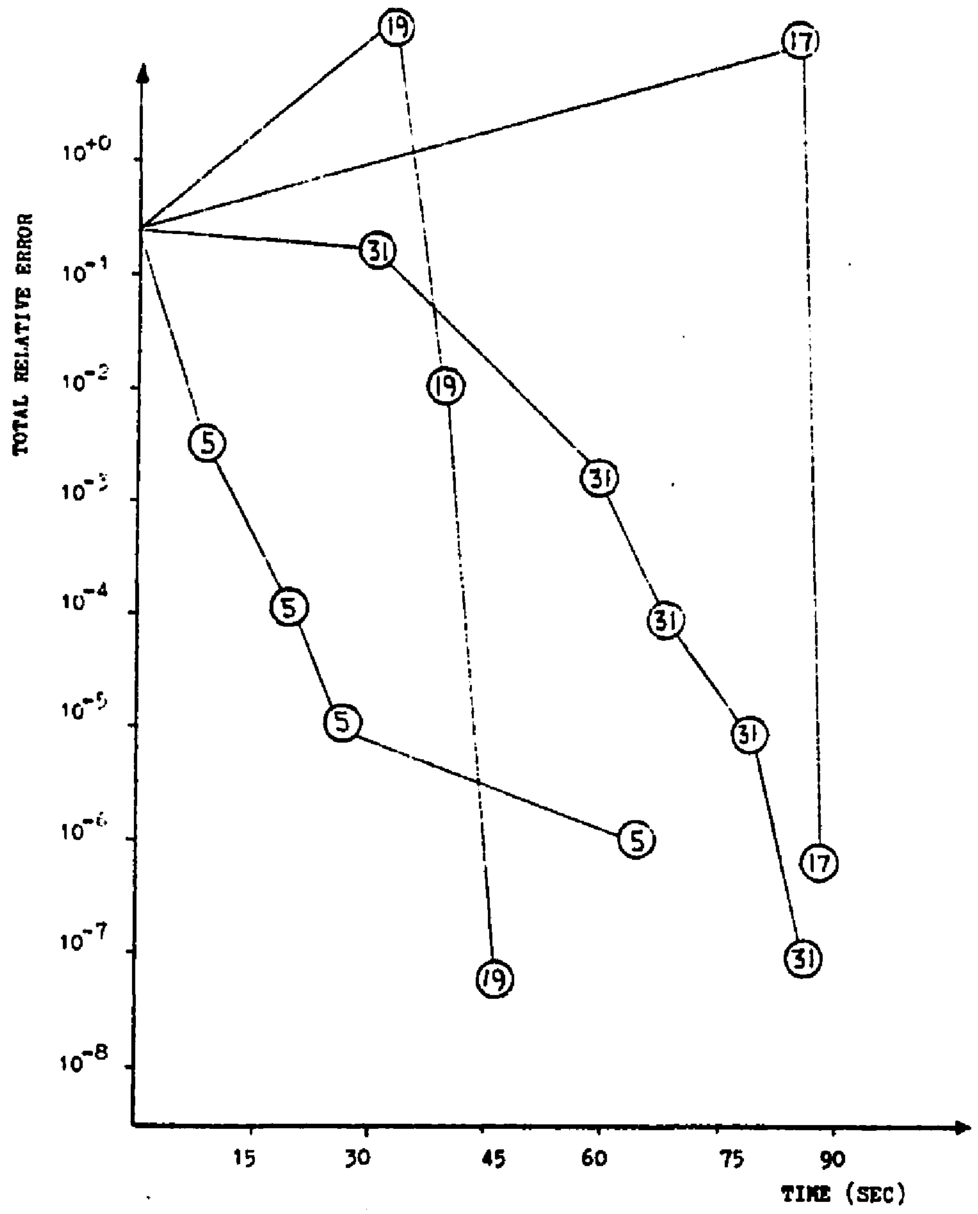

Figure C.14, (Cont'd)

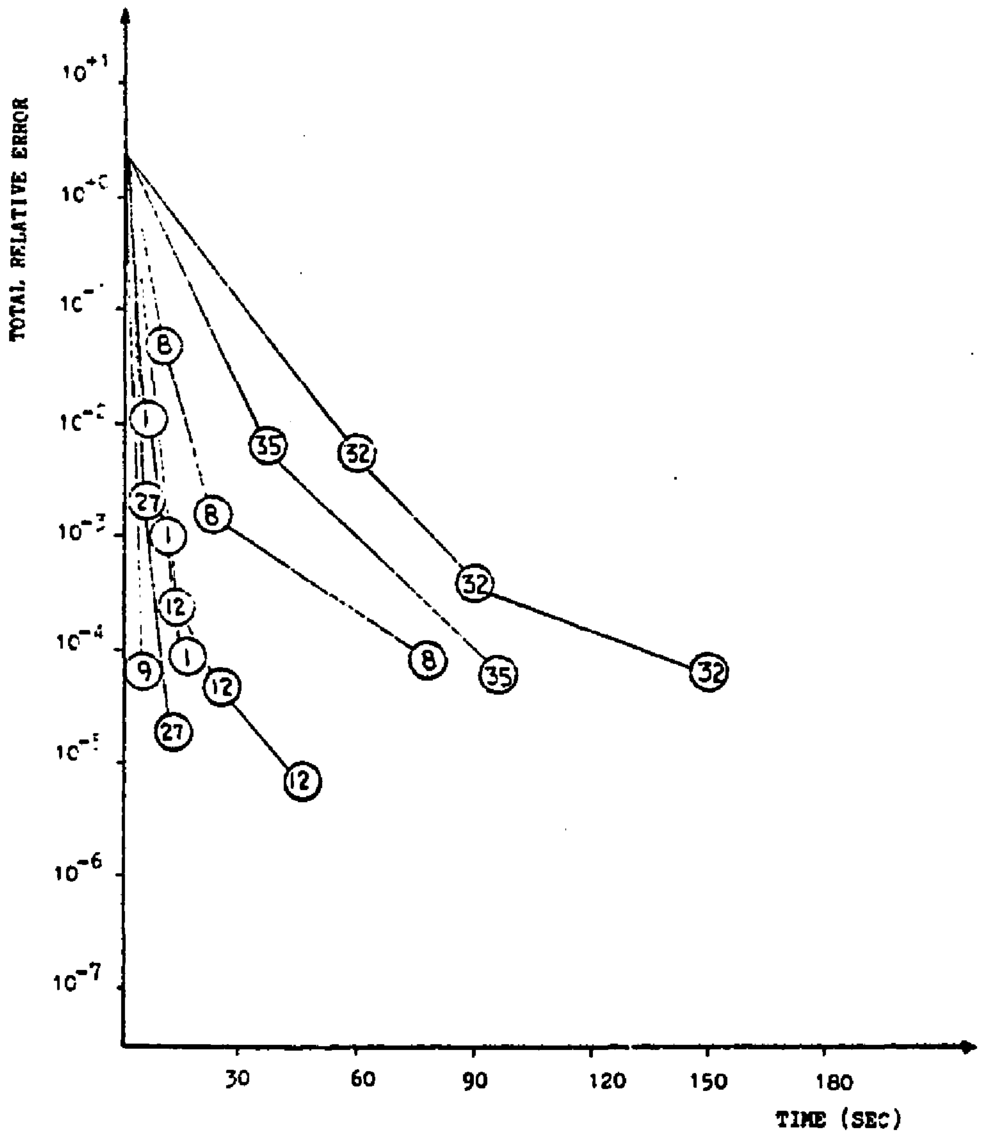

Figure C.15 Total Relative Error versus Time for Problem Number 17.

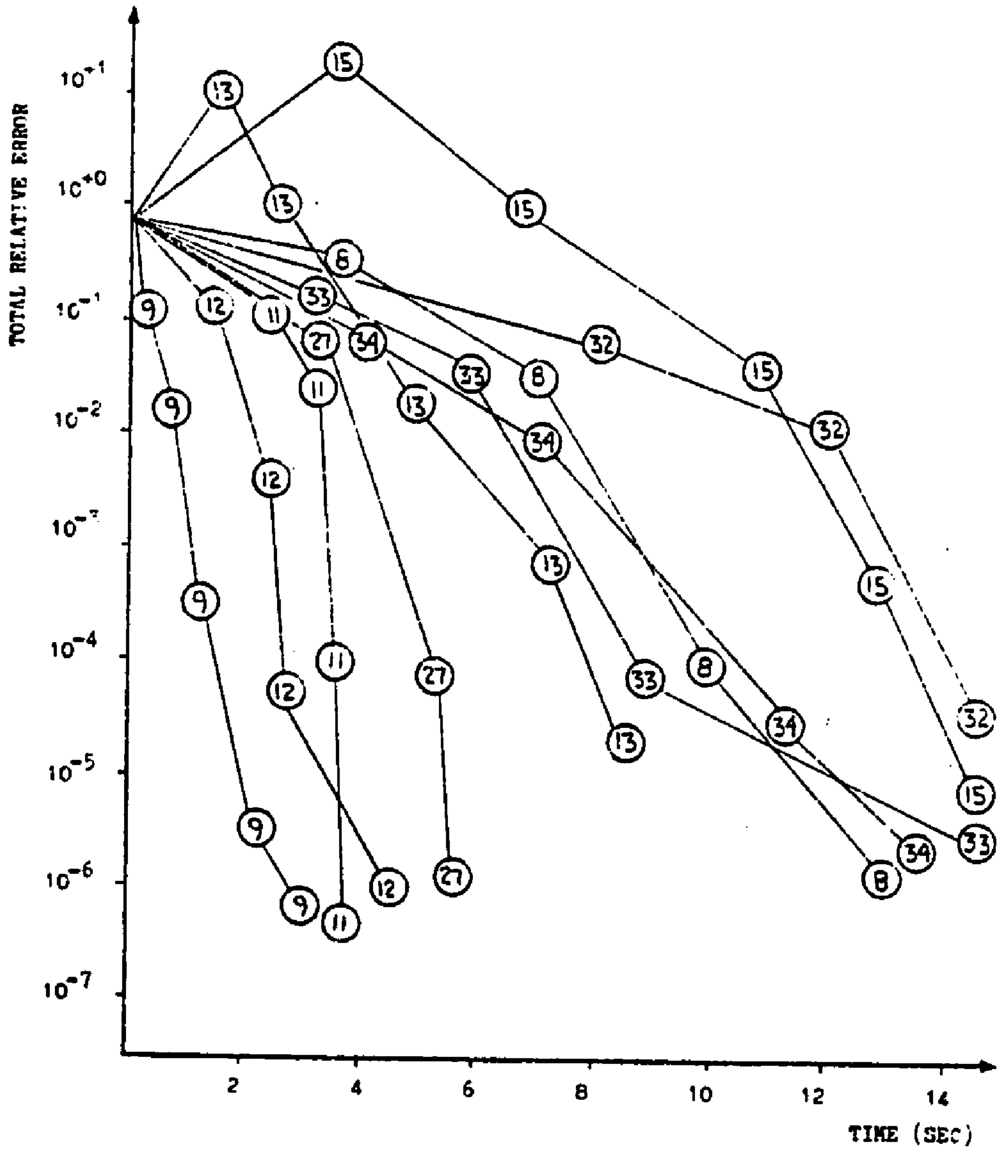

Figure C.16 Total Relative Error versus Time for Problem Number 18.

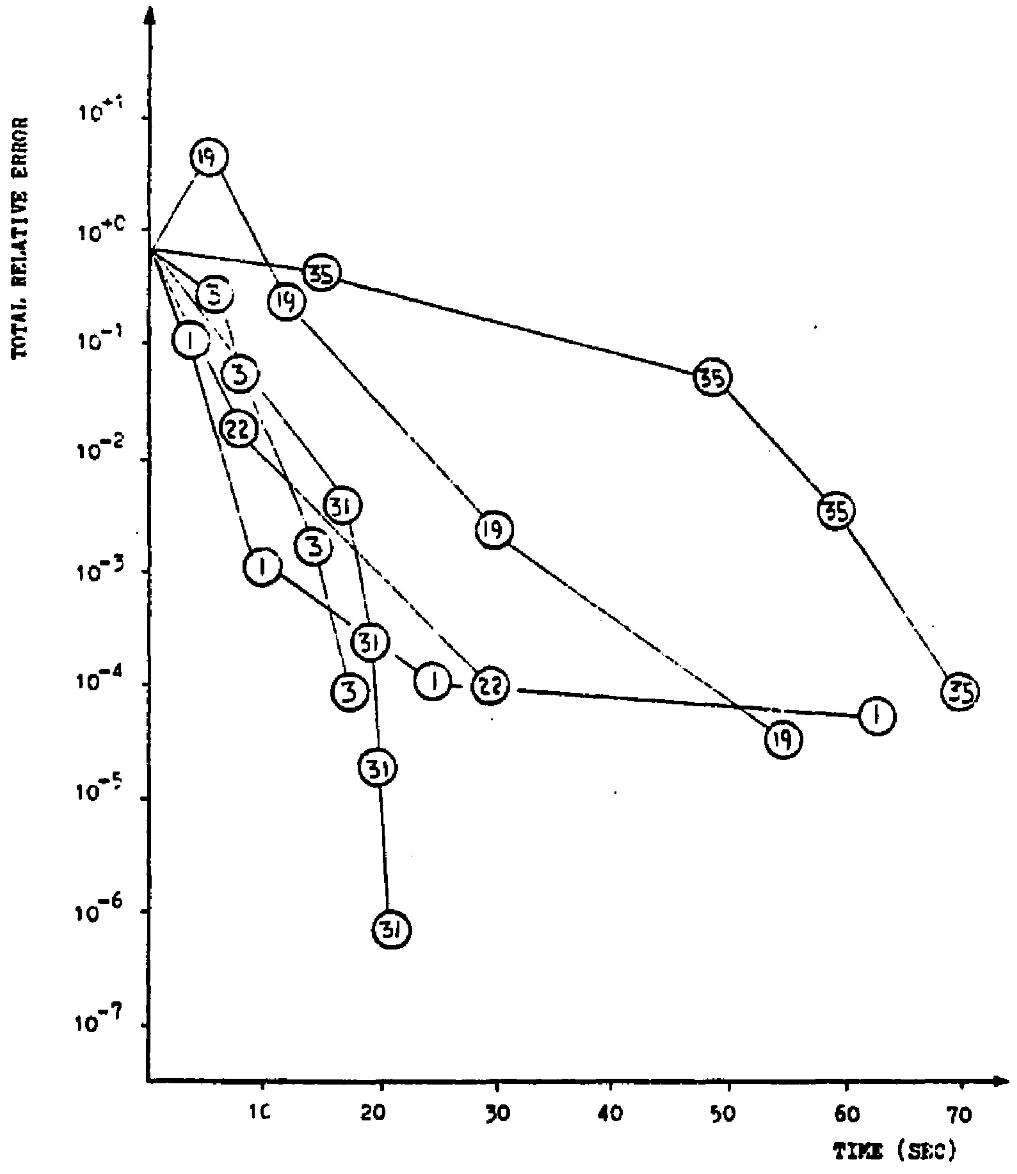

Figure C.16, (Cont'd)

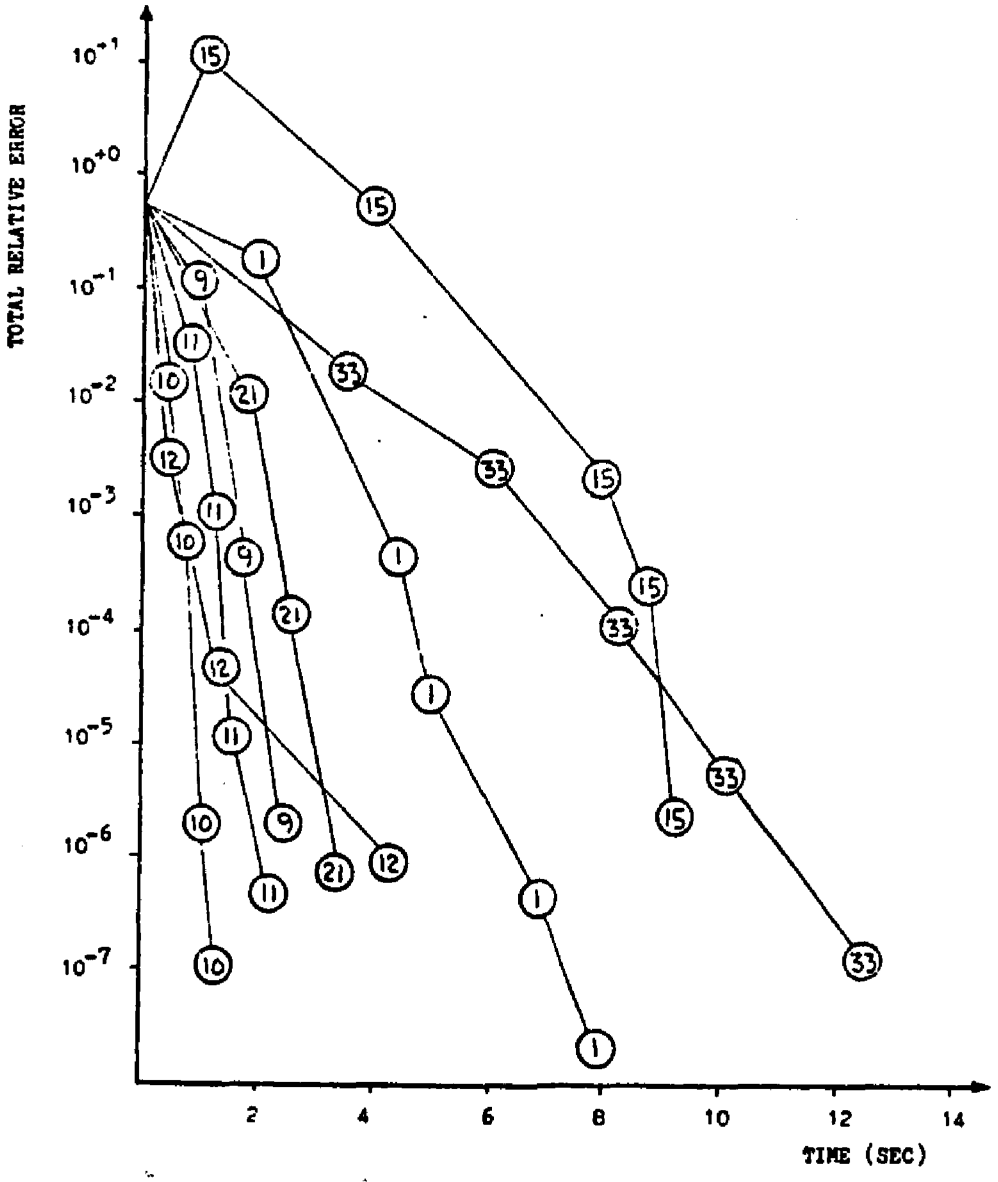

Figure C.17 Total Relative Error versus Time for Problem Number 19.

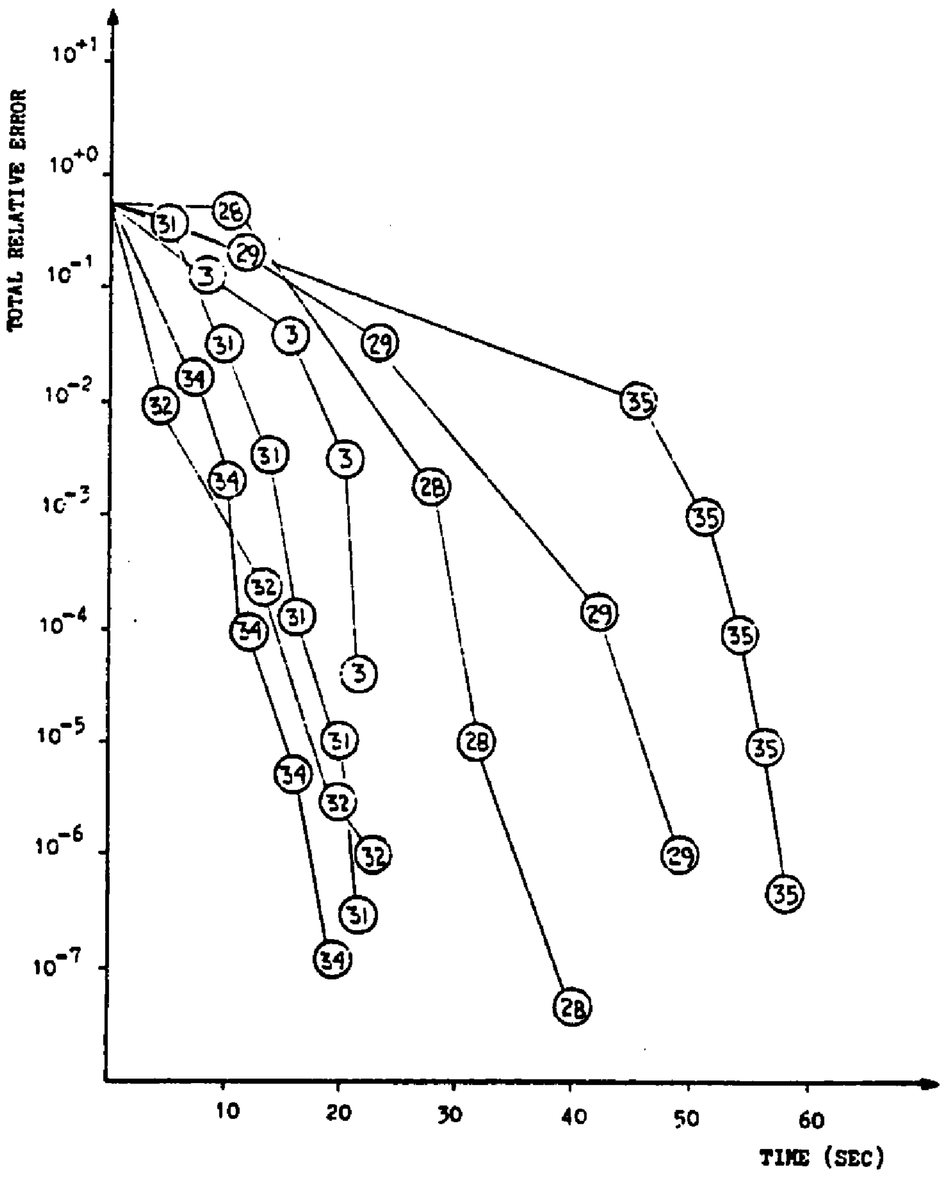

Figure C.17, (Cont'd)

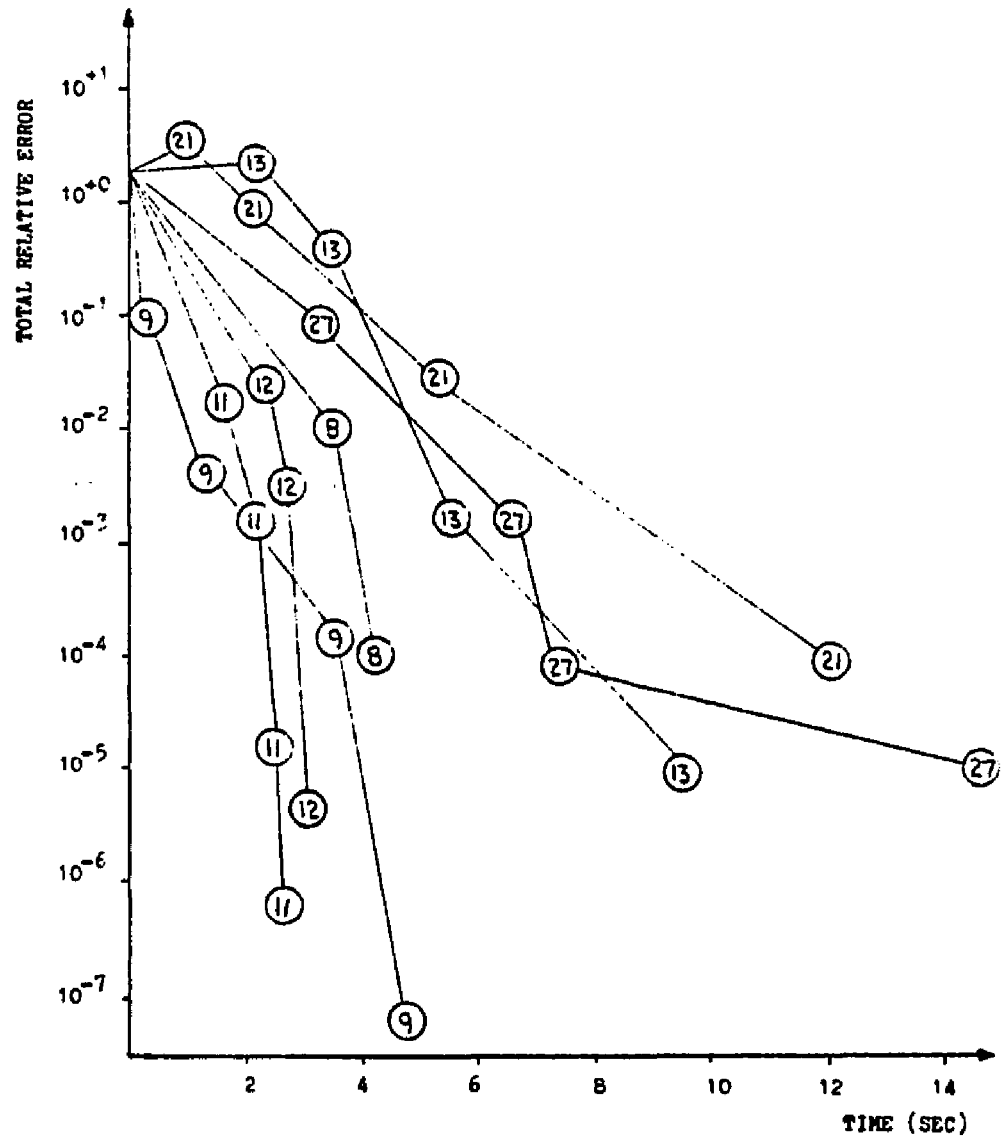

Figure C.18 Total Relative Error versus Time for Problem Number 20.

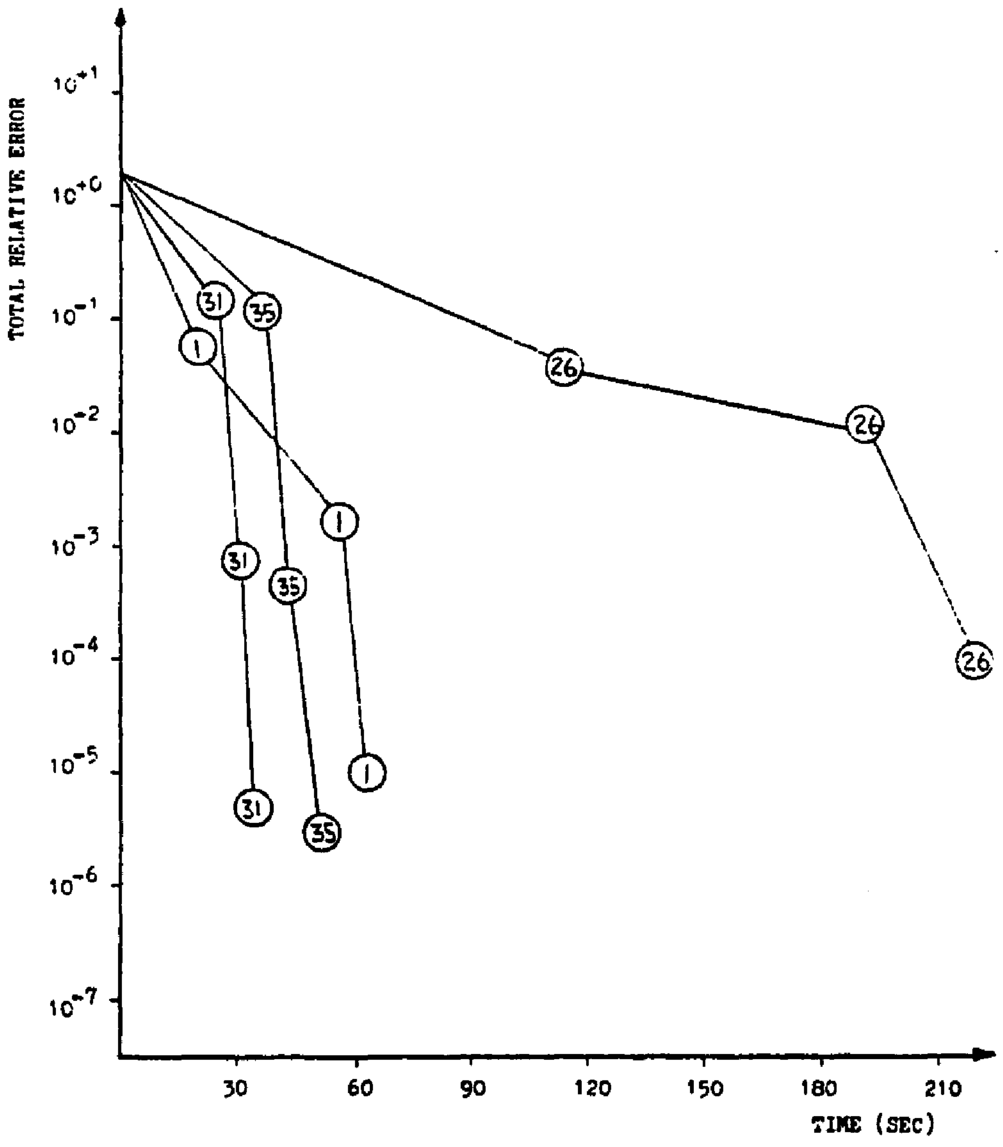

Figure C.18, (Cont'd)

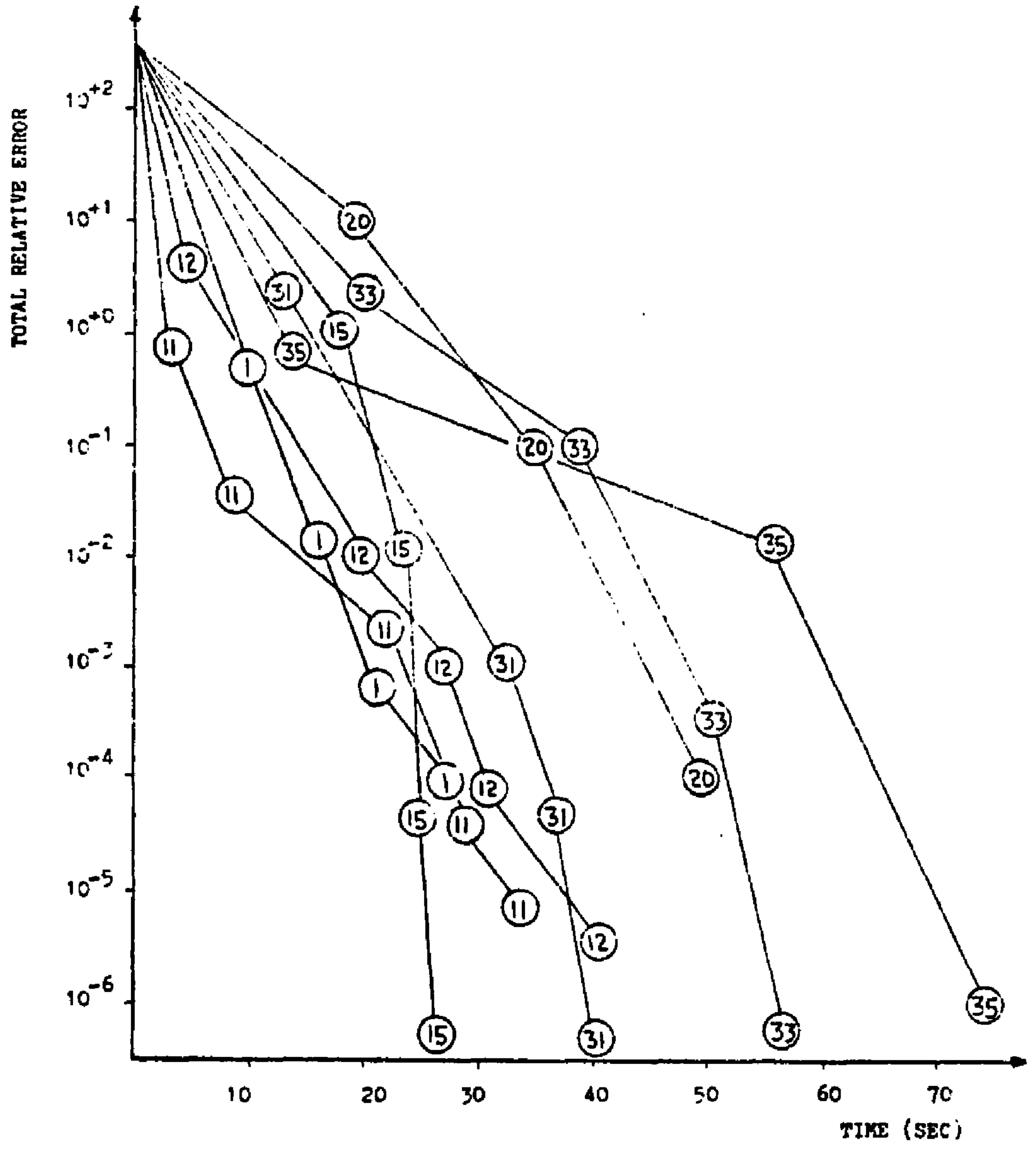

Figure C.19 Total Relative Error versus Time for Problem Number 23.

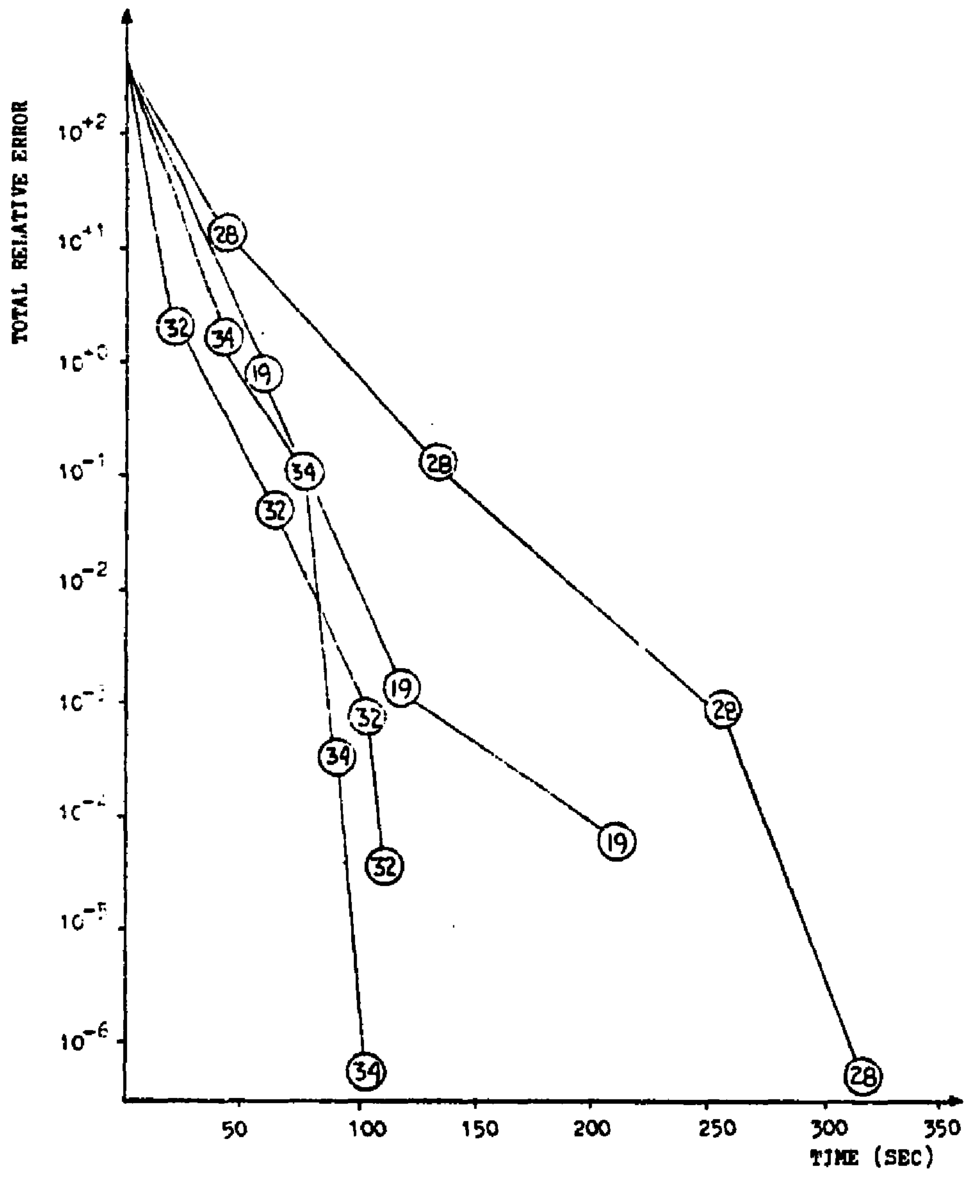

Figure C.19, (Cont'd)

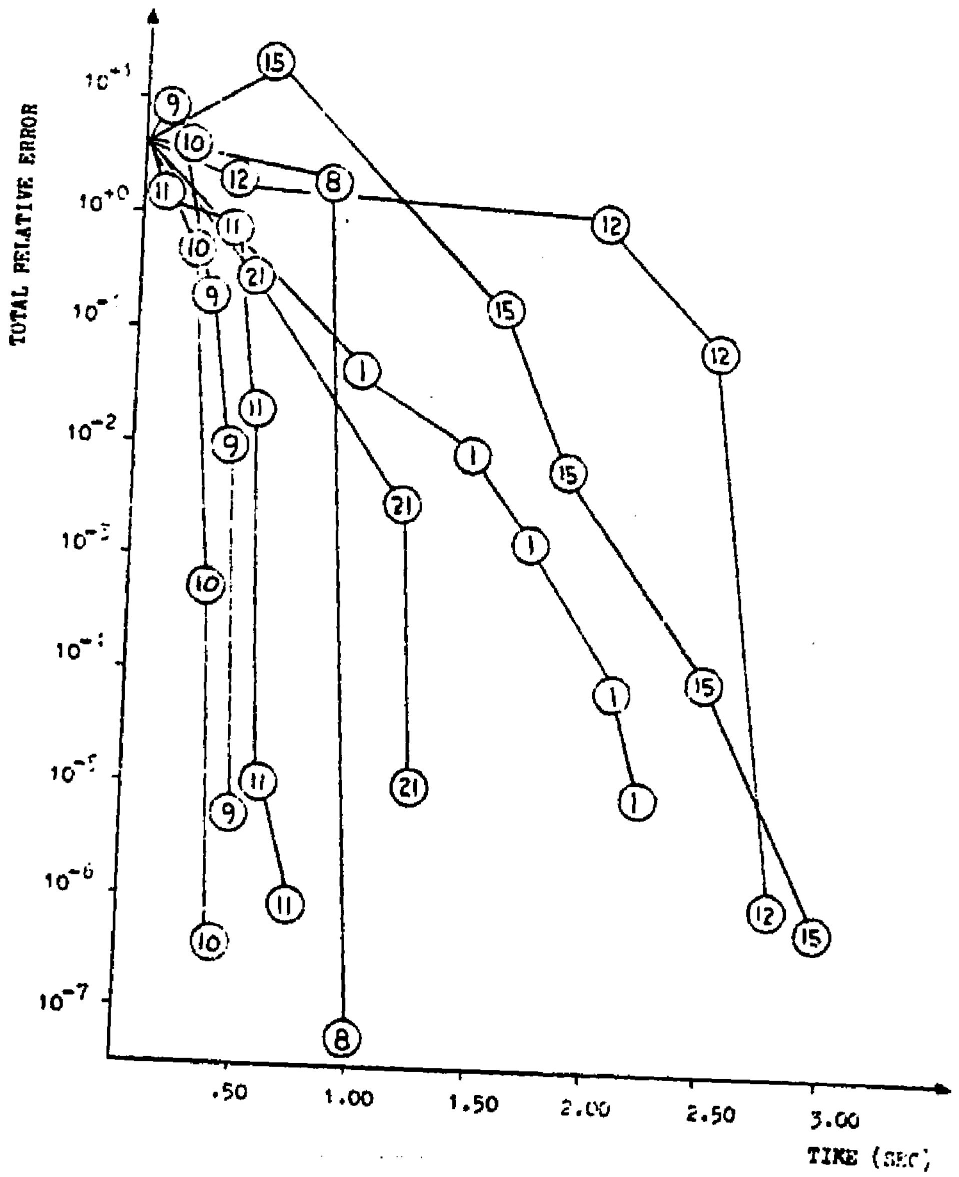

Figure C.20 Total Relative Error versus Time for Problem Number 24.

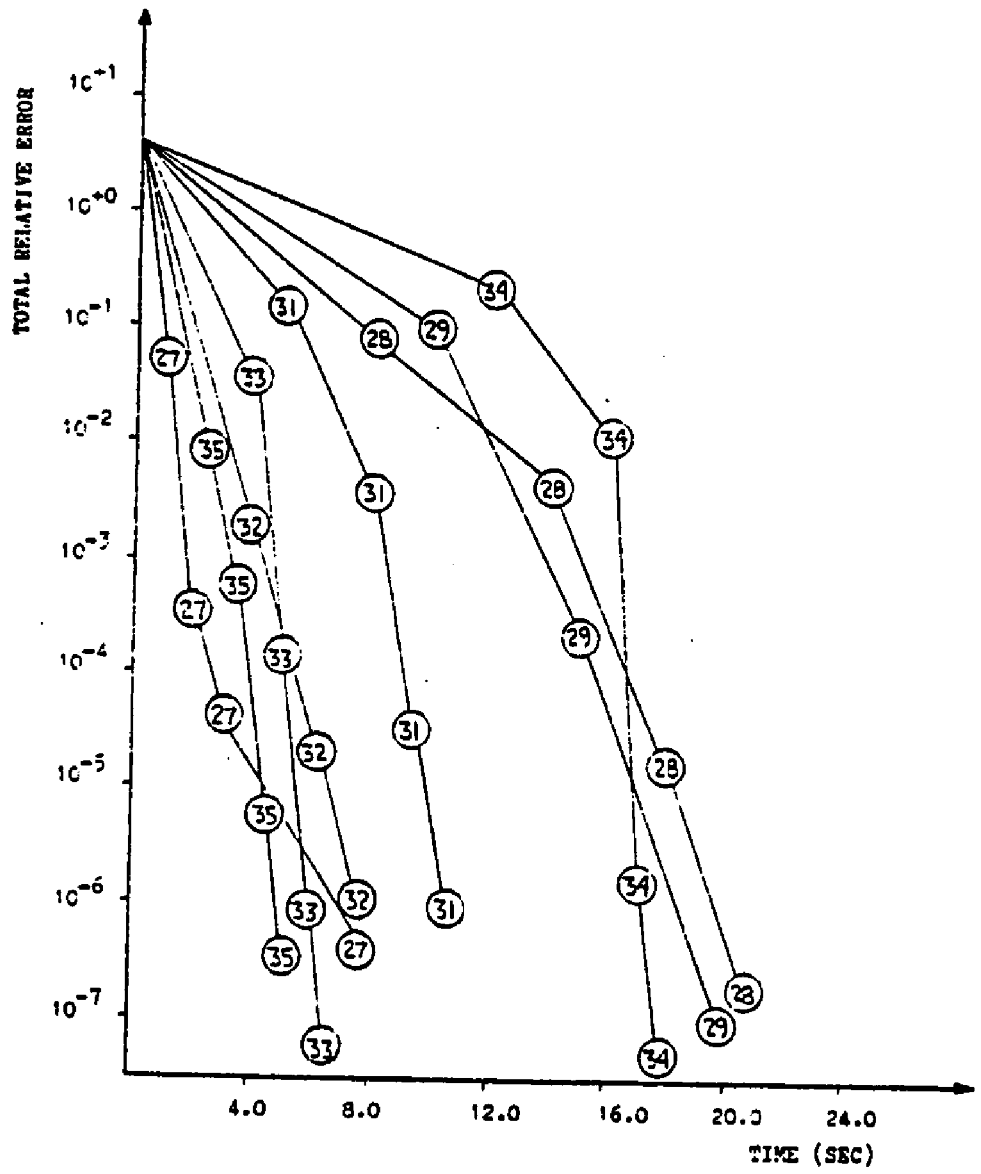

Figure C.20, (Cont'd)

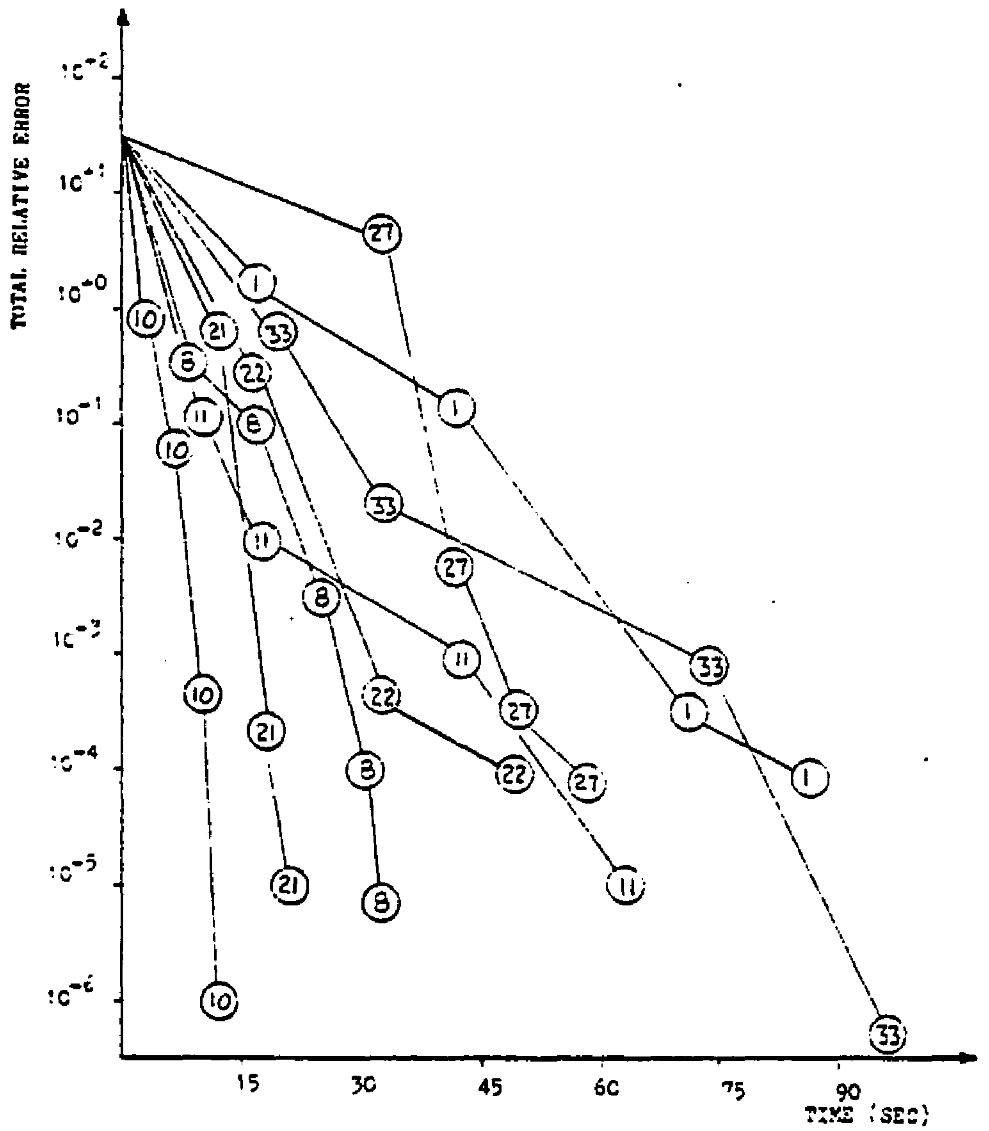

Figure C.21 Total Relative Error versus Time for Problem Number 25.

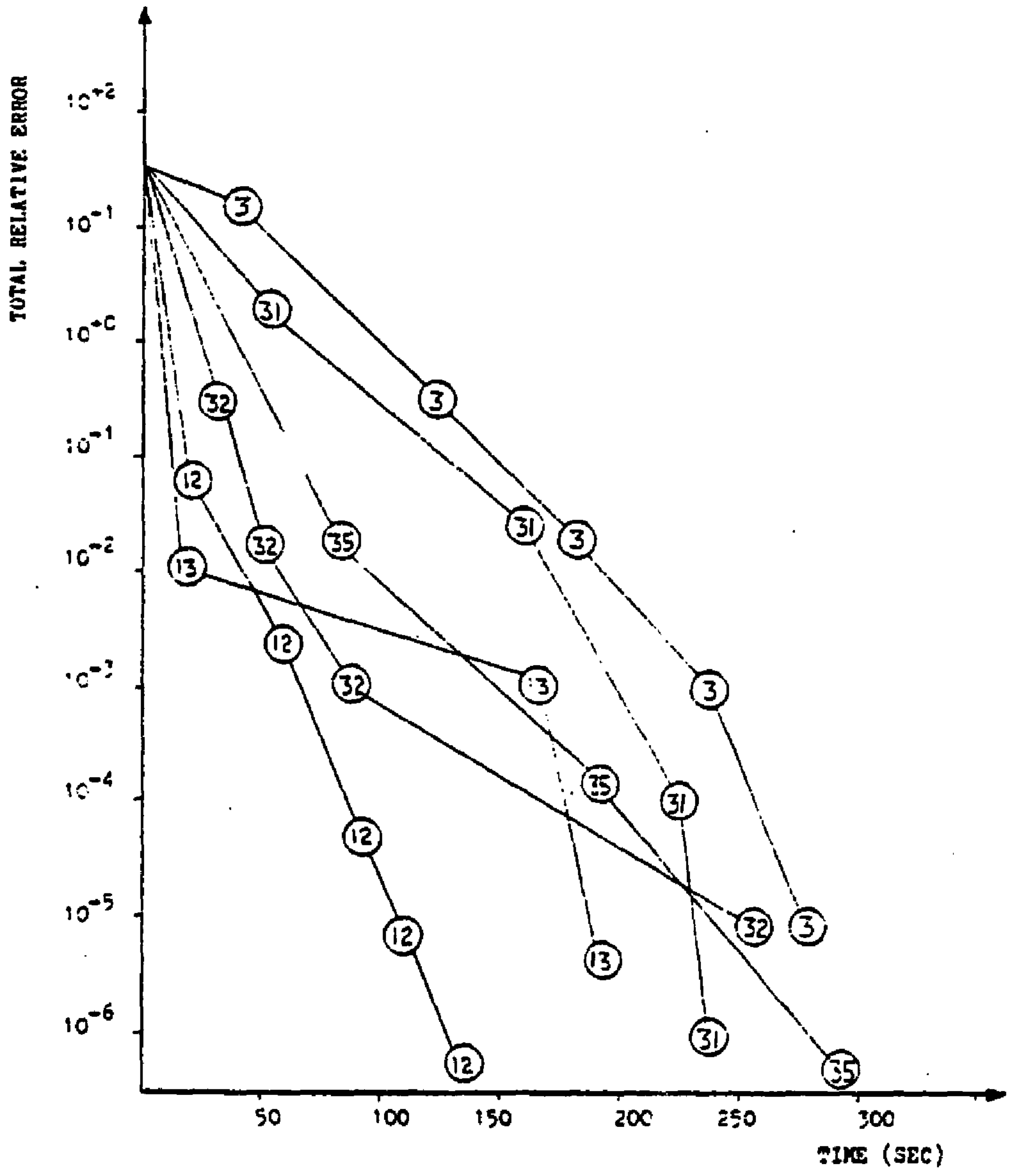

Figure C.21, (Cont'd)

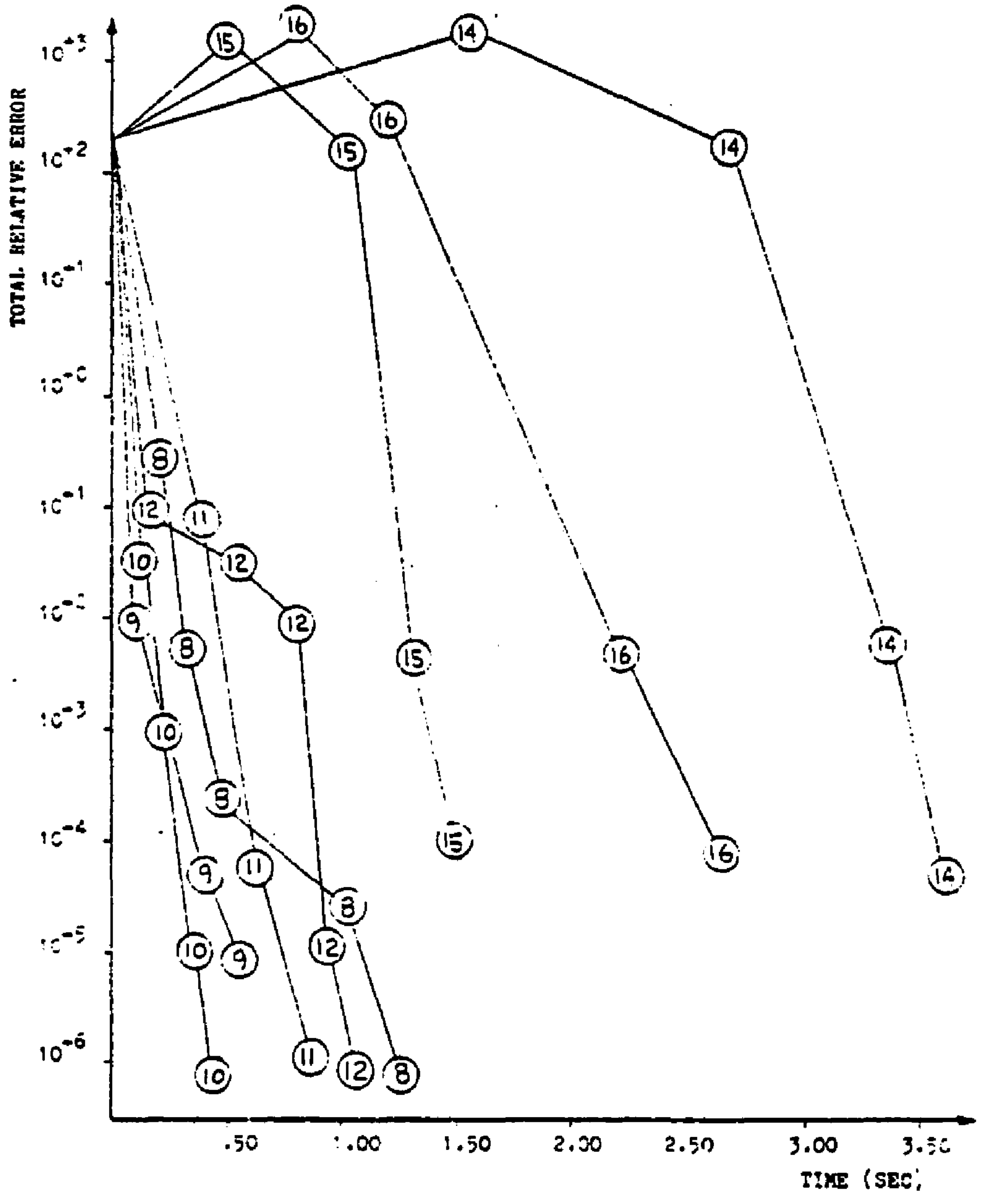

Figure C.22 Total Relative Error versus Time for Problem Number 26.

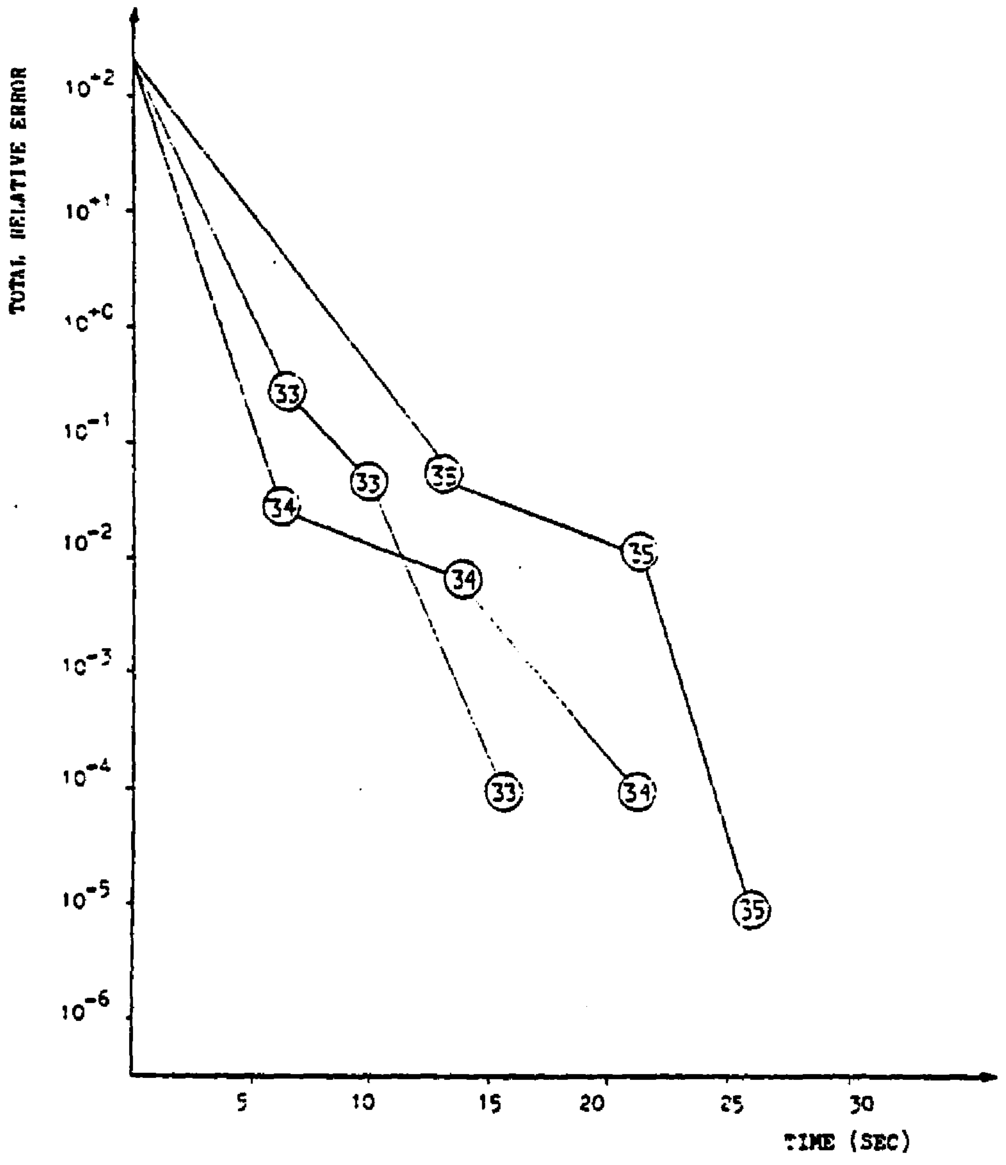

Figure C.22, (Cont'd)

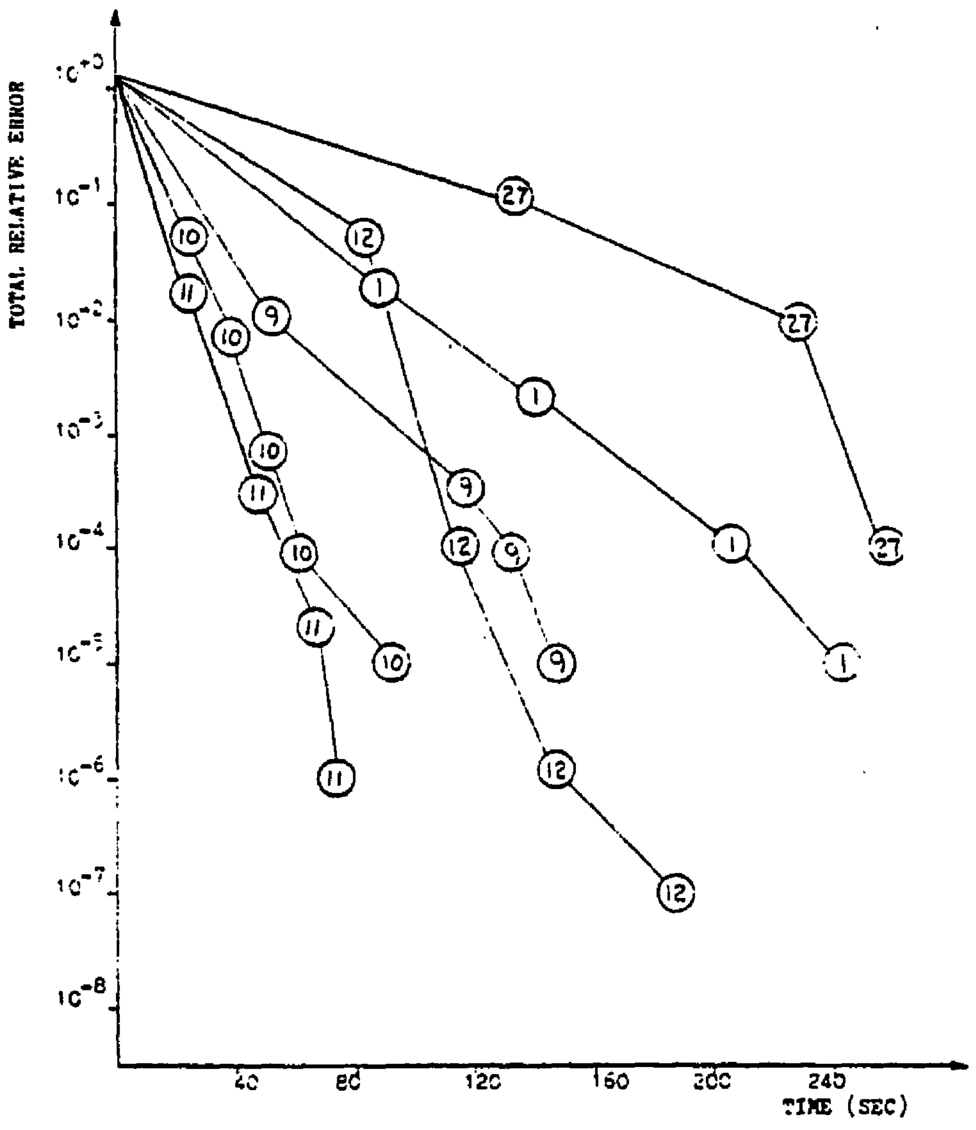

Figure C.23. Total Relative Error versus Time for Problem Number 27.

VITA

VITA

Eric Sandgren was born in Oak Park, Illinois on August 16, 1951, the son of Find and Jytte Sandgren. Upon completion of his studies at East Tipp High School, Tippecanoe County, Indiana in 1969, he enrolled in the School of Mechanical Engineering at Purdue Univesrity in West Lafayette, Indiana where he received his Bachelor of Science Degree in Mechanical Engineering in May, 1973. In the summer of 1973, Eric began graduate study in Mechanical Engineering at Purdue University at the Herrick Laboratories under the direction of Co-Major Professors K. M. Ragsdell and W. Soedel. In December, 1974, Eric received his Master of Science Degree in Mechanical Engineering. Upon completion of his Masters Degree, Eric continued his graduate work under K. M. Ragsdell in the Department of Mechanical Engineering at Purdue University in the area of optimal design. It was during this phase of his life that Eric was married to Mary Elizabeth Ober on June 7, 1975. The couple will reside in Lexington, Kentucky where Eric has accepted a job with I.B.M.