A COMPARATIVE STUDY OF NONLINEAR PROGRAMMING ROUTINES ON THE MICROCOMPUTER VERSUS THE LARGE COMPUTER

by

Frank P. Hwang

B.S., Kansas State University
Manhattan, Kansas 1981

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1984

Approved by:

Ching-Lai Hwang

Major Professor

A11202 961253

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I would like to thank by father, Dr. C.L. Hwang, who was by major advisor, for continuous guidance and invaluable advice in planning the study and preparing this thesis. I would also like to thank Dr. Stanley Lee for his suggestions and Dr. Doris Grosh, Dr. Do Sup Chung and Dr. Frank Tillman for serving on my advisory committee.

Finally, I wish to thank by parents, sisters, and friends for their continuous encouragement.

CHAPTER 1

INTRODUCTION

1.1 HISTORY

The Hooke and Jeeves Pattern Search technique is used to find the local minimum of a multivariable, unconstrained, nonlinear function. The procedure is based on the direct search method proposed by R. Hooke and T.A. Jeeves [9]. At Kansas State University, the method was programmed in Fortran for the campus mainframe computer by S. Kumar [11] in 1969.

The sequential unconstrained minimization technique (SUMT) is used to find a solution to a nonlinear programming problem with nonlinear inequality and/or equality constraints. The basic scheme of this technique is that a constrained minimization problem is transformed into a sequence of unconstrained minimization problems which can be solved by any of the available unconstrained minimization techniques. The SUMT technique was originally proposed by C.W. Carroll [1,2] in 1959 and further developed by A.V. Fiacco and G.P. McCormick [3,4,5,6,7] in 1964.

At KSU, a computer program was written which uses a modified Hooke and Jeeves pattern search technique as the unconstrained minimization technique for use in the SUMT method. This program was written in Fortran for the large computer by K.C. Lai in 1970 as part of his master's thesis [10,12,13].

Also at KSU, S.V. Gopalakrishna wrote a computer program using a conjugate gradient method as the unconstrained minimization technique for use in the SUMT method in 1971 [8]. However, the results obtained from the program were not good so the program was never used.

In 1964, at the Research Analysis Corporation, a computer program was written in Fortran by G.P. McCormick, W.C. Mylander III, and A.V. Fiacco using a second order gradient method to determine the direction of search and the Fibonacci Search method to determine the optimum step size. The program was entitled "RAC Computer Program Implementing the Sequential Unconstrained Minimization Technique for Nonlinear Programming" (RAC-SUMT) and its share number is 3189 [14]. The program could not handle equality constraints however. A later version of the program, version 4, written in 1971 was able to handle equality constraints and in addition, three more methods were added to be used in determining the direction of search : a conjugate gradient method, a first order gradient method, and a revised version of the second order method used in version 1 [15]. The method used to determine the optimum step size was also changed to the Golden Section Method.

The first version of the RAC-SUMT computer program was checked and modified by F.T. Hsu [16] so that it would run on the computer at KSU in 1969. Version 4 of the RAC-SUMT computer program had not yet been tried here.

1.2 ADVANTAGES OF MICRO/PERSONAL COMPUTER OVER LARGE COMPUTER

There are a few major advantages which the micro/personal computer has over the large computer which make it attractive to use. One of the major advantages of the micro/personal computer over the large computer is the easy accessibility of the micro/personal computer. One reason why the microcomputer is easily accessible is because there is no need to have a security number or computer funds to operate the micrc/personal computer as there is for the large computer. Another reason is that there is no need to wait for a terminal or card punch to become available. A third reason is that there is no restriction on the hours when the micro/personal computer may be used as there is for the large computer. These reasons make the micro/personal computer more easily accessible than the large computer.

Another major advantage of micro/personal computers over the large computer is cost. The cost of a micro/personal computer is now at a price where many middle and upper class families can purchase one. In addition to the purchase price being low, the operating cost is also low because there is no need for a staff of computer personnel to keep the micro/personal computer running as there is for the large computer. There is also no charge for using the micro/personal computer as there is for the large computer.

A third reason for using micro/personal computers as opposed to large computers is because of the adequate capability of available micros to handle many types of problems. The capability of the micro/personal computer has improved greatly over the last few years and many of the limitations which once restricted the types of problems that could be solved on a microcomputer no longer exist.

For example, although microcomputers were once limited to a maximum memory size of 64K (North Star Horizon), now they can be expanded up to 640K bytes (IBM PC). See Table 1.1 for a comparison of the features of the two machines. The increase in memory size allows larger programs to be run on the microcomputer and also increases the size of problems which the programs can solve.

Table 1.1 Features of the North Star Horizon and IBM PC

North Star Horizon
CPU : Z80A, 8 bit
Memory : 64K (not expandable)
Operating system : CP/M, North Star DOS
Storage : 360K per 5 1/4 inch floppy disk double sided, double density
IBM PC
CPU : 8088, 16 bit
Memory : 64K (expandable to 640K)
Operating System : PC-DOS
Storage : 360K per 5 1/4 - inch floppy disk double sided, double density

1.3 LANGUAGE AND COMPUTER USED IN STUDY

All of the programs used in this study were written in Fortran and developed using a North Star Horizon II microcomputer which has a Z80A CPU. The operating system used was the Lifeboat 2.21A version of CP/M. The source programs were written using Micro Pro's WordStar version 2.26 and compiled with Microsoft's Fortran-80, 1980 version for the North Star microcomputer. The version of Fortran includes the American National Standard Fortran language as described in ANSI document X3.9--1966, approved on March 7, 1966, plus a number of language extensions and some restrictions. Of these extensions, the ones which were used in the programs were:

1. The literal form of Hollerith data (character string between apostrophe characters) is permitted in place of the standard nH form.

2. Mixed mode expressions and assignments are allowed, and conversions are done automatically.

1.4 THE OBJECTIVES OF THIS STUDY

The objectives of this study are as follows. First, a study was needed to determine the feasibility or practicality of putting the nonlinear programming programs into the microcomputer. When this study first started, only a North Star Horizon microcomputer was available which was limited to 64K bytes of memory. Because of the limited memory of this microcomputer and many others, it was not known whether the programs would fit into the available memory. Also because of its slower speed it was not known whether the programs would be practical to run on the microcomputer.

A second objective was to do a comparative study of the nonlinear programming routines on the large computer versus the microcomputer in terms of ease of use, accuracy of results, size of problem, and total time needed to prepare and run a problem including the time needed to enter data into the terminal, wait for results, etc.

A third objective concerned the checking of the programs. Over a period of 12 years, the Hooke and Jeeves pattern search program, the KSU-SUMT program and the RAC-SUMT program have been used for research at KSU. Many students have made minor changes to the programs but there has been no systematic checking of the logic of the changes made to the programs. In this study, a third objective was to systematically check, modify, and correct the complete programs including any modifications made to them.

A fourth objective is to prepare the programs and the complete documentation of the programs so that they can be used for educational purposes. Included in the documentation is the introduction of the theory behind the techniques used in the programs, numerical examples to illustrate the techniques, the description of the input to the program and how to use the programs, the output from the programs, and a description of the program. The preparation of the programs included making the programs as readable and understandable as possible, restructuring the program if necessary. An input routine also needed to be written for each program to allow input to be entered from the keyboard in an interactive manner.

A fifth objective was to test version 4 of the RAC-SUMT program on the microcomputer. Although version 1 of the RAC-SUMT program had been checked and used at KSU, version 4 had not yet been checked or tested here.

1.5 WHAT HAS BEEN DONE IN THE MS THESIS

The first objective of this study was to determine the feasibility or practicality of putting the nonlinear programming routines into the microcomputer. From the printout of the program run on the large computer, the amount of core used could give an indication of whether the program might fit into the microcomputer. However, the exact size of core needed on the microcomputer could not be known until it was actually compiled on the microcomputer.

When the Hooke and Jeeves pattern search program and the KSU-SUMT program were compiled, they both fit into the 37K bytes of available memory but when the RAC-SUMT program was compiled, it exceeded the available memory of the microcomputer. However, by placing the input routine into a separate program, the main program fit into memory.

To determine whether the programs would be practical to run on the microcomputer, the length of time it took to solve a problem had to be determined. Originally, the Hooke and Jeeves pattern search program was programmed using double precision arithmetic. However, test problem 2 which had twenty variables was not finished even after one hour of execution time. Thereafter, the Hooke and Jeeves program and the KSU-SUMT and RAC-SUMT programs were converted to single precision. All test problems solved by the single precision version of the programs took less than four minutes of execution time demonstrating that it was practical to solve small to moderate size nonlinear programming problems on the microcomputer.

The second objective was to do a comparative study of the nonlinear programming routines on the large computer versus the microcomputer in terms of ease of use, accuracy of results, size of problem, and total time to prepare and run a problem including the time needed to enter data into the terminal, wait for results, and so forth. To accomplish this objective, a set of criteria was chosen to be used in making the comparison. The set of criteria used was similar to those used in comparing competing techniques on the same computer. The test problems were then run on both the microcomputer and the large computer, and finally, the results were compared.

The third objective was to systematically check, modify, and correct the complete programs including any modifications made to them. In order to accomplish this objective, first the methodology used in the programs had to be understood. Then the details of the program were studied and finally, any corrections or improvements needed were made to the programs. Because of the usual difficulty in understanding programs written by other people, the sections of code which were not fully clear were not changed. A major change made to all three programs was to add an input routine which allowed input to be entered interactively from the terminal.

The fourth objective was to prepare the programs and the complete documentation of the programs so they could be used for educational purposes. Much of the documentation was already written by the people who wrote the original programs. It was necessary though to check and update the documentation. More comments were added to the KSU-SUMT program to make it easier to understand. In addition, the step numbers in the algorithm, flowcharts and the program were matched up.

The fifth objective was to test version 4 of the RAC-SUMT program on the microcomputer. When the main program along with the input routine was entered into the microcomputer, it would not fit into the 37K bytes of available memory of the North Star Horizon microcomputer. However, after placing the input routine into a separate program, the main program would finally fit into memory. A few test problems were then run to test out the program.

1.6 PREFACE TO THE REST OF THE THESIS

In chapter two, the Hooke and Jeeves pattern search technique for unconstrained minimization is presented along with a computer program for it written in Fortran and documentation for the program.

Chapter three presents the KSU-SUMT computer program and the methodology behind the program. The KSU-SUMT technique is implemented using a combination of a modified Hooke and Jeeves pattern search and a heuristic programming technique for moving infeasible points back into the feasible region. A computer program written in Fortran is included along with documentation for the program.

Chapter four presents the implementation of the SUMT algorithm using the Golden Section method to determine the optimum step size and using one of four gradient methods to determine the direction of search : a first order gradient method, a conjugate gradient method, and two versions of a second order gradient method. The computer program written in Fortran is included along with documentation on how to use the program.

Chapter five presents a discussion of the large computer versus the micro/personal computer in terms of nonlinear programming routines.

1.6 REFERENCES

1. Carrol, C. W., "An Operations Research Approach to the Economic Optimization of a Kraft Pulping Process", Ph.D. Dissertion, Institute of Paper Chemistry, Appletown, Wisc., 1959.

2. Carroll, C. W., "The Created Response Surface Technique for Optimizing Nonlinear Restrained Systems", Operations Research, 9, 169-184, 1961.

3. Fiacco, A. V., and G. P. McCormick, "The Sequential Unconstrained Minimization Technique for Nonlinear Programming : A Primal-Dual Method", Management Sci., 10, 601-617, 1964.

4. Fiacco, A. V., and G. P. McCormick, "Computational Algorithm for the Sequential Unconstrained Minimizatin Technique for Nonlinear Programming", Management Sci., 10, 601-617, 1964.

5. Fiacco, A. V., and G. P. McCormick, "SUMT without parameters", Systems Research Memorandum No. 121, Technical Institute, Northwestern University, Evanston, Illinois, 1965.

6. Fiacco, A. V., and G. P. McCormick, "Extension of SUMT for Nonlinear Programming : Equality Constraints and Extrapolation", Management Sci., 12 (11) : 816-829, 1966.

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

8. Krishna, S. V., "Nonlinear Optimization by the Sequential Unconstrained Minimization Technique Using Conjugate Gradients Methods", M.S. Report, Department of Industrial Engineering, Kansas State University, 1971.

9. Hooke, R., and T. A. Jeeves, "Direct Search Solution of Numerical and Statistical Problems", J. Assoc. Comp. Mach., 8, p. 212, 1961.

10. Hwang, C. L., K. C. Lai, F. A. Tillman, and L. T. Fan, "Optimization of System Reliability by the Sequential Unconstrained Minimization Technique", IEEE Trans. on Reliability, vol. R-24., pp. 133-135.

11. Hwang, C. L., L. T. Fan, and S. Kumar, "Hooke and Jeeves Pattern Search Solution to Optimal Production Planning Problems", Report No. 18, Insititute of Systems Design and Optimization, Kansas State University, 1969.

12. Lai, K. C., "Optimization of Industrial Management Systems by the Sequential Unconstrained Minimization Technique", M.S. Report, Dept. of Industrial Engineering, Kansas State University, 1970.

13. Tillman, F. A., C. L. Hwang, and W. Kuo, Optimization of Systems Reliability, Marcel Dekker, New York, 1980.

14. McCormick, G. P., W. C. Mylander III, A. V. Fiacco, "Computer Program Implementing the Sequential Unconstrained Minimization Technique for Nonlinear Programming", Advanced Research Department Technical Paper RAC-TP-151, Research Analysis Corporation, April 1965.

15. Kuester, J. L., and J. H. Mize, Optimization Techniques with Fortran, McGraw-Hill Book Company, New York, 1973.

16. Hsu, F. T., L. T. Fan, and C. L. Hwang, "Sequential Unconstrained Minimization Technique (SUMT) for Optimal Production Planning", Report No. 26, Institute for Systems Design and Optimization, Kansas State University, 1971.

CHAPTER 2

HOOKE AND JEEVES PATTERN SEARCH

2.1. INTRODUCTION

This program finds the local minimum of a multivariable, unconstrained, nonlinear function :

$$\text{Minimize } F(x_1, x_2, \ldots, x_r)$$

The procedure is based on the direct search method proposed by Hooke and Jeeves [2]. No derivatives are required. The procedure assumes a unimodal function; therefore, if more than one minimum exists or the shape of the surface is unknown, several sets of starting values are recommended.

2.2. METHOD

2.2.1 ALGORITHM AND FLOWCHARTS

The direct search method of Hooke and Jeeves [2] is a sequential search routine for minimizing a function $f(\underline{x})$ of more than one variable $\underline{x} = (x_1, x_2, \ldots, x_r)$. The argument $\underline{x}$ is varied until the minimum of $f(\underline{x})$ is obtained. The search routine determines the sequence of values for $\underline{x}$. The successive values of $\underline{x}$ can be interpreted as points in an r-dimensional space. The procedure consists of two types of moves: Exploratory and Pattern. The descriptive flow diagram for the Hooke and Jeeves pattern search is given in Figure 2.1.

A move is defined as the procedure of going from a given point to the following point. A move is a success if the value of $f(\underline{x})$ decreases (for minimization); otherwise, it is a failure. The first type of move is an exploratory move which is designed to explore the local behavior of the objective function, $f(\underline{x})$. The success or failure of the exploratory moves

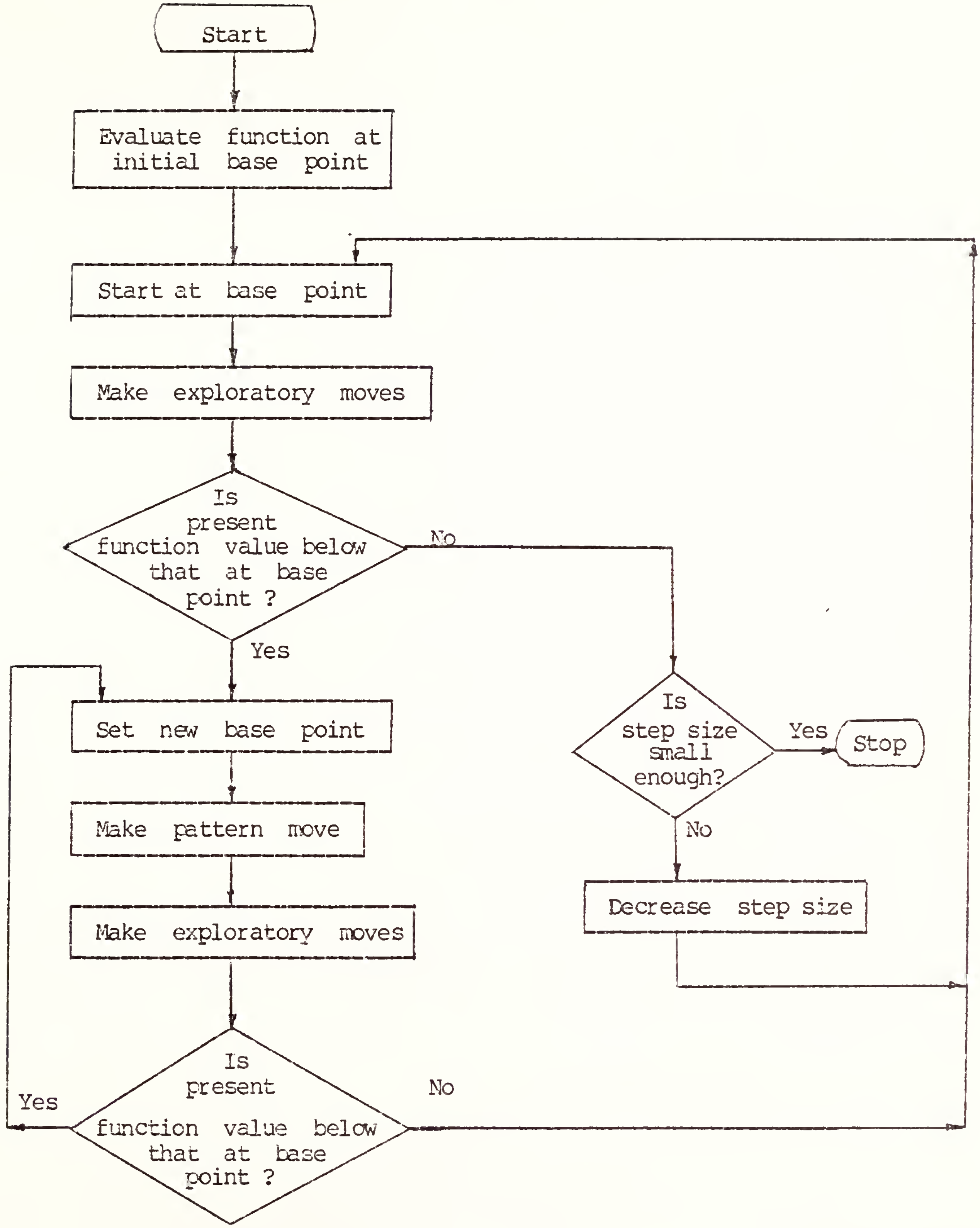

Fig. 2.1. Descriptive flow diagram for Hooke and Jeeves pattern search [2]

is utilized by combining it into a pattern which indicates a probable direction for a successful move [2,3].

The exploratory move is performed as follows :

1. Introduce a starting point $\underline{x}$ with a prescribed step length d_i in each of the independent variables x_i, $i = 1, 2, \ldots, r$.
2. Compute the objective function, $f(\underline{x})$ where $\underline{x} = (x_1, x_2, \ldots, x_r)$.

Repeat the following four steps for $i = 1$ to r. (see Figure 2.2)

3. Set $x_{old} = x_i$ where x_{old} holds the original value of x_i before a step size is taken in that dimension.
4. Take a step in the ith dimension by setting $x_i = x_{old} + d_i$.
5. Compute $f_i(\underline{x})$ at the trial point $\underline{x}$ where only x_i, the value at the ith dimension, has been changed.
6. Compare $f_i(\underline{x})$ with $f(\underline{x})$:
   (i) If $f_i(\underline{x}) < f(\underline{x})$, then the move is a success so set $f(\underline{x}) = f_i(\underline{x})$ and return to step 3.
   (ii) If $f_i(\underline{x}) \geq f(\underline{x})$, set $x_i = x_{old} - d_i$, compute $f_i(\underline{x})$ and see if $f_i(\underline{x}) < f(\underline{x})$
      a) If $f_i(\underline{x}) < f(\underline{x})$ then the move is a success so set $f(\underline{x}) = f_i(\underline{x})$ and repeat from step 3.
      b) If $f_i(\underline{x}) \geq f(\underline{x})$, then the move is a failure and set $x_i = x_{old}$, its original value, and repeat from step 3.

The point $\underline{x}_B$ obtained at the end of the exploratory moves, which is reached by repeating step 3 until $i=r$, is defined as a <u>base point</u>. The starting point introduced in step 1 of the exploratory move is either a starting base point or a point obtained by the pattern move.

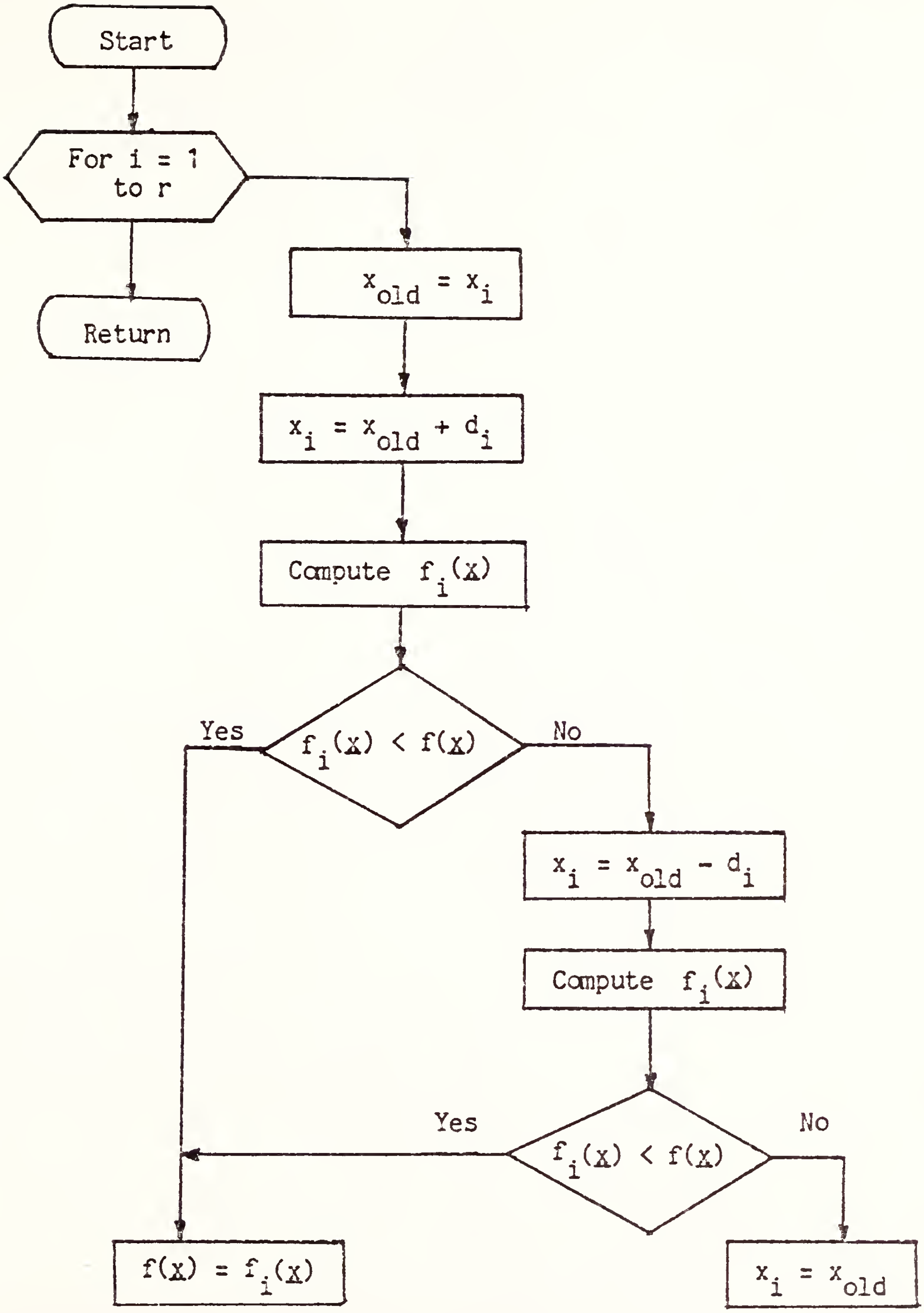

Fig. 2.2 Structured diagram for the exploratory moves procedure

The pattern move is designed to utilize the information acquired in the exploratory moves, and executes the actual minimization of the function by moving in the direction of the established pattern. The pattern move is a simple step from the current base to the point

$$\underline{x} = \underline{x}_B + (\underline{x}_B - \underline{x}_B^*) \tag{1}$$

where $\underline{x}_B^*$ is the preceding base point.

Following the pattern move a series of exploratory moves is conducted to further improve the pattern. If the pattern move followed by the exploratory moves brings no improvement, the pattern move is a failure. Then we return to the last base which becomes a starting base and the process is repeated.

If the exploratory moves from any starting base do not yield a point which is better than this base, the lengths of all the steps are reduced and the moves are repeated. Convergence is assumed when the step lengths, d_i, have been reduced below predetermined limits.

2.2.2 NUMERICAL EXAMPLE

To illustrate the method a simple production scheduling problem will be considered [3]. The function to be minimized is

$$f(x_1,x_2) = 100(x_1-15)^2 + 20(28-x_1)^2 + 100(x_2-x_1)^2 + 20(38-x_1-x_2)^2 \tag{2}$$

To illustrate the procedure, contour lines for equal values of the total cost given by equation (2) are shown in Fig. 2.3. Also presented in the figure are the steps of the Hooke and Jeeves pattern search procedure described in the preceding section. The numbers on the points indicate the sequence in which they are selected. The number on each point also

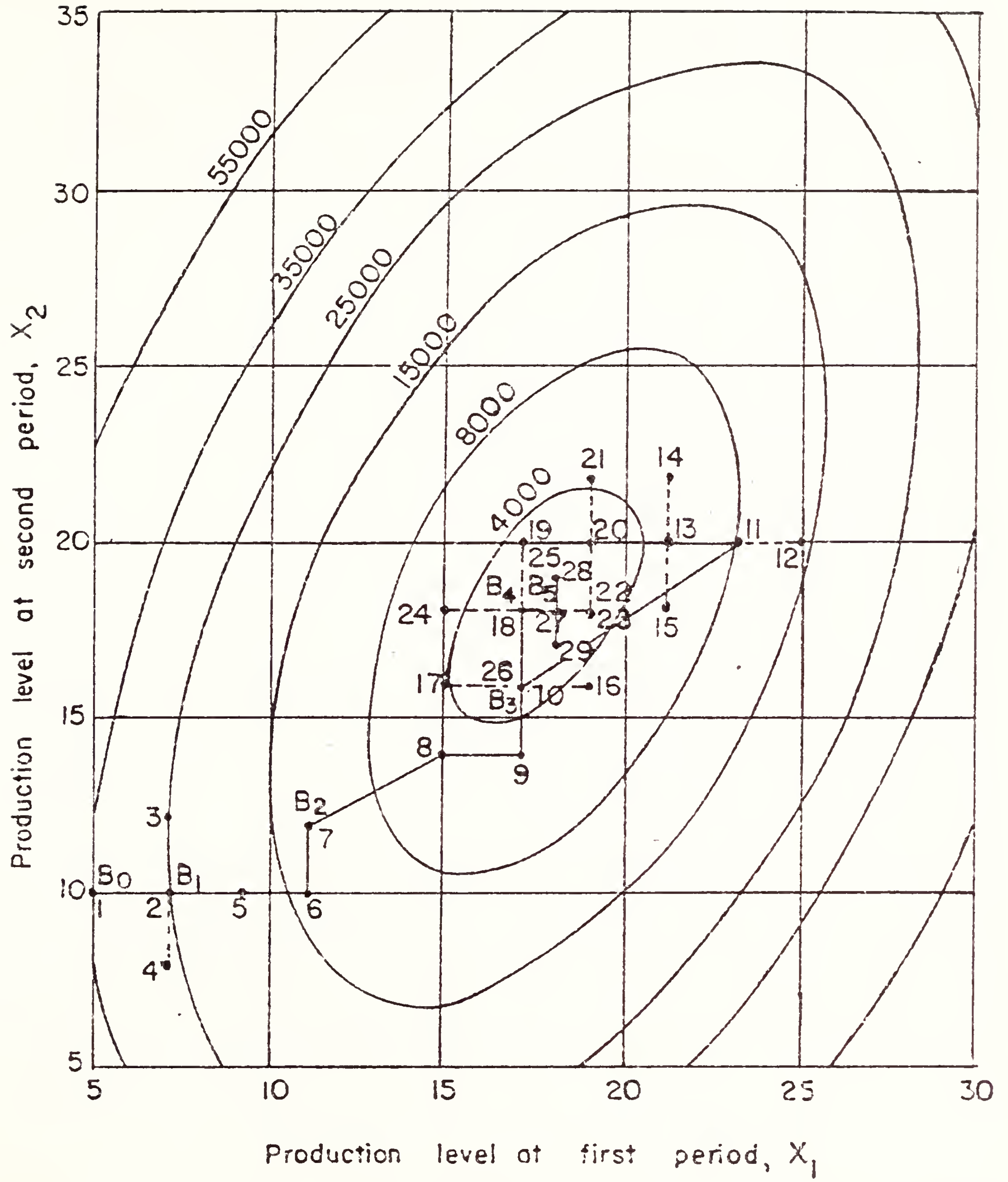

Fig. 2.3 Hooke and Jeeves pattern search applied to production scheduling problem involving two decision variables.

corresponds to the number of the function values computed from the beginning of the procedure up to and including that point. Table 2.1 presents step by step results of applying the Hooke and Jeeves pattern search method to the two dimensional production scheduling problem.

The point, $\underline{x}^1(x_1,x_2) = \underline{x}^1(5,10)$, is the starting base. The step length is $\underline{d} = (d_1,d_2) = (2,2)$. The new base $\underline{x}^2(7,10)$ is obtained by the exploratory moves where $\underline{x}^3(7,12)$ and $\underline{x}^4(7,8)$ are failures. Note that $f(\underline{x}^2) < f(\underline{x}^1)$ whereas $f(\underline{x}^3) < f(\underline{x}^2)$ and $f(\underline{x}^4) > f(\underline{x}^2)$.

Point $\underline{x}^5(9,10)$ is obtained by the pattern move based on equation (1) where $\underline{x}_B^* = \underline{x}^1$ and $\underline{x}_B = \underline{x}^2$.

From $\underline{x}^5$ the exploratory moves are performed again; $\underline{x}^7(11,12)$ becomes a base because $f(\underline{x}^7) < f(\underline{x}^2)$. Note that among these exploratory moves both points $\underline{x}^6$ and $\underline{x}^7$ are successes, that is, $f(\underline{x}^6) < f(\underline{x}^5)$ and $f(\underline{x}^7) < f(\underline{x}^6)$.

Point $\underline{x}^8(15,14)$ is reached by the pattern move according to equation (1) where the last base point $\underline{x}_B^*$ is $\underline{x}^2$ and the new base point $\underline{x}_B$ is $\underline{x}^7$.

Point $\underline{x}^{10}(17,16)$ is the result of the exploratory moves where moves to $\underline{x}^9(17,14)$ and to $\underline{x}^{10}(17,16)$ are successes because $f(\underline{x}^9) < f(\underline{x}^8)$ and $f(\underline{x}^{10}) < f(\underline{x}^9)$. Since $f(\underline{x}^{10}) < f(\underline{x}^7)$, $\underline{x}^{10}$ becomes a new base point. The base points are denoted by B_0, B_1, B_2, ... on Fig. 2.3.

The following pattern move where $\underline{x}_B^* = \underline{x}^7$ and $\underline{x}_B = \underline{x}^{10}$ results in point $\underline{x}^{11}(23,20)$. Point $\underline{x}^{13}(21,20)$ is the result of the exploratory moves following the pattern move, where $\underline{x}^{12}(f(\underline{x}^{12}) > f(\underline{x}^{11}))$, $\underline{x}^{14}(f(\underline{x}^{14}) > f(\underline{x}^{13}))$, and $\underline{x}^{15}(f(\underline{x}^{15}) > f(\underline{x}^{13}))$ are failures, and $\underline{x}^{13}(f(\underline{x}^{13}) < f(\underline{x}^{11}))$ is a success. However, $\underline{x}^{13}$ is not accepted as a new base point because $f(\underline{x}^{13}) > f(\underline{x}^{10})$. We have to return to the last base point $\underline{x}^{10}$, which becomes a starting base and the process is restarted from it.

Starting from base point $\underline{x}^{10}$ with the original step length $\underline{d}$ = (2,2), the new base point $\underline{x}^{18}$(17,18) is obtained by the exploratory moves where $\underline{x}^{16}$ and $\underline{x}^{17}$ are failures.

A pattern move along the direction of the line connecting $\underline{x}^{10}$ and $\underline{x}^{18}$ leads to point $\underline{x}^{19}$. Following this pattern move, the exploratory moves are carried out where $\underline{x}^{21}$, and $\underline{x}^{22}$ are failures and $\underline{x}^{20}$(19,20) is a success; however, $\underline{x}^{20}$ is not accepted as a base because $f(\underline{x}^{20}) > f(\underline{x}^{18})$, and we have to return to the last base $\underline{x}^{18}$ which becomes a starting base.

The exploratory moves from the starting base, $\underline{x}^{18}$, to points [$\underline{x}^{23}$ (=$\underline{x}^{22}$), $\underline{x}^{24}$, $\underline{x}^{25}$(=$\underline{x}^{19}$), and $\underline{x}^{26}$(=$\underline{x}^{10}$)] are all failures. Therefore, the step lengths are reduced from $\underline{d}$ = (2,2) to $\underline{d}$ = (1,1).

The procedure is continued until the limit of the step length, $\underline{d}$ = (0.05,0.05), as the stopping criterion is satisfied. The optimal point $\underline{x}$(x_1=17.81, x_2=18.21) where the value of $f(\underline{x})$ is 2960.74 required 100 calculations of the objective function. The step lengths at this optimal point are $\underline{d}$ = (0.03125, 0.03125).

Table 2.1. Step by Step Results of the Two-Dimensional Production Scheduling Problem

n	$\underline{x}_B$	$\underline{d}$	$\underline{x}$	$f(\underline{x})$	$\underline{x}^n$	$f_i(\underline{x})$	Comments
1	B_0	(2,2)	(5,10)	33,660			Starting base point
2			(5,10)	33,660	(7,10)	24,940	Exp suc
3			(7,10)	24,940	(7,12)	24,940	Exp fail
4			(7,10)	24,940	(7,8)	25,900	Exp fail
2	B_1		(7,10)	24,940			$f(x^2) < f(x^1)$
5			(9,10)	18,140			Pattern
6			(9,10)	18,140	(11,10)	13,260	Exp suc
7			(11,10)	13,260	(11,12)	11,980	Exp suc
7	B_2		(11,12)	11,980			$f(x^7) < f(x^2)$
8			(15,14)	5,100			Pattern
9			(15,14)	5,100	(17,14)	4,700	Exp suc
10			(17,14)	4,700	(17,16)	3,420	Exp suc
10	B_3		(17,16)	3,420			$f(x^{10}) < f(x^7)$
11			(23,20)	8,300			Pattern
12			(23,20)	8,300	(25,20)	13,660	Exp fail
13			(23,20)	8,300	(21,20)	4,860	Exp suc
14			(21,20)	4,860	(21,22)	5,180	Exp fail
15			(21,20)	4,860	(21,18)	5,500	Exp fail
13			(21,20)	4,860			Pattern move failure
							$f(x^{13}) > f(x^{10})$
							Return to $x^{10}(=B_3)$
10	B_3		(17,16)	3,420			Starting base point
16			(17,16)	3,420	(19,16)	4,300	Exp fail

Table 2.1. Step by Step Results of the Two-Dimensional Production Scheduling Problem

n	$\underline{x}_B$	$\underline{d}$	$\underline{x}$	$f(\underline{x})$	$\underline{x}^n$	$f_i(\underline{x})$	Comments
17			(17,16)	3,420	(15,16)	4,460	Exp fail
18			(17,16)	3,420	(17,18)	3,100	Exp suc
18	B_4		(17,18)	3,100			$f(x^{18}) < f(x^{10})$
19			(17,20)	3,740			Pattern
20			(17,20)	3,740	(19,20)	3,340	Exp suc
21			(19,20)	3,340	(19,22)	4,300	Exp fail
22			(19,20)	3,340	(19,18)	3,340	Exp fail
20			(19,20)	3,340			$f(x^{20}) > f(x^{18})$ Pattern move failure Return to $x^{18}(=B_4)$
18	B_4		(17,18)	3,100			Starting base point
23			(17,18)	3,100	(19,18)	3,340	Exp fail
24			(17,18)	3,100	(15,18)	4,780	Exp fail
25			(17,18)	3,100	(17,20)	3,740	Exp fail
26			(17,18)	3,100	(17,16)	3,420	Exp fail
18	B_4		(17,18)	3,100			No better base Exp failures d(2,2)>(0.05,0.05) Reduce d(2,2) to d(1,1).
18	B_4	(1,1)	(17,18)	3,100			Starting base point
27			(17,18)	3,100	(18,18)	2,980	Exp suc
28			(18,18)	2,980	(18,19)	3,020	Exp fail
29			(18,18)	2,980	(18,17)	3,180	Exp fail
27	B_5		(18,18)	2,980			$f(x^{27}) < f(x^{18})$

Table 2.1. Step by Step Results of the Two-Dimensional Production Scheduling Problem

n	$\underline{x}_B$	$\underline{d}$	$\underline{x}$	$f(\underline{x})$	$\underline{x}^n$	$f_i(\underline{x})$	Comments
30			(19,18)	3,340			Pattern
31			(19,18)	3,340	(20,18)	4,180	Exp fail
32			(19,18)	3,340	(18,18)	2,980	Exp suc
33			(18,18)	2,980	(18,19)	3,020	Exp fail
34			(18,18)	2,980	(18,17)	3,180	Exp fail
32			(18,18)	2,980			$f(x^{32}) < f(x^{27})$ Pattern move failure
							Return to x^{27} (=B_5)
27	B_5		(18,18)	2,980			Starting base point
35			(18,18)	2,980	(19,18)	3,340	Exp fail
36			(18,18)	2,980	(17,18)	3,100	Exp fail
37			(18,18)	2,980	(18,19)	3,020	Exp fail
38			(18,18)	2,980	(18,17)	3,180	Exp fail
27			(18,18)	2,980			No better base Exp failure
							d(1,1) > (.05,.05)
							Reduce d(1,1) to d(0.5,0.5)
27	B_5	(0.5,0.5)	(18,18)	2,980			Starting base point
39			(18,18)	2,980	(18.5,18)	3,100	Exp fail
40			(18,18)	2,980	(17.5,18)	2,980	Exp fail
.							
.							
.							
100			(17.81,18.21)	2,961			Optimal point

2.3 COMPUTER PROGRAM DESCRIPTION

2.3.1 DESCRIPTION OF SUBROUTINES

The program consists of a main program, a block data subroutine, an exploratory moves subroutine, an input subroutine, and a user supplied objective function subroutine.

The main program makes the pattern moves, checks the stopping criterion, and reduces the step sizes. It calls on the INPUT subroutine to enter the data needed and the EXPLOR subroutine to perform the searches. It also prints out the intermediate and final solution.

The following subroutines are called by main :

BLOCK DATA INIT initializes the variables in the common block CONST.

EXPLOR performs the exploratory moves and also prints intermediate results.

INPUT reads in the data needed to solve the problem. This includes the problem title, the number of variables, the initial point, the initial step size, the stopping criterion and the printout option.

OBJFUN is a user supplied routine which defines the objective function.

2.3.2 PROGRAM LIMITATIONS

The program will presently handle up to 50 variables. To solve a larger problem the following changes need to be made.

(1) The constant MAXVAR in the Block Data subroutine should be increased.

(2) The dimensions of the arrays in the main program should be increased to the value of MAXVAR.

```
REAL  X(50), STEP(50), NEWBAS(50), OLDBAS(50)
```

The FORMAT statements for printing out results is set up to print a maximum number of function evaluations of 6 digits.

2.3.3 TABLE OF PROGRAM SYMBOLS AND EXPLANATION

TABLE 2.2 Program Symbols and Explanation

FORTRAN Program Symbol	Explanation	Mathematical Symbol
ALPHA	Acceleration factor for pattern move	
BETA	Reduction factor for step size	
CONSOL	The logical unit number of the CRT console.	
COUNT	The objective function counter	
EXPCNT	The 'COUNT' of the current best point found as a result of an exploratory move	
FTRIAL	Function value at a trial point during exploratory moves	$f_i(x)$
FX	Function value at the current best point found from an exploratory move	$f(x)$
FXNB	Function value at current base point	$f(x_B)$
IPRINT	Print option IPRINT = 0 prints optimal solution only = 1 prints values before each step size reduction = 2 prints all steps = 3 prints all details	
LASTBS	The 'COUNT' of the last base point	
MAXCUT	Maximum number of step size reductions. This is used as the stopping criterion.	
MAXVAR	Maximum number of variables which the program can handle. (Presently MAXVAR = 50)	
NEWBAS	An array containing the current base point	x_B
NUMBAS	Base point counter	
NUMCUT	Number of step size reductions performed	
NUMFOR	The 'COUNT' of the point before the exploratory moves begin	
NUMVAR	Number of variables in the problem to be solved.	
N1	Set equal to (NUMCUT + 1) and only used to identify the point to be printed before a step size reduction	

TABLE 2.2 Program Symbols and Explanation

FORTRAN Program Symbol	Explanation	Mathematical Symbol
OLDBAS	An array containing the previous base point	x_B^*
OLDCNT	The 'COUNT' of the previous successful point found during the exploratory moves	
PRINTR	The logical unit number of the printer	
STEP	An array containing the current step size	
STEPOP	The step size option STEPOP = 0 uses computed values STEPOP = 1 allows the user to specify own values	
TITLE	An array containing the title of the problem to be solved	
TZER	Tolerance of zero. (Because of roundoff errors a number which is supposed to be zero may appear on the printout as a small finite number (eg. 1.0E-24). The program checks for a a zero value within the tolerance interval before printing.)	
X	An array containing the current values of the variables	x
XOLD	Used to store the value of the ith dimension of X before a step size is taken in that dimension.	

2.3.4 LISTING OF FORTRAN PROGRAM

```
C               HOOKE AND JEEVES PATTERN SEARCH
C*************************************************************
C
C            THIS PROGRAM IS FOR FINDING THE LOCAL MINIMUM
C       OF A MULTIVARIABLE, UNCONSTRAINED, NONLINEAR FUNCTION.
C       THE PROCEDURE IS BASED ON THE DIRECT SEARCH METHOD
C       PROPOSED BY HOOKE AND JEEVES.
C
C       THE PROGRAM MODIFIED FOR THE MICROCOMPUTER IS WRITTEN BY
C             FRANK HWANG,   I.E,   KSU,   1983.
C
C*************************************************************
C
      BLOCK DATA INIT
      REAL  TZER
      INTEGER  CONSOL, PRINTR, MAXVAR, NUMVAR, IPRINT
      COMMON /CONST/ TZER,CONSOL,PRINTR,MAXVAR,NUMVAR,IPRINT
      DATA  TZER /1.0E-08/
      DATA  CONSOL, PRINTR /1,2/
      DATA  MAXVAR /50/
      END
C
C
      PROGRAM  HOOKE
C
      EXTERNAL  OBJFUN, INIT
C
      INTEGER  CONSOL, PRINTR, MAXVAR, NUMVAR, IPRINT
      INTEGER  MAXCUT, NUMCUT, COUNT, NUMBAS, LASTBS, EXPCNT
      REAL  TZER, FX, FXNB, ALPHA, BETA
      REAL  X(50), STEP(50), NEWBAS(50), OLDBAS(50)
C
      COMMON /CONST/ TZER,CONSOL,PRINTR,MAXVAR,NUMVAR,IPRINT
      DATA  ALPHA, BETA /1.0, 0.5/
      DATA  NUMCUT /0/
      DATA  COUNT, NUMBAS, LASTBS, EXPCNT /0,0,1,0/
C
  299 FORMAT ('0',8X,'BEFORE EXPLORATORY MOVES',4X,'PT',I6,
     1          4X, 'OBJFUN =',E14.6)
  298 FORMAT (' ',8X,4E15.6)
  297 FORMAT ('0',8X,'AFTER EXPLORATORY MOVES ',4X,'PT',I6,
     1          4X, 'OBJFUN =',E14.6)
  295 FORMAT ('0',8X,'AFTER PATTERN MOVE',10X, 'PT',I6,
     1          4X, 'OBJFUN =', E14.6 )
  294 FORMAT (' ',8X,4E15.6)
  293 FORMAT (' ',8X,'BASE POINT NUMBER  ',I5)
  292 FORMAT ('0',8X,'FAILED PATTERN MOVE , RETURN ',
     1           'TO LAST BASE POINT')
C
  290 FORMAT ('0',8X,'* FAILED EXPLORATORY MOVES, CHECK',
     1          ' THE STEP SIZE')
  289 FORMAT (/ '0',8X,'BEFORE STEP-SIZE REDUCTION # ',I2,
     1    / 15X,'FUNCTION COUNT = ',I6,
```

```
     2      / 15X,'OBJFUN =',E14.6 )
   288   FORMAT (' ',8X, 4E15.6)
   286   FORMAT ('0',11X,'* STEP SIZE REDUCED TO : ')
   285   FORMAT (' ',8X, 4E14.5)
   280   FORMAT ('0',//,15X,'**  OPTIMAL RESULTS  **' /
     1    '0',8X,'TOTAL NUMBER OF FUNCTION CALCULATIONS = ',I6/
     2    '0',8X,'OBJECTIVE FUNCTION = ',E15.6)
   279   FORMAT('0',11X,'VARIABLE',6X,'OPTIMAL POINT',5X,
     1          'FINAL STEPSIZE')
   278   FORMAT (' ',13X, I3, 7X, E14.6, 4X, E14.5)
C
C
C           ** READ IN INPUT FROM THE CRT CONSOLE **
C
         CALL  INPUT ( MAXCUT, NEWBAS, STEP )
C
         FXNB = OBJFUN (NEWBAS)
         COUNT = COUNT + 1
C
C            **  START AT BASE POINT **
C
     1   DO 10 I=1,NUMVAR
            X(I) = NEWBAS(I)
    10   CONTINUE
         FX = FXNB
C
C              ** EXPLORATORY MOVES **
C
         IF (IPRINT.GE.2) WRITE (PRINTR,299) LASTBS, FX
         IF (IPRINT.GE.2) WRITE (PRINTR,298) (X(I),I=1,NUMVAR)
         CALL EXPLOR ( FX, X, STEP, LASTBS, EXPCNT, COUNT )
         IF (IPRINT.GE.2) WRITE (PRINTR,297) EXPCNT, FX
         IF (IPRINT.GE.2) WRITE (PRINTR,298) (X(I),I=1,NUMVAR)
         IF (FX .GE. FXNB)  GO TO 110
C
C   **** WHILE EXPLORATORY MOVES MAKE PROGRESS ***
C             ** SET NEW BASE POINT **
C
    15   NUMBAS = NUMBAS + 1
         IF (IPRINT.EQ.3) WRITE (PRINTR,293) NUMBAS
         DO 20 I=1,NUMVAR
            OLDBAS(I) = NEWBAS(I)
            NEWBAS(I) = X(I)
    20   CONTINUE
         FXNB = FX
         LASTBS = EXPCNT
C                          ** PATTERN MOVE **
         DO 30 I=1,NUMVAR
           X(I) = NEWBAS(I) + ALPHA * ( NEWBAS(I) - OLDBAS(I) )
    30   CONTINUE
         FX = OBJFUN(X)
         COUNT = COUNT + 1
         IF ( ABS(FX) .LE. TZER )  FX = 0.0
         IF (IPRINT.GE.2) WRITE (PRINTR,295) COUNT, FX
         IF (IPRINT.GE.2) WRITE (PRINTR,294) (X(I),I=1,NUMVAR)
```

```
C               ** MAKE EXPLORATORY MOVES **
C
      IF (IPRINT.GE.2) WRITE (PRINTR,299) COUNT, FX
      IF (IPRINT.GE.2) WRITE (PRINTR,298) (X(I),I=1,NUMVAR)
      CALL EXPLOR (FX, X, STEP, COUNT, EXPCNT, COUNT )
      IF (IPRINT.GE.2) WRITE (PRINTR,297) EXPCNT, FX
      IF (IPRINT.GE.2) WRITE (PRINTR,298) (X(I),I=1,NUMVAR)
C
      IF (FX.LT.FXNB)  GO TO 15
C   ** END   (* WHILE LOOP *)  **
C
C               ** PATTERN MOVE FAILED **
C
      IF (IPRINT.GE.2) WRITE (PRINTR,292)
      GO TO 1
C
C               ** EXPLORATORY MOVE FAILED **
C            ** CHECK THE STOPPING CRITERION **
C
  110 IF ( IPRINT.GE.2)  WRITE (PRINTR,290)
      IF ( NUMCUT.EQ.MAXCUT )  GO TO 190
C
C            ** STOPPING CRITERION NOT SATISFIED **
C   ** PRINT OUT RESULTS BEFORE THE STEP SIZE REDUCTION **
         N1 = NUMCUT + 1
         WRITE (CONSOL,289) N1, COUNT, FXNB
         WRITE (CONSOL,288)  ( X(I), I=1,NUMVAR )
         IF(IPRINT.EQ.1) WRITE(PRINTR,289) N1, COUNT, FXNB
         IF(IPRINT.EQ.1) WRITE(PRINTR,288) ( X(I), I=1,NUMVAR )
C
C               ** REDUCE THE STEP SIZE **
C
      DO 35 I=1,NUMVAR
        STEP(I) = BETA * STEP(I)
   35 CONTINUE
      NUMCUT = NUMCUT + 1
      WRITE (CONSOL,286)
      WRITE (CONSOL,285)  ( STEP(I), I=1,NUMVAR )
      IF (IPRINT.GE.1) WRITE (PRINTR,286)
      IF(IPRINT.GE.1) WRITE(PRINTR,285) (STEP(I),I=1,NUMVAR)
      GO TO 1
C
C

C          ** OUTPUT THE OPTIMAL RESULTS **
C
  190 WRITE (CONSOL,280) COUNT, FXNB
      WRITE (PRINTR,280) COUNT, FXNB
      WRITE (CONSOL,279)
      WRITE (PRINTR,279)
      WRITE (CONSOL,278) (I, NEWBAS(I), STEP(I), I=1,NUMVAR)
      WRITE (PRINTR,278) (I, NEWBAS(I), STEP(I), I=1,NUMVAR)
C
      STOP
      END
```

```
      SUBROUTINE EXPLOR (FX, X, STEP, NUMFOR, EXPCNT, COUNT)
C
      INTEGER  CONSOL, PRINTR, MAXVAR, NUMVAR, IPRINT
      INTEGER  COUNT, OLDCNT, NUMFOR, EXPCNT
      REAL  X(MAXVAR),  XOLD, STEP(MAXVAR)
      REAL  FX, FTRIAL, TZER
      COMMON /CONST/ TZER,CONSOL,PRINTR,MAXVAR,NUMVAR,IPRINT
C
      IF (IPRINT.EQ.3) WRITE (PRINTR,200)
      OLDCNT = NUMFOR
C
      DO 90 I=1,NUMVAR
         XOLD = X(I)
         X(I) = XOLD + STEP(I)
         FTRIAL = OBJFUN(X)
         COUNT = COUNT + 1
         IF ( ABS( FTRIAL) .LE. TZER )  FTRIAL = 0.0
         IF(IPRINT.EQ.3) WRITE(PRINTR,199) I, COUNT, FTRIAL
         IF(IPRINT.EQ.3) WRITE(PRINTR,198) (X(J),J=1,NUMVAR)
         IF (FTRIAL.LT.FX) GO TO 80
C
C    ** EXPLORATORY MOVE FAILED IN POSITIVE DIRECTION **
C         TRY MOVE IN OPPOSITE DIRECTION
C
         X(I) = XOLD - STEP(I)
         FTRIAL = OBJFUN(X)
         COUNT = COUNT + 1
         IF ( ABS( FTRIAL) .LE. TZER )  FTRIAL = 0.0
         IF(IPRINT.EQ.3) WRITE(PRINTR,199) I,COUNT,FTRIAL
         IF(IPRINT.EQ.3) WRITE(PRINTR,198) (X(J),J=1,NUMVAR)
         IF (FTRIAL.LT.FX) GO TO 80
C
C     **  WHEN EXPLORATORY MOVE FAILS IN OPPOSITE DIRECTION  **
C            MOVE BACK TO ORIGINAL POINT
C
         X(I) = XOLD
         IF(IPRINT.EQ.3) WRITE(PRINTR,199) I, OLDCNT, FX
         IF(IPRINT.EQ.3) WRITE(PRINTR,198) (X(J),J=1,NUMVAR)
         GO TO 90
C
   80    FX = FTRIAL
         OLDCNT = COUNT
   90 CONTINUE
C
      EXPCNT = OLDCNT
C
  200 FORMAT (' ',8X,31('* ') //
     1        ' ',8X,'EXPLORATORY MOVE IN :')
  199 FORMAT (' ',11X,'X(',I2,') DIRECTION ',3X,
     1        'PT',I6, 4X, 'OBJFUN =',E14.6 )
  198 FORMAT (' ',8X, 4E15.6)
C
      RETURN
      END
```

```
      SUBROUTINE INPUT ( MAXCUT, X, STEP )
C
C     THIS SUBROUTINE READS IN THE DATA NEEDED TO SOLVE
C     THE PROBLEM. THIS INCLUDES THE  PROBLEM TITLE,
C     THE NUMBER OF VARIABLES, THE STARTING POINT,
C     THE STARTING STEP SIZES, THE STOPPING CRITERION,
C     AND THE PRINTOUT OPTION.
C
      INTEGER*1  TITLE(58)
      INTEGER  CONSOL, PRINTR, MAXVAR, NUMVAR, IPRINT
      INTEGER  MAXCUT, STEPOP
      REAL  X(MAXVAR), STEP(MAXVAR), TZER
      COMMON /CONST/ TZER,CONSOL,PRINTR,MAXVAR,NUMVAR,IPRINT
C
      WRITE (CONSOL,199)
      WRITE (PRINTR,199)
      WRITE (CONSOL,198)
      WRITE (PRINTR,198)
      WRITE (CONSOL,197)
      WRITE (PRINTR,197)
      WRITE (CONSOL,196)
      READ (CONSOL,195) TITLE
      WRITE (PRINTR,194) TITLE
   20 WRITE (CONSOL,193)
      READ (CONSOL,192) NUMVAR
C
C  *CHECK THAT THE MAXIMUM NUMBER OF VARIABLES IS NOT EXCEEDED
      IF (NUMVAR.LE.MAXVAR) GO TO 50
         WRITE (CONSOL,191)
         WRITE (PRINTR,191)
         WRITE (CONSOL,190)
         WRITE (PRINTR,190)
         STOP
C
   50 WRITE (PRINTR,189)
      WRITE (PRINTR,188) NUMVAR
      WRITE (CONSOL,180)
      DO 70 I=1,NUMVAR
         WRITE (CONSOL,179) I
         READ (CONSOL,178) X(I)
   70 CONTINUE
C
      WRITE (CONSOL,177)
      READ (CONSOL,176) STEPOP
      IF (STEPOP.EQ.1) GO TO 100
         DO 90 I=1,NUMVAR
           STEP(I) = 0.02 * X(I)
           IF ( ABS(STEP(I) ).LE.TZER )  STEP(I) = 0.01
   90    CONTINUE
           GO TO 130
C
  100    DO 110 I=1,NUMVAR
            WRITE (CONSOL,175) I
            READ (CONSOL,174) STEP(I)
  110    CONTINUE
```

```
C
 130   WRITE (CONSOL,173)
       WRITE (PRINTR,173)
       DO 120  I=1,NUMVAR
          WRITE (CONSOL,172) I, X(I), I, STEP(I)
          WRITE (PRINTR,172) I,X(I), I, STEP(I)
 120   CONTINUE
C
       WRITE (CONSOL,171)
       READ (CONSOL,170) MAXCUT
       IF (MAXCUT.EQ.0) MAXCUT = 3
       WRITE (CONSOL,169) MAXCUT
       WRITE (PRINTR,169) MAXCUT
       WRITE (CONSOL,187)
       READ (CONSOL,186) IPRINT
       IF ( IPRINT.EQ.0) WRITE (PRINTR,185)
       IF ( IPRINT.EQ.1) WRITE (PRINTR,184)
       IF ( IPRINT.EQ.2) WRITE (PRINTR,183)
       IF ( IPRINT.EQ.3) WRITE (PRINTR,182)
       WRITE (CONSOL,149)
       WRITE (PRINTR,150)
       IF (IPRINT.GE.1) WRITE (PRINTR,149)
C
 199   FORMAT ('0',20X,'HOOKE AND JEEVES PATTERN SEARCH ')
 198   FORMAT ('0',8X,'MINIMIZES AN UNCONSTRAINED, ',
    1           'MULTIVARIABLE, NONLINEAR FUNCTION')
 197   FORMAT ('0',8X, 31('* ') )
 196   FORMAT ('0','ENTER PROBLEM TITLE : ')
 195   FORMAT (58A1)
 194   FORMAT ('0',15X,58A1)
 193   FORMAT ('0','NUMBER OF VARIABLES : ')
 192   FORMAT (I3)
 191   FORMAT ('0',8X,'*** ERROR ***  THE MAXIMUM NUMBER OF',
    1           ' VARIABLES' /
    2           ' ',8X,' THIS PROGRAM CAN HANDLE IS 20')
 190   FORMAT ('0',8X,'TO SOLVE A LARGER PROBLEM, THE',
    1    ' DIMENSIONS OF THE ARRAYS ' / ' ',8X,
    1    'IN THE MAIN PROGRAM WILL HAVE TO BE MODIFIED' /)
C
 189   FORMAT ('0',8X,'*** INPUT DATA ECHO ***')
 188   FORMAT ('0',8X,'NUMBER OF VARIABLES = ',I2)
 187   FORMAT ('0','PRINTOUT OPTION : ' /
    1    5X,'RETURN  for printout of optimal solution only'/
    2    5X,'   1    for results before each step-size',
    2                 ' cut --- SUGGESTED OPTION' /
    2    5X,'   2    for printout of all steps' /
    3    5X,'   3    for printout of all details' /
    4           ' ','ENTER OPTION : ')
 186   FORMAT (I1)
 185   FORMAT ('0',8X,'PRINT OPTION SELECTED --- PRINTOUT',
    1           ' OF OPTIMAL SOLUTION ONLY')
 184   FORMAT ('0',8X,'PRINT OPTION SELECTED ---  RESULTS',
    1           ' AT EACH STEP-SIZE CUT')
 183   FORMAT ('0',8X,'PRINT OPTION SELECTED ---  PRINTOUT',
    1           ' OF ALL STEPS')
```

```
 182   FORMAT ('0',8X,'PRINT OPTION SELECTED ---  PRINTOUT',
   1           ' OF ALL DETAILS')
C
 180   FORMAT ('0',3X,'ENTER THE INITIAL POINT : ')
 179   FORMAT (' ','STARTING X(',I2,') = ')
 178   FORMAT (F15.0)
 177   FORMAT ('0','STEP SIZE OPTIONS : ' /
   1           5X,'RETURN  to use computed value ',
   1           ' STEP(I) = 0.02 * X(I)' /
   2           5X,'  1     to specify own values ' /
   3           5X,'ENTER OPTION : ')
 176   FORMAT (I1)
 175   FORMAT (' ','STEP(',I2,') = ')
 174   FORMAT (F15.0)
 173   FORMAT ('0',15X,'INITIAL POINT  AND  STEP SIZE')
 172   FORMAT (' ',11X,'X(',I2,') = ',G14.6,
   1             6X,'STEP(',I2,') = ', G14.5)
 171   FORMAT ('0',' THE MAXIMUM NUMBER OF STEP-SIZE',
   1            ' REDUCTIONS :' /
   2           5X, 'RETURN  for default of 3 ' /
   3           5X,'ENTER NUMBER : ')
 170   FORMAT (I2)
 169   FORMAT ('0',8X,'THE MAXIMUM NUMBER OF STEP-SIZE',
   1           ' REDUCTIONS = ',I2 /
   1           ' ',8X,'THE REDUCING FACTOR =  0.5 ')
 150   FORMAT ('0',8X,'**** END OF INPUT ECHO ****'//)
 149   FORMAT ('0',8X,'IN THE FOLLOWING OUTPUT, THE VALUES',
   1    ' PRINTED ARE, RESPECTIVELY : '/
   2    ' ',12X,'THE FUNCTION COUNTER, THE FUNCTION VALUE'/
   3    ' ',12X,'AND THE DECISION VARIABLE VECTOR '// )
C
       RETURN
       END
```

2.3.5 DESCRIPTION OF OUTPUT :

The initial parameter values and the final solution are always printed. Intermediate results are printed if the user specifies IPRINT = 1,2, or 3 on the printout option.

Printout options include :
- 0 Only optimal solution
- 1 Results at each step-size reduction
- 2 Results at each step
- 3 All details

2.3.6 SUMMARY OF USER REQUIREMENTS

1. Create a file on disk that contains OBJFUN, the objective function subroutine.
2. Determine the initial estimate of the optimal point to be used as the starting point.
3. Determine the initial step size and the final step sizes. The program asks for the initial step sizes and MAXCUT, the maximum number of step size reductions. MAXCUT is determined as the number of times the the initial step size must be reduced by 1/2 to get the final step size.

Note : The next two steps will vary depending on the particular compiler used. The following applies if using Microsoft FORTRAN.

4. Compile the objective function subroutine using the F80 command.

```
F80 =B:objfile
```

where objfile is the name of the file which contains the objective function subroutine.

5. Run the program using the L80 command as follows :

```
L80 B:HJSEARCH,B:objfile/G
```

where the B refers to drive B where the program and objective function files are. The /G tells the computer to Go and execute the program.

2.3.7 USER SUPPLIED SUBROUTINE

FUNCTION OBJFUN (X) is the user supplied subroutine in Fortran which defines the objective function to be minimized. The function should be defined in terms of the variable X(I), I=1,N where N is the number of variables. The subroutine should contain a declaration statement

REAL X(50)

An example of the subroutine is shown below for the function

$$\text{Minimize} \quad f(x) = x_1^2 + x_1 x_2 + x_2^2 - 3x_2$$

Note that Fortran statements begin in column 7 or beyond.

```
      FUNCTION  OBJFUN (X)
      REAL  X(50)
      OBJFUN  =  X(1)**2 + X(1)*X(2) + X(2)**2 - 3.*X(2)
      RETURN
      END
```

2.4 INPUT TO THE COMPUTER PROGRAM

2.4.1 CRT DISPLAY OF QUESTIONS

```
                HOOKE AND JEEVES PATTERN SEARCH
USED TO MINIMIZE AN UNCONSTRAINED, MULTIVARIABLE, NONLINEAR FUNCTION

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    ENTER PROBLEM TITLE :

    NUMBER OF VARIABLES :


       ENTER THE  INITIAL POINT :
    STARTING X( 1) =

    STARTING X( 2) =


    STEP SIZE OPTIONS :
       RETURN  to use computed value  STEP(I) = 0.02 * X(I)
          1    to specify own values
       ENTER OPTION :

    STEP( 1) =

    STEP( 2) =


         INITIAL POINT AND STEP SIZE ECHO

       X( 1) =      10.000          STEP( 1) =       1.0000
       X( 2) =      10.000          STEP( 2) =       1.0000

    THE MAXIMUM NUMBER OF STEP-SIZE REDUCTIONS
       RETURN  for default of 3
       ENTER NUMBER :

    THE MAXIMUM NUMBER OF STEP-SIZE REDUCTIONS = 3
    THE REDUCING FACTOR =  0.5


    PRINTOUT OPTION :
        RETURN  for printout of optimal solution only
           1    for results before each step-size  cut --- SUGGESTED
           2    for printout of all steps
           3    for printout of all details
    ENTER OPTION :


    ****  END OF INPUT ECHO  ****
```

2.4.2 NOTES ABOUT THE INPUT

Print options 2 and 3 produce a large amount of data and should only be used for small problems (2 or 3 variables). These two options are mainly a teaching tool used for learning the details of the method.

2.5 TEST PROBLEMS

2.5.1 TEST PROBLEM 1 : SIMPLE PRODUCTION SCHEDULING

2.5.1.1 SUMMARY

NUMBER OF VARIABLES : 2

FUNCTION :

$$\text{Min } F(\underline{x}) = 100(x_1-15)^2 + 20(28-x_1)^2 + 100(x_2-x_1)^2 + 20(38-x_1-x_2)^2$$

STARTING POINT : $x_1 = 5.0$, $x_2 = 10.0$

INITIAL STEP SIZE : $d_1 = 2.0$, $d_2 = 2.0$

MAXIMUM NUMBER OF STEP SIZE REDUCTION : 6

OPTIMAL POINT :

$F(\underline{x}) = 2960.74$

$x_1 = 17.81$

$x_2 = 18.22$

NUMBER OF FUNCTION EVALUATIONS : 100

	MICROCOMPUTER SINGLE PRECISION	MICROCOMPUTER DOUBLE PRECISION	LARGE COMPUTER SINGLE PRECISION
EXECUTION TIME :	0.04 min.	1.57 min.	0.02 min.

2.5.1.2 COMPUTER PRINTOUT OF RESULTS

```
             HOOKE AND JEEVES PATTERN SEARCH

MINIMIZES AN UNCONSTRAINED, MULTIVARIABLE, NONLINEAR FUNCTION

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

        SIMPLE PRODUCTION SCHEDULING PROBLEM

*** INPUT DATA ECHO ***

NUMBER OF VARIABLES =  2

       INITIAL POINT   AND   STEP SIZE
    X( 1) =     5.00000          STEP( 1) =     2.0000
    X( 2) =    10.00000          STEP( 2) =     2.0000

THE MAXIMUM NUMBER OF STEP-SIZE REDUCTIONS =  6
THE REDUCING FACTOR =  0.5

PRINT OPTION SELECTED ---  PRINTOUT OF ALL DETAILS

**** END OF INPUT ECHO ****


IN THE FOLLOWING OUTPUT, THE VALUES PRINTED ARE, RESPECTIVELY :
     THE FUNCTION COUNTER, THE FUNCTION VALUE
     AND THE DECISION VARIABLE VECTOR


BEFORE EXPLORATORY MOVES     PT      1     OBJFUN =    .336600E+05
    .500000E+01     .100000E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
    X( 1) DIRECTION     PT      2     OBJFUN =    .249400E+05
    .700000E+01     .100000E+02
    X( 2) DIRECTION     PT      3     OBJFUN =    .249400E+05
    .700000E+01     .120000E+02
    X( 2) DIRECTION     PT      4     OBJFUN =    .259000E+05
    .700000E+01     .800000E+01
    X( 2) DIRECTION     PT      2     OBJFUN =    .249400E+05
    .700000E+01     .100000E+02

AFTER EXPLORATORY MOVES      PT      2     OBJFUN =    .249400E+05
    .700000E+01     .100000E+02
BASE POINT NUMBER       1

AFTER PATTERN MOVE           PT      5     OBJFUN =    .181400E+05
    .900000E+01     .100000E+02
```

```
BEFORE EXPLORATORY MOVES     PT      5     OBJFUN =    .181400E+05
    .900000E+01    .100000E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
    X( 1) DIRECTION     PT      6     OBJFUN =    .132600E+05
    .110000E+02    .100000E+02
    X( 2) DIRECTION     PT      7     OBJFUN =    .119800E+05
    .110000E+02    .120000E+02

AFTER EXPLORATORY MOVES      PT      7     OBJFUN =    .119800E+05
    .110000E+02    .120000E+02
BASE POINT NUMBER       2

AFTER PATTERN MOVE           PT      8     OBJFUN =    .510000E+04
    .150000E+02    .140000E+02

BEFORE EXPLORATORY MOVES     PT      8     OBJFUN =    .510000E+04
    .150000E+02    .140000E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
    X( 1) DIRECTION     PT      9     OBJFUN =    .470000E+04
    .170000E+02    .140000E+02
    X( 2) DIRECTION     PT     10     OBJFUN =    .342000E+04
    .170000E+02    .160000E+02

AFTER EXPLORATORY MOVES      PT     10     OBJFUN =    .342000E+04
    .170000E+02    .160000E+02
BASE POINT NUMBER       3

AFTER PATTERN MOVE           PT     11     OBJFUN =    .830000E+04
    .230000E+02    .200000E+02

BEFORE EXPLORATORY MOVES     PT     11     OBJFUN =    .830000E+04
    .230000E+02    .200000E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
    X( 1) DIRECTION     PT     12     OBJFUN =    .136600E+05
    .250000E+02    .200000E+02
    X( 1) DIRECTION     PT     13     OBJFUN =    .486000E+04
    .210000E+02    .200000E+02
    X( 2) DIRECTION     PT     14     OBJFUN =    .518000E+04
    .210000E+02    .220000E+02
    X( 2) DIRECTION     PT     15     OBJFUN =    .550000E+04
    .210000E+02    .180000E+02
    X( 2) DIRECTION     PT     13     OBJFUN =    .486000E+04
    .210000E+02    .200000E+02

AFTER EXPLORATORY MOVES      PT     13     OBJFUN =    .486000E+04
    .210000E+02    .200000E+02

FAILED PATTERN MOVE , RETURN TO LAST BASE POINT
```

```
BEFORE EXPLORATORY MOVES     PT     10     OBJFUN =     .342000E+04
     .170000E+02     .160000E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
     X( 1) DIRECTION     PT     16     OBJFUN =     .430000E+04
     .190000E+02     .160000E+02
     X( 1) DIRECTION     PT     17     OBJFUN =     .446000E+04
     .150000E+02     .160000E+02
     X( 1) DIRECTION     PT     10     OBJFUN =     .342000E+04
     .170000E+02     .160000E+02
     X( 2) DIRECTION     PT     18     OBJFUN =     .310000E+04
     .170000E+02     .180000E+02

AFTER EXPLORATORY MOVES     PT     18     OBJFUN =     .310000E+04
     .170000E+02     .180000E+02
BASE POINT NUMBER     4

AFTER PATTERN MOVE          PT     19     OBJFUN =     .374000E+04
     .170000E+02     .200000E+02

BEFORE EXPLORATORY MOVES     PT     19     OBJFUN =     .374000E+04
     .170000E+02     .200000E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
     X( 1) DIRECTION     PT     20     OBJFUN =     .334000E+04
     .190000E+02     .200000E+02
     X( 2) DIRECTION     PT     21     OBJFUN =     .430000E+04
     .190000E+02     .220000E+02
     X( 2) DIRECTION     PT     22     OBJFUN =     .334000E+04
     .190000E+02     .180000E+02
     X( 2) DIRECTION     PT     20     OBJFUN =     .334000E+04
     .190000E+02     .200000E+02

AFTER EXPLORATORY MOVES     PT     20     OBJFUN =     .334000E+04
     .190000E+02     .200000E+02

FAILED PATTERN MOVE , RETURN TO LAST BASE POINT

BEFORE EXPLORATORY MOVES     PT     18     OBJFUN =     .310000E+04
     .170000E+02     .180000E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
     X( 1) DIRECTION     PT     23     OBJFUN =     .334000E+04
     .190000E+02     .180000E+02
     X( 1) DIRECTION     PT     24     OBJFUN =     .478000E+04
     .150000E+02     .180000E+02
     X( 1) DIRECTION     PT     18     OBJFUN =     .310000E+04
     .170000E+02     .180000E+02
     X( 2) DIRECTION     PT     25     OBJFUN =     .374000E+04
     .170000E+02     .200000E+02
     X( 2) DIRECTION     PT     26     OBJFUN =     .342000E+04
     .170000E+02     .160000E+02
```

```
    X( 2) DIRECTION     PT    18    OBJFUN =    .310000E+04
    .170000E+02    .180000E+02

AFTER EXPLORATORY MOVES     PT    18    OBJFUN =    .310000E+04
    .170000E+02    .180000E+02

* FAILED EXPLORATORY MOVES, CHECK THE STEP SIZE

   * STEP SIZE REDUCED TO :
    .10000E+01    .10000E+01

BEFORE EXPLORATORY MOVES    PT    18    OBJFUN =    .310000E+04
    .170000E+02    .180000E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
    X( 1) DIRECTION     PT    27    OBJFUN =    .298000E+04
    .180000E+02    .180000E+02
    X( 2) DIRECTION     PT    28    OBJFUN =    .302000E+04
    .180000E+02    .190000E+02
    X( 2) DIRECTION     PT    29    OBJFUN =    .318000E+04
    .180000E+02    .170000E+02
    X( 2) DIRECTION     PT    27    OBJFUN =    .298000E+04
    .180000E+02    .180000E+02

AFTER EXPLORATORY MOVES     PT    27    OBJFUN =    .298000E+04
    .180000E+02    .180000E+02
BASE POINT NUMBER      5

AFTER PATTERN MOVE          PT    30    OBJFUN =    .334000E+04
    .190000E+02    .180000E+02

BEFORE EXPLORATORY MOVES    PT    30    OBJFUN =    .334000E+04
    .190000E+02    .180000E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
    X( 1) DIRECTION     PT    31    OBJFUN =    .418000E+04
    .200000E+02    .180000E+02
    X( 1) DIRECTION     PT    32    OBJFUN =    .298000E+04
    .180000E+02    .180000E+02
    X( 2) DIRECTION     PT    33    OBJFUN =    .302000E+04
    .180000E+02    .190000E+02
    X( 2) DIRECTION     PT    34    OBJFUN =    .318000E+04
    .180000E+02    .170000E+02
    X( 2) DIRECTION     PT    32    OBJFUN =    .298000E+04
    .180000E+02    .180000E+02

AFTER EXPLORATORY MOVES     PT    32    OBJFUN =    .298000E+04
    .180000E+02    .180000E+02

FAILED PATTERN MOVE , RETURN TO LAST BASE POINT

BEFORE EXPLORATORY MOVES    PT    27    OBJFUN =    .298000E+04
    .180000E+02    .180000E+02
```

.

.

4 more pages of intervening printout is left out

.

.

```
EXPLORATORY MOVE IN :
    X( 1) DIRECTION     PT    75    OBJFUN =    .296750E+04
    .180000E+02     .182500E+02
    X( 1) DIRECTION     PT    76    OBJFUN =    .296250E+04
    .177500E+02     .182500E+02
    X( 1) DIRECTION     PT    67    OBJFUN =    .296125E+04
    .178750E+02     .182500E+02
    X( 2) DIRECTION     PT    77    OBJFUN =    .296312E+04
    .178750E+02     .183750E+02
    X( 2) DIRECTION     PT    78    OBJFUN =    .296312E+04
    .178750E+02     .181250E+02
    X( 2) DIRECTION     PT    67    OBJFUN =    .296125E+04
    .178750E+02     .182500E+02

AFTER EXPLORATORY MOVES      PT    67    OBJFUN =    .296125E+04
    .178750E+02     .182500E+02

* FAILED EXPLORATORY MOVES, CHECK THE STEP SIZE

   * STEP SIZE REDUCED TO :
    .62500E-01     .62500E-01

BEFORE EXPLORATORY MOVES     PT    67    OBJFUN =    .296125E+04
    .178750E+02     .182500E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
    X( 1) DIRECTION     PT    79    OBJFUN =    .296344E+04
    .179375E+02     .182500E+02
    X( 1) DIRECTION     PT    80    OBJFUN =    .296094E+04
    .178125E+02     .182500E+02
    X( 2) DIRECTION     PT    81    OBJFUN =    .296203E+04
    .178125E+02     .183125E+02
    X( 2) DIRECTION     PT    82    OBJFUN =    .296078E+04
    .178125E+02     .181875E+02

AFTER EXPLORATORY MOVES      PT    82    OBJFUN =    .296078E+04
    .178125E+02     .181875E+02
BASE POINT NUMBER     10

AFTER PATTERN MOVE           PT    83    OBJFUN =    .296187E+04
    .177500E+02     .181250E+02

BEFORE EXPLORATORY MOVES     PT    83    OBJFUN =    .296187E+04
    .177500E+02     .181250E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
    X( 1) DIRECTION     PT    84    OBJFUN =    .296156E+04
```

```
    .178125E+02    .181250E+02
    X( 2) DIRECTION      PT    85     OBJFUN =    .296078E+04
    .178125E+02    .181875E+02

AFTER EXPLORATORY MOVES      PT    85     OBJFUN =    .296078E+04
    .178125E+02    .181875E+02

FAILED PATTERN MOVE , RETURN TO LAST BASE POINT

BEFORE EXPLORATORY MOVES     PT    82     OBJFUN =    .296078E+04
    .178125E+02    .181875E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
    X( 1) DIRECTION      PT    86     OBJFUN =    .296172E+04
    .178750E+02    .181875E+02
    X( 1) DIRECTION      PT    87     OBJFUN =    .296172E+04
    .177500E+02    .181875E+02
    X( 1) DIRECTION      PT    82     OBJFUN =    .296078E+04
    .178125E+02    .181875E+02
    X( 2) DIRECTION      PT    88     OBJFUN =    .296094E+04
    .178125E+02    .182500E+02
    X( 2) DIRECTION      PT    89     OBJFUN =    .296156E+04
    .178125E+02    .181250E+02
    X( 2) DIRECTION      PT    82     OBJFUN =    .296078E+04
    .178125E+02    .181875E+02

AFTER EXPLORATORY MOVES      PT    82     OBJFUN =    .296078E+04
    .178125E+02    .181875E+02

* FAILED EXPLORATORY MOVES, CHECK THE STEP SIZE

   * STEP SIZE REDUCED TO :
    .31250E-01    .31250E-01

BEFORE EXPLORATORY MOVES     PT    82     OBJFUN =    .296078E+04
    .178125E+02    .181875E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
    X( 1) DIRECTION      PT    90     OBJFUN =    .296102E+04
    .178437E+02    .181875E+02
    X( 1) DIRECTION      PT    91     OBJFUN =    .296102E+04
    .177812E+02    .181875E+02
    X( 1) DIRECTION      PT    82     OBJFUN =    .296078E+04
    .178125E+02    .181875E+02
    X( 2) DIRECTION      PT    92     OBJFUN =    .296074E+04
    .178125E+02    .182187E+02

AFTER EXPLORATORY MOVES      PT    92     OBJFUN =    .296074E+04
    .178125E+02    .182187E+02
BASE POINT NUMBER      11

AFTER PATTERN MOVE           PT    93     OBJFUN =    .296094E+04
    .178125E+02    .182500E+02
```

```
BEFORE EXPLORATORY MOVES     PT    93    OBJFUN =    .296094E+04
    .178125E+02    .182500E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
    X( 1) DIRECTION     PT    94    OBJFUN =    .296086E+04
    .178437E+02    .182500E+02
    X( 2) DIRECTION     PT    95    OBJFUN =    .296113E+04
    .178437E+02    .182812E+02
    X( 2) DIRECTION     PT    96    OBJFUN =    .296082E+04
    .178437E+02    .182187E+02

AFTER EXPLORATORY MOVES     PT    96    OBJFUN =    .296082E+04
    .178437E+02    .182187E+02

FAILED PATTERN MOVE , RETURN TO LAST BASE POINT

BEFORE EXPLORATORY MOVES     PT    92    OBJFUN =    .296074E+04
    .178125E+02    .182187E+02
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

EXPLORATORY MOVE IN :
    X( 1) DIRECTION     PT    97    OBJFUN =    .296082E+04
    .178437E+02    .182187E+02
    X( 1) DIRECTION     PT    98    OBJFUN =    .296113E+04
    .177812E+02    .182187E+02
    X( 1) DIRECTION     PT    92    OBJFUN =    .296074E+04
    .178125E+02    .182187E+02
    X( 2) DIRECTION     PT    99    OBJFUN =    .296094E+04
    .178125E+02    .182500E+02
    X( 2) DIRECTION     PT   100    OBJFUN =    .296078E+04
    .178125E+02    .181875E+02
    X( 2) DIRECTION     PT    92    OBJFUN =    .296074E+04
    .178125E+02    .182187E+02

AFTER EXPLORATORY MOVES     PT    92    OBJFUN =    .296074E+04
    .178125E+02    .182187E+02

* FAILED EXPLORATORY MOVES, CHECK THE STEP SIZE


       **  OPTIMAL RESULTS  **

TOTAL NUMBER OF FUNCTION CALCULATIONS =    100

OBJECTIVE FUNCTION =     .296074E+04

   VARIABLE      OPTIMAL POINT     FINAL STEPSIZE
       1          .178125E+02        .31250E-01
       2          .182187E+02        .31250E-01
```

2.5.1.3 USER SUPPLIED SUBROUTINE

```
      REAL  FUNCTION OBJFUN (X)
C
C         THE EXAMPLE PROBLEM --- TEST PROBLEM 1
C
      REAL  X(50)
C
      OBJFUN = 100. *( X(1)-15. )**2  +  20. *( 28.-X(1) )**2
     X       + 100. *( X(2)-X(1) )**2 +  20. *( 38.-X(1)-X(2) )**2
C
      RETURN
      END
```

2.5.2 TEST PROBLEM 2 : PERSONNEL AND PRODUCTION SCHEDULING - TEN STAGE

2.5.2.1 SUMMARY

NUMBER OF VARIABLES : 20

FUNCTION :

$$\text{Min} \quad F(\underline{x}) = \sum_{n=1}^{10} S_n$$

where

$$\begin{aligned} S_n = \; & [340.0W_n] + [64.3(W_n - W_{n-1})^2] \\ & + [0.2(P_n - 5.67W_n)^2 + 51.2P_n - 281.0W_n] \\ & + [0.0825(I_n - 320.0)^2] \end{aligned}$$

STARTING POINT :

$$\begin{aligned} \underline{x} &= (x_1, \ldots, x_{10}, x_{11}, \ldots, x_{20}) \\ &= (300, \ldots, 300, 50, \ldots, 50) \end{aligned}$$

INITIAL STEP SIZE :

$$\begin{aligned} \underline{d} &= (d_1, \ldots, d_{10}, d_{11}, \ldots, d_{20}) \\ &= (6.0, \ldots, 6.0, 1.0, \ldots, 1.0) \end{aligned}$$

MAXIMUM NUMBER OF STEP SIZE REDUCTIONS : 3

OPTIMAL POINT :

$$F(\underline{x}) = 241{,}516$$

$$\begin{aligned} \underline{x} = (& 471.00, 444.00, 416.25, 381.75, 376.50, \\ & 364.50, 348.75, 359.25, 329.25, 272.25, \\ & 77.62, 74.25, 70.88, 67.75, 65.12, \\ & 62.75, 60.62, 59.00, 57.38, 56.12) \end{aligned}$$

$$\begin{aligned} \underline{d}_{final} &= (d_1, \ldots, d_{10}, d_{11}, \ldots, d_{20}) \\ &= (0.75, \ldots, 0.75, 0.125, \ldots, 0.125) \end{aligned}$$

NUMBER OF FUNCTION EVALUATIONS : 1709

	MICROCOMPUTER SINGLE PRECISION	MICROCOMPUTER DOUBLE PRECISION	LARGE COMPUTER SINGLE PRECISION
EXECUTION TIME :	3.15 min.	> 60 min.	.02 min.

2.5.2.2 DESCRIPTION OF TEST PROBLEM 2

Numerical Example 2 : A Personnel and Production Scheduling Problem

The capability and practicality of the method is demonstrated by obtaining an optimal solution to a well-known model of Holt, Modigliani, Muth and Simon [1]. This model which has been derived for their paint factory scheduling problem considers the production and inventory system with two independent variables in each planning period. The schematic representation of the problem is shown in Fig. 2.4.

The two independent variables are the production rate and work force level at each month. The problem is to determine the optimal production rate and work force level such that the total operating cost for the planning horizon is minimized.

Let us define

n = a month in the planning horizon

N = the duration, in months

P_n = production rate at the n-th month

W_n = work force level in the n-th month

Q_n = sales rate at the n-th month

I_n = inventory level at the end of the n-th month

Inventory level at the end of each month is computed by using the recursive relationship between sales, production and inventory as follows :

$$I_n = I_{n-1} + P_n - Q_n , \quad n = 1, 2, \ldots, N$$

The model considers that the total operating cost consists of the following four cost items.

1. Regular payroll cost = $340.0W_n$
2. Hiring and layoff cost = $64.3\,(W_n - W_{n-1})^2$
3. Overtime cost = $0.2\,(P_n - 5.67W_n)^2 + 51.2P_n - 281.0W_n$
4. Inventory cost = $0.0825\,(I_n - 320.0)^2$

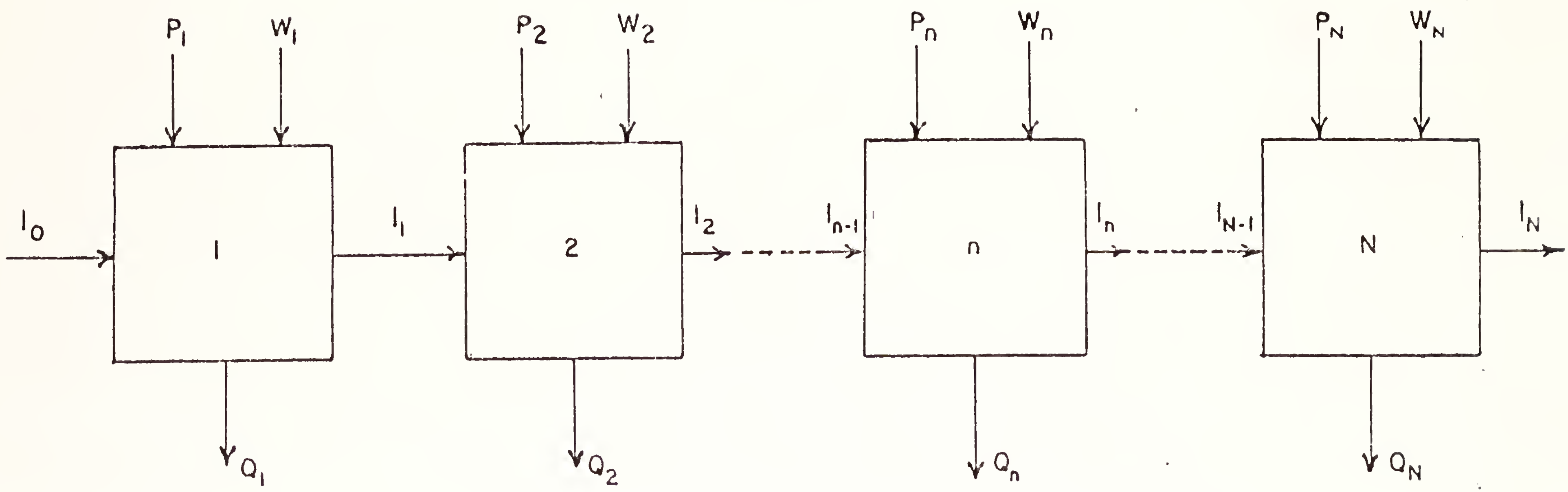

Fig. 2.4 Block diagram for personnel and production scheduling

It is assumed that backlog of orders or negative inventories are permitted.

The decision problem can now be stated as follows :

Choose the optimum values for production rate, P_n, and workforce level, W_n, at each month of the planning horizon so that the total cost S_n which is given by

$$S_N = \sum_{n=1}^{N} S_n$$

is minimized. S_n is defined as

$$\begin{aligned} S_n = \; & [340.0W_n] + [64.3(W_n-W_{n-1})^2] \\ & + [0.2(P_n-5.67W_n)^2 + 51.2P_n - 281.0W_n] \\ & + [0.0825(I_n-320.0)^2] \end{aligned}$$

The numerical data for the ten-stage (20 dimensional) example follows :

$Q_1 = 430, \quad Q_2 = 447, \quad Q_3 = 440, \quad Q_4 = 316, \quad Q_5 = 397,$

$Q_6 = 375, \quad Q_7 = 292, \quad Q_8 = 458, \quad Q_9 = 400, \quad Q_{10} = 350.$

$I_0 = 263$

$W_0 = 81$

Table 2.3 shows the computational results of the example.

In the example, the starting point is selected arbitrarily at $\underline{x}^0 = (P_1^0, \ldots, P_{10}^0, W_1^0, \ldots, W_{10}^0) = (300,\ldots,300, 50,\ldots,50)$. 1709 calculations of the functional value are required for an optimal solution which satisfies the stopping criterion, $\underline{d}_{stop} = (1.0, \ldots, 1.0)$.

Table 2.3 Results of the Personnel and Production Scheduling Problem (20 dimensions)

Month n	Sales Q_n	Production P_n	Inventory I_n	Work Force W_n
0			263.00	81.00
1	430	471.00	304.00	77.62
2	447	444.00	301.00	74.25
3	440	416.25	277.25	70.87
4	316	381.75	343.00	67.75
5	397	376.50	322.50	65.12
6	375	364.50	312.00	62.75
7	292	348.75	368.75	60.62
8	458	359.25	270.00	59.00
9	400	329.25	199.25	57.37
10	350	272.25	121.50	56.12

Total cost S_{10} = \$241,516

2.5.2.3 COMPUTER PRINTOUT OF RESULTS

```
                HOOKE AND JEEVES PATTERN SEARCH

MINIMIZES AN UNCONSTRAINED, MULTIVARIABLE, NONLINEAR FUNCTION

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

     PRODUCTION SCHEDULING --- 10 STAGE

*** INPUT DATA ECHO ***

NUMBER OF VARIABLES = 20

       INITIAL POINT  AND  STEP SIZE
   X( 1) =    300.000        STEP( 1) =     6.0000
   X( 2) =    300.000        STEP( 2) =     6.0000
   X( 3) =    300.000        STEP( 3) =     6.0000
   X( 4) =    300.000        STEP( 4) =     6.0000
   X( 5) =    300.000        STEP( 5) =     6.0000
   X( 6) =    300.000        STEP( 6) =     6.0000
   X( 7) =    300.000        STEP( 7) =     6.0000
   X( 8) =    300.000        STEP( 8) =     6.0000
   X( 9) =    300.000        STEP( 9) =     6.0000
   X(10) =    300.000        STEP(10) =     6.0000
   X(11) =    50.0000        STEP(11) =    1.00000
   X(12) =    50.0000        STEP(12) =    1.00000
   X(13) =    50.0000        STEP(13) =    1.00000
   X(14) =    50.0000        STEP(14) =    1.00000
   X(15) =    50.0000        STEP(15) =    1.00000
   X(16) =    50.0000        STEP(16) =    1.00000
   X(17) =    50.0000        STEP(17) =    1.00000
   X(18) =    50.0000        STEP(18) =    1.00000
   X(19) =    50.0000        STEP(19) =    1.00000
   X(20) =    50.0000        STEP(20) =    1.00000

THE MAXIMUM NUMBER OF STEP-SIZE REDUCTIONS =  3
THE REDUCING FACTOR =  0.5

PRINT OPTION SELECTED ---  RESULTS AT EACH STEP-SIZE CUT

**** END OF INPUT ECHO ****


IN THE FOLLOWING OUTPUT, THE VALUES PRINTED ARE, RESPECTIVELY :
     THE FUNCTION COUNTER, THE FUNCTION VALUE
     AND THE DECISION VARIABLE VECTOR
```

```
BEFORE STEP-SIZE REDUCTION #  1
      FUNCTION COUNT =     671
      OBJFUN =    .241676E+06
   .474000E+03    .438000E+03    .420000E+03    .384000E+03
   .372000E+03    .360000E+03    .348000E+03    .360000E+03
   .336000E+03    .282000E+03    .780000E+02    .740000E+02
   .710000E+02    .680000E+02    .650000E+02    .630000E+02
   .610000E+02    .600000E+02    .590000E+02    .580000E+02

  * STEP SIZE REDUCED TO :
   .30000E+01    .30000E+01    .30000E+01    .30000E+01
   .30000E+01    .30000E+01    .30000E+01    .30000E+01
   .30000E+01    .30000E+01    .50000E+00    .50000E+00
   .50000E+00    .50000E+00    .50000E+00    .50000E+00
   .50000E+00    .50000E+00    .50000E+00    .50000E+00


BEFORE STEP-SIZE REDUCTION #  2
      FUNCTION COUNT =     897
      OBJFUN =    .241571E+06
   .468000E+03    .444000E+03    .417000E+03    .381000E+03
   .378000E+03    .363000E+03    .351000E+03    .360000E+03
   .333000E+03    .276000E+03    .775000E+02    .740000E+02
   .710000E+02    .680000E+02    .655000E+02    .635000E+02
   .615000E+02    .600000E+02    .585000E+02    .570000E+02

  * STEP SIZE REDUCED TO :
   .15000E+01    .15000E+01    .15000E+01    .15000E+01
   .15000E+01    .15000E+01    .15000E+01    .15000E+01
   .15000E+01    .15000E+01    .25000E+00    .25000E+00
   .25000E+00    .25000E+00    .25000E+00    .25000E+00
   .25000E+00    .25000E+00    .25000E+00    .25000E+00


BEFORE STEP-SIZE REDUCTION #  3
      FUNCTION COUNT =    1201
      OBJFUN =    .241540E+06
   .471000E+03    .442500E+03    .417000E+03    .381000E+03
   .376500E+03    .364500E+03    .349500E+03    .360000E+03
   .331500E+03    .274500E+03    .777500E+02    .742500E+02
   .710000E+02    .680000E+02    .655000E+02    .632500E+02
   .612500E+02    .597500E+02    .582500E+02    .570000E+02

  * STEP SIZE REDUCED TO :
   .75000E+00    .75000E+00    .75000E+00    .75000E+00
   .75000E+00    .75000E+00    .75000E+00    .75000E+00
   .75000E+00    .75000E+00    .12500E+00    .12500E+00
   .12500E+00    .12500E+00    .12500E+00    .12500E+00
   .12500E+00    .12500E+00    .12500E+00    .12500E+00
```

** OPTIMAL RESULTS **

TOTAL NUMBER OF FUNCTION CALCULATIONS = 1709

OBJECTIVE FUNCTION = .241516E+06

VARIABLE	OPTIMAL POINT	FINAL STEPSIZE
1	.471000E+03	.75000E+00
2	.444000E+03	.75000E+00
3	.416250E+03	.75000E+00
4	.381750E+03	.75000E+00
5	.376500E+03	.75000E+00
6	.364500E+03	.75000E+00
7	.348750E+03	.75000E+00
8	.359250E+03	.75000E+00
9	.329250E+03	.75000E+00
10	.272250E+03	.75000E+00
11	.776250E+02	.12500E+00
12	.742500E+02	.12500E+00
13	.708750E+02	.12500E+00
14	.677500E+02	.12500E+00
15	.651250E+02	.12500E+00
16	.627500E+02	.12500E+00
17	.606250E+02	.12500E+00
18	.590000E+02	.12500E+00
19	.573750E+02	.12500E+00
20	.561250E+02	.12500E+00

2.5.2.4 USER SUPPLIED SUBROUTINE

```
      FUNCTION OBJFUN (X)
C
C       A PERSONNEL AND PRODUCTION SCHEDULING PROBLEM ---  10 STAGES
C
C       NSTAGE --- THE NUMBER OF STAGES (MONTHS IN THE PLANNING HORIZON)
C       P(N) --- THE PRODUCTION RATE AT THE N-TH MONTH
C       W(N) --- WORK FORCE LEVEL IN THE N-TH MONTH
C       Q(N) --- SALE RATE AT THE N-TH MONTH
C       I(N) --- INVENTORY LEVEL AT THE END OF THE N-TH MONTH
C       S(N) --- OPERATING COSTS FOR THE N-TH MONTH
C       TOTAL --- THE TOTAL OPERATING COSTS FOR PLANNING HORIZON
C
      REAL  X(50)
      REAL  P(25), W(25), I(25), Q(25)
      REAL  S(11), TOTAL
      INTEGER  NSTAGE, J, K, N, N1
C
      DATA  W(1) /81.0/
      DATA  I(1) /263.0/
      DATA  Q(1) / 430.0/
      DATA  Q(2), Q(3), Q(4), Q(5) / 447.0, 440.0, 316.0, 397.0 /
      DATA  Q(6), Q(7), Q(8), Q(9) / 375.0, 292.0, 458.0, 400.0 /
      DATA  Q(10) / 350.0 /
C
       NSTAGE = 10
      DO 10 J = 1,NSTAGE
        P(J) = X(J)
        K = J + NSTAGE
        W(J+1) = X(K)
   10 CONTINUE
C
      TOTAL = 0.0
C
      DO 50  N = 1,NSTAGE
        N1 = N + 1
        I(N1) = I(N1-1) + P(N) - Q(N)
        S(N) = 340.0 * W(N1)  +  64.3 * ( W(N1) - W(N1-1) )**2
     1     + 0.20 * ( P(N) - 5.67 * W(N1) )**2 + 51.2 * P(N)
     2     - 281.0 * W(N1)  +  0.0825 * ( I(N1) - 320.0 )**2
        TOTAL = TOTAL + S(N)
   50 CONTINUE
C
      OBJFUN = TOTAL
C
      RETURN
      END
```

2.6 REFERENCES

1. Holt, C.C., F. Modigliani, J.F. Muth and H.A. Simon, Planning Production, Inventories, and Work Force, Prentice-Hall, Englewood Cliffs, New Jersey, 1960.

2. Hooke, R., and T.A. Jeeves, "Direct Search Solution of Numerical and Statistical Problems", J. Assoc. Comput. Mach., vol. 8, p.212, 1961.

3. Hwang, C.L., L.T. Fan, and S. Kumar, "Hooke and Jeeves Pattern Search Solution to Optimal Production Planning Problems", Report No. 18, Institute for Systems Design and Optimization, Kansas State University, Manhattan, Kansas, 1969.

CHAPTER 3

KSU - SEQUENTIAL UNCONSTRAINED MINIMIZATION TECHNIQUE BASED ON HOOKE AND JEEVES PATTERN SEARCH AND HEURISTIC PROGRAMMING

3.1 INTRODUCTION

The general nonlinear programming problem with nonlinear (and/or linear) inequaltiy and/or equality constraints is to choose $\underline{x}$ to

$$\left.\begin{array}{l} \text{minimize} \quad f(\underline{x}) \\ \text{subject to} \\ \qquad g_i(\underline{x}) \geq 0,\ i = 1, 2, \ldots, m \\ \text{and} \\ \qquad h_j(\underline{x}) = 0,\ j = 1, 2, \ldots, \ell \end{array}\right\} \tag{3.1}$$

where $\underline{x}$ is an n-dimensional vector $(x_1, x_2, \ldots, x_n)$. A number of techniques have been developed to solve this problem. Among them, a technique which was originally proposed by Carroll [1,2] and further developed by Fiacco and McCormick [3,4,5,6,7] is introduced here.

This technique, known as the sequential unconstrained minimization technique (SUMT), is considered one of the simplest and most efficient methods for solving the problem given by equation (3.1). The basic scheme of this technique is that a constrained minimization problem is transformed into a sequence of unconstrained minimization problems which can be optimized by any available techniques for solving unconstrained miminization.

The unconstrained minimization technique which is employed here is the well-known Hooke and Jeeves pattern search technique [8,9]. For increasing the efficiency of the method, some modifications have been made. Among these modifications, a heuristic programming technique [10] is used to handle the inequality constraints of the problem given by equation (3.1).

The method and its computational procedure is illustrated in detail in the following sections of this chapter. The method has been presented in [11,12,13].

3.2 KSU - SEQUENTIAL UNCONSTRAINED MINIMIZATION TECHNIQUE (KSU-SUMT)

The KSU-SUMT technique for solving the problem given by equation (3.1) is based on the minimization of a function

$$P(\underline{x},r_k) = f(\underline{x}) + r_k \sum_{i=1}^{m} 1/g_i(\underline{x}) + r_k^{-1/2} \sum_{j=1}^{\ell} h_j^2(\underline{x}) \tag{3.2}$$

over a strictly monotonic decreasing sequence $\{r_k\}$. The sequential minimization of the unconstrained P function, $P(\underline{x},r_k)$, converges to the solution of the original objective function, $f(\underline{x})$, under certain requirements. The essential requirement is the convexity of the P function.

The intuitive concept of the P function is described below:

Since the sequence $\{r_k\}$ is strictly monotonic decreasing, as $r_k \to 0$ the third term of the P function, $r_k^{-1/2} \sum_{j=1}^{\ell} h_j^2(\underline{x})$, will approach to ∞ unless $h_j(\underline{x}) = 0$ for $j = 1,2,...,\ell$. Thus, in the process of minimizing the P function, the equality constraints will be forced to zero.

The second term of the P function, $r_k \sum_{i=1}^{m} 1/g_i(\underline{x})$, approaches infinity as the value of $\underline{x}$ approaches one of the boundaries of the inequality constraints, $g_i(\underline{x}) \geq 0$. Hence, the value of $\underline{x}$ will tend to remain inside the inequality-constrained feasible region.

The motivation behind this formulation of the P function is the transformation of the original constrained problem into a sequence of unconstrained minimization problems, $\{P(\underline{x},r_k)\}$.

The solution to the problem is to first define the P function as shown in equation (3.2). The search for the minimum P function value is started at an arbitrary point which is inside the feasible region bounded by the inequality constraints. After a minimum P function value is reached, the value of r_k is reduced, and the search is repeated starting from the previous minimum point of the P function. By employing a strictly monotonic decreasing sequence $\{r_k\}$, a monotonic decreasing sequence $\{P_{min}(\underline{x},r_k)\}$ inside the feasible region bounded by the inequality constraints is obtained. The equality constraints, $h_j(\underline{x}) = 0$ for $j = 1,2,\ldots,\ell$, will be satisfied automatically by the nature of the formulation of the P function as r_k approaches zero as explained before.

When $r_k \rightarrow 0$, the second term of equation (3.2), $r_k \sum_{i=1}^{m} 1/g_i(\underline{x})$ approaches zero, while the third term, $r_k^{-1/2} \sum_{j=1}^{m} h_j^2(\underline{x})$, is forced to approach zero, as described before. In other words, as $r_k \rightarrow 0$, $P(\underline{x},r_k) \rightarrow f(\underline{x})$, where $\underline{x}$ is the optimum point which yields the minimum $P(\underline{x},r_k)$ and is the optimum point of the problem given by equation (3.1). Further mathematical proof of the convergence of the method can be seen in reference [3,4,5,6,7].

3.3 COMPUTATIONAL PROCEDURE

The computational procedure for KSU-SUMT based on Hooke and Jeeves pattern search and heuristic programming is summarized below (see Fig. 3.1).

Step (1) Select a starting point $\underline{x}^0 = (x_1^0, x_2^0, \ldots, x_n^0)$, the initial value of the penalty coefficient r_k^0, the initial tolerance limit of the violation to constraints, B^0, and the initial step sizes, $\underline{d}^0$, needed in the search process.

Step (2) Check if the initial point is feasible subject to the inequality constraints. If it is, go to step 3; otherwise, go to step 2a.

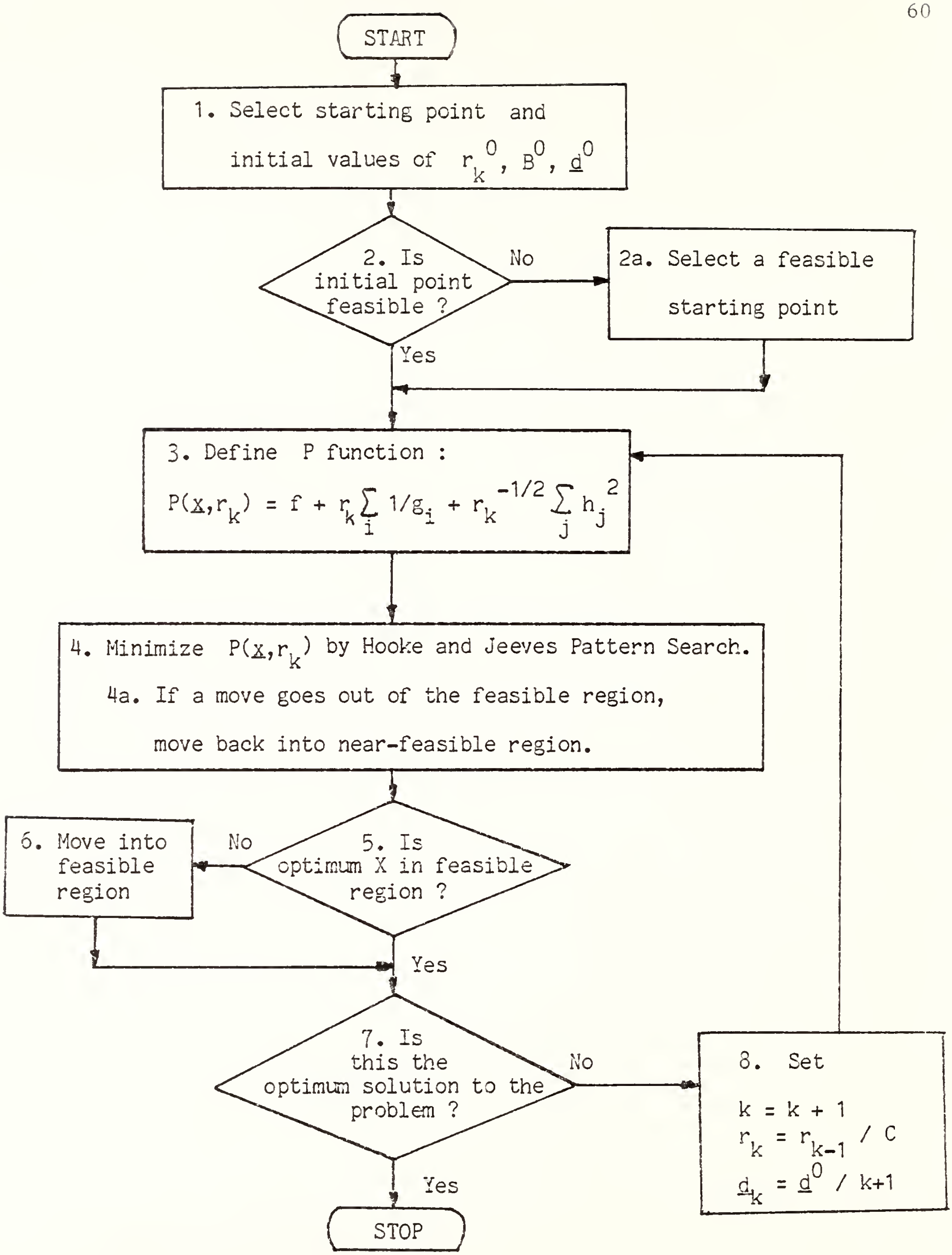

Fig. 3.1. Descriptive flow diagram for KSU-SUMT with modified Hooke and Jeeves Pattern Search.

Step (2a) Locate a feasible starting point by minimizing the total weight of violation, TGH, defined as

$$\text{TGH} = \left[\sum_{t \in T} g_t^{\,2}(\underline{x}^0) + \sum_{s \in R} h_s^{\,2}(\underline{x}^0) \right]^{1/2} \tag{3.3}$$

where $T = \{t \mid g_t(\underline{x}^0) < 0\}$ and $R = \{s \mid h_s(\underline{x}^0) \neq 0\}$. Note that TGH includes only the violated constraints.

Step (3) Define the P function as [6,7]

$$P(\underline{x}, r_k) = f(\underline{x}) + r_k \sum_i 1/g_i(\underline{x}) + r_k^{-1/2} \sum_j h_j^{\,2}(\underline{x}) \tag{3.4}$$

where $g_i(\underline{x}) \geq 0$, $i = 1, 2, \ldots, m$ are inequality constraints, and $h_j(\underline{x}) = 0$, $j = 1, 2, \ldots, \ell$, are equality constraints.

Step (4) Minimize the P function by Hooke and Jeeves pattern search technique. After every move during the search check if the move went out of the feasible region. If it did, go to step 4a; if it did not, continue the search. When the minimum P function value is reached, go to step 5.

Step (4a) Move back to the near-feasible region and then return to step 4. The near-feasible region is defined as the region where all points in the region satisfy the following condition [10]

$$\text{TGH} \leq B$$

where B is the tolerance limit of violation which is sequentially decreased.

Step (5) Check if the P optimum point, $\underline{x}$, obtained in step 4 is inside the feasible region. If it is feasible, go to step 7; if it is near-feasible or not feasible, go to step 6.

Step (6) Move the P optimum point, $\underline{x}$, from the infeasible region into the feasible region along the direction toward the last optimum point, then go to step 7.

Step (7) Check if a stopping criterion such as

$$\left| \left| \frac{f(\underline{x})}{G(\underline{x}, r_k)} \right| - 1 \right| < \varepsilon$$

is satisfied. If the criterion is satisfied, the P optimum point, $\underline{x}$, is also the solution to the original objective function, $f(\underline{x})$; otherwise, go to step 8. The dual value $G(\underline{x}, r_k)$, is defined as [6,7]

$$G(\underline{x}, r_k) = f(\underline{x}) - r_k \sum_{i=1}^{m} 1/g_i(\underline{x}) + r_k^{-1/2} \sum_{j=1}^{\ell} h_j^2(\underline{x})$$

Step (8) Set $k = k+1$; $r_k = r_{k-1}/C$, where C is a constant greater than 1; and $\underline{d}_k = \underline{d}^0/(k+1)$; and go back to step 3.

The following sections present the details of each step described above. The basic Hooke and Jeeves pattern search technique is presented in chapter 2.

3.4 PROCEDURE FOR FINDING A FEASIBLE STARTING POINT FROM THE INFEASIBLE INITIAL POINT

The procedure for selecting a feasible starting point when the initial point is out of the feasible region bounded by inequality constraints, $g_i(\underline{x}) \geq 0$ for $i = 1,2,\ldots,m$, is based on Hooke and Jeeves pattern search technique. For increasing the speed and efficiency of the process, some modifications from the basic Hooke and Jeeves pattern search technique have been made.

Note that in the above description of the feasible region only the inequality constraints are included. The violation to equality constraints is not considered here but is taken into account in the SUMT formulation automatically as explained in Section 3.2 [6,7].

The procedure is summarized below (refer to Figure 3.2).

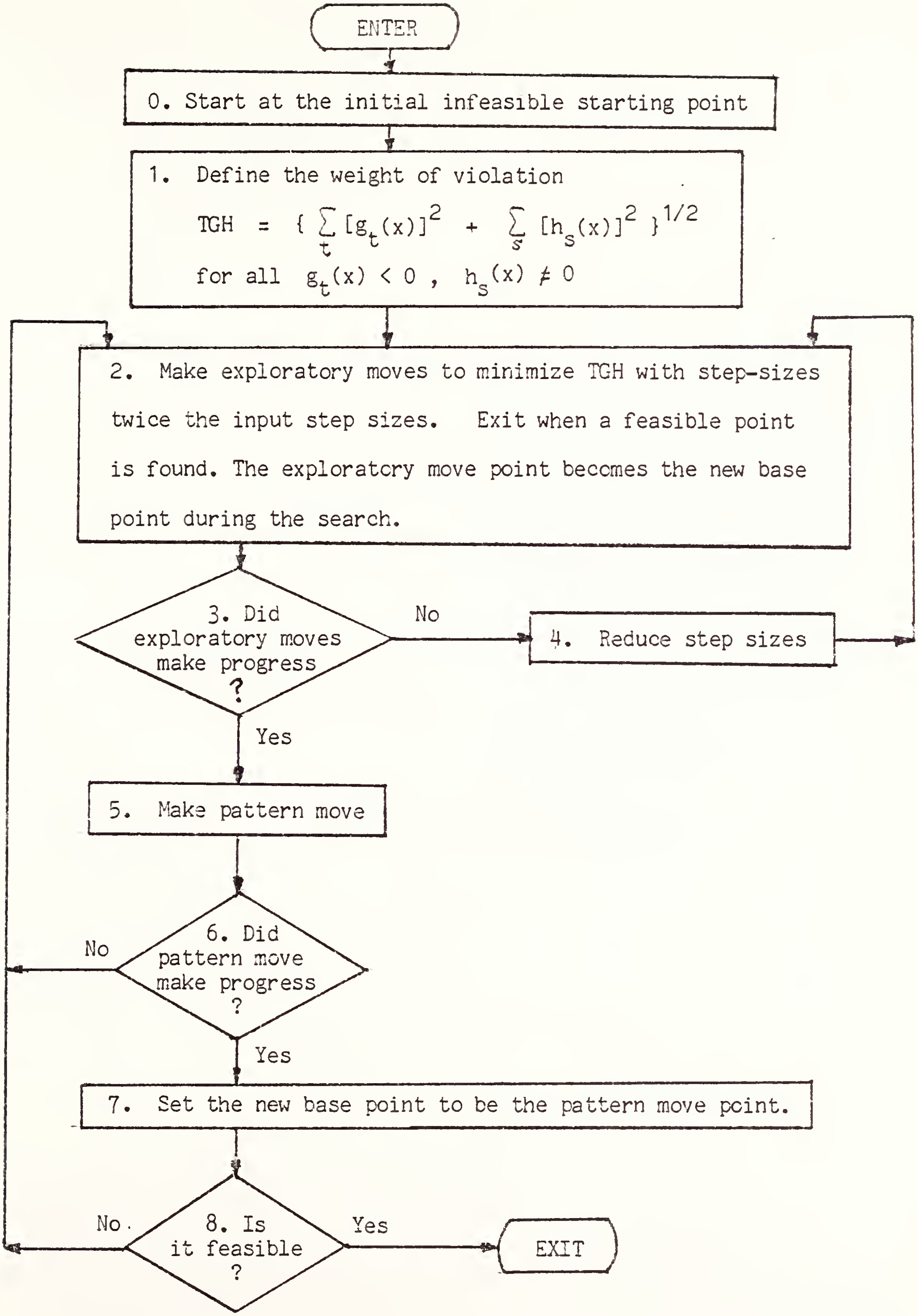

Fig. 3.2 Descriptive flow diagram for locating a feasible starting point

Step (0) Start at the input initial point, $\underline{x}^0$, which is out of the feasible region bounded by the inequality constraints and needs to be moved into the feasible region.

Step (1) Define the weight of violation, TGH, as

$$TGH = \left[\sum_{t \in T} [g_t(\underline{x}^0)]^2 + \sum_{s \in R} [h_s(\underline{x}^0)]^2 \right]^{1/2}$$

where $T = \{t | g_t(\underline{x}^0) < 0\}$ and $R = \{s | h_s(\underline{x}^0) \neq 0\}$.

Step (2) Make an exploratory move to minimize the weight of violation. Note, that TGH includes only the violated constraints. Also note that the objective function to be minimized in this step is TGH. The point obtained at the end of the exploratory moves is defined as the new base point.

For increasing the efficiency of the process, two modifications are made here. First, the starting step-sizes used are twice the input starting step-sizes. Second, after every successful move, the feasibility is checked; whenever a move has reached a point which is inside the feasible region bounded by inequality constraints, the process of selecting a feasible starting point is terminated.

Step (3) Check if the exploratory moves have made any progress in decreasing the value of TGH. If progress has been made, go to step 5; otherwise, go to step 4.

Step (4) Decrease the step sizes and return to step 2.

Step (5) Make a pattern move along the line connecting the two base points to a new pattern move point $\underline{x}_p$.

Step (6) Check if the value of TGH at $\underline{x}_p$ is less than that at $\underline{x}_B$. If it is, go to step 7, otherwise, return to step 2.

Step (7) Set $\underline{x}_B = \underline{x}_p$.

Step (8) Check if $\underline{x}_B$ is in the feasible region bounded by the inequality constraints. If $\underline{x}_B$ is feasible, set the step-sizes back to the original step-sizes and exit this procedure. Otherwise, if $\underline{x}_B$ is still infeasible, return to step 2.

3.5 COMPUTATIONAL PROCEDURE FOR MINIMIZING $P(\underline{x},r_k)$ FUNCTION BY THE MODIFIED HOOKE AND JEEVES PATTERN SEARCH

The computational procedure for minimizing the $P(\underline{x},r_k)$ function is a modification of Hooke and Jeeves pattern search technique [8,9]. The method is a sequential search routine for locating a point $\underline{x} = (x_1, x_2, \ldots, x_n)$ which minimizes the function $P(\underline{x},r_k)$. The original Hooke and Jeeves pattern search method is presented in chapter 2. The procedure presented here is a modification of the technique so that it will handle constraints. The procedure is performed as follows : (see Fig. 3.3)

Step (1) Make exploratory moves to minimize the P function. If an exploratory move goes out of the feasible region, check if Y, the original objective function has decreased. If it has, then move the infeasible point back into the feasible region according to the procedure in Fig. 3.4. Otherwise, if Y has not inproved, then either make a move in the opposite direction or move back to the original point.

Step (2) Check if the exploratory moves have made progress in decreasing the P function. If progress has been made, go to step 3; otherwise, go to step 10.

Step (3) Set the new base point equal to the exploratory move point.

Step (4) Make a pattern move.

Step (5) Check if the pattern move point is feasible. If it is, go to step 8; otherwise, go to step 6.

Step (6) Check if the Y value has improved from its previous best value. If it has, go to step 7; otherwise, return to step 1.

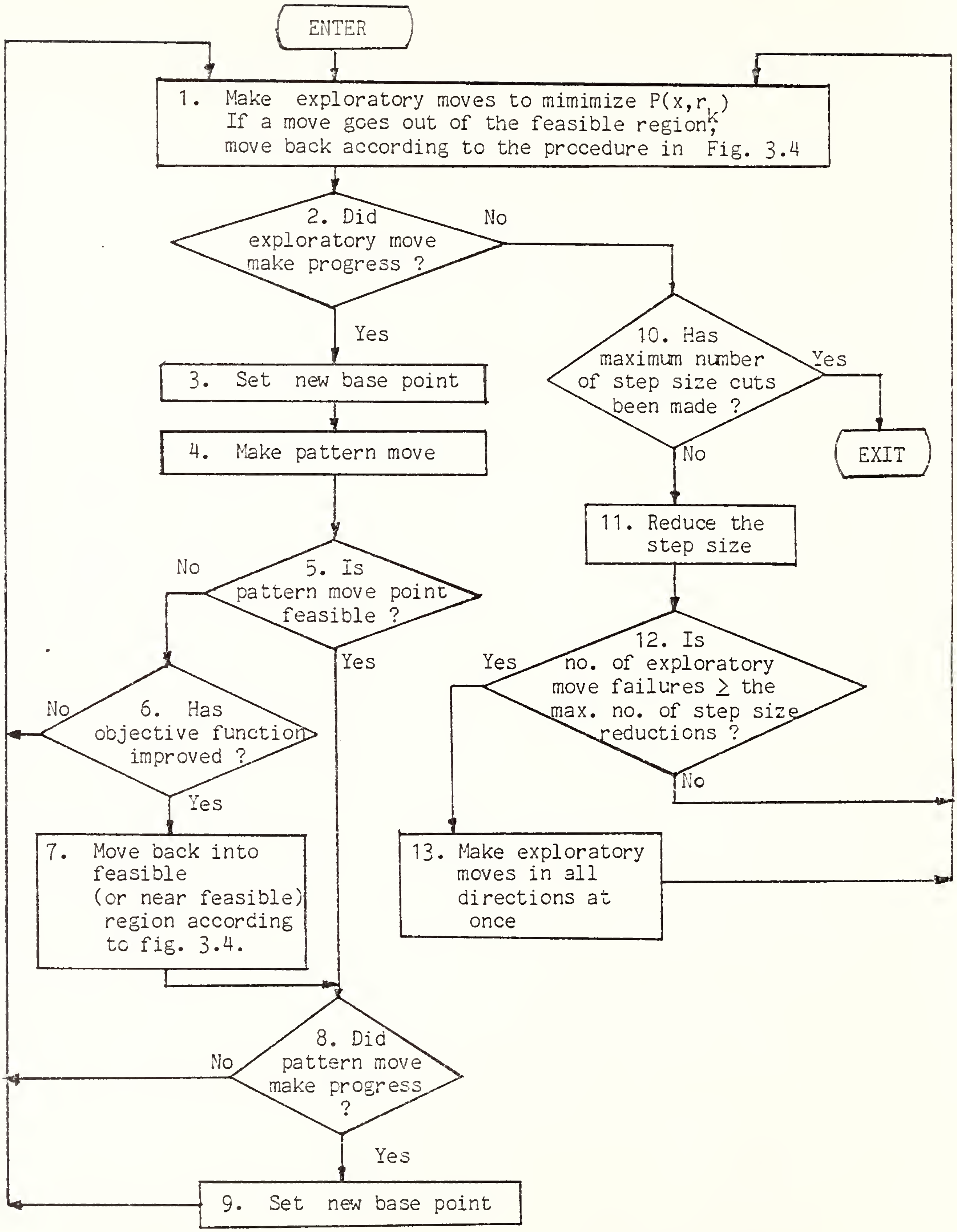

Fig. 3.3 Descriptive flow diagram for minimizing $P(x,r_k)$ function

Step (7) Move back into the feasible or near-feasible region according to the procedure in Fig. 3.4.

Step (8) Check the pattern move point to see whether the P function value has decreased. If it has, go to step 9; otherwise, return to step 1.

Step (9) Set the new base point equal to the pattern move point and return to step 1.

Step (10) Check if the maximum number of step size reductions have been made. If it has, exit the procedure; otherwise, go to step 11.

Step (11) Reduce the step sizes.

Step (12) Check if the number of exploratory move failures is greater than or equal to the maximum number of step size reductions. If it is, go to step 13; otherwise, return to step 1.

Step (13) Reduce the R value, increase the step size, and increase the maximum number of step size reductions by one. Make an exploratory move by taking step size moves in all directions at once. If the move goes out of the feasible region, check if Y, the original objective function value has decreased. If it has, then move the infeasible point back into the feasible or near-feasible region according to the procedure in Fig. 3.4. Otherwise, make simultaneous exploratory moves in the opposite directions. Return to step 1 after completing this step.

3.6 PROCEDURE FOR MOVING AN INFEASIBLE POINT INTO THE FEASIBLE OR NEAR-FEASIBLE REGION BOUNDED BY INEQUALITY CONSTRAINTS

The procedure for moving an infeasible point into the feasible or the near-feasible region bounded by the inequality constraints is based on a simplified Hooke and Jeeves pattern search. Since the optimum will be located at somewhere very close to the boundary of the set of constraints for most of the constrained problems, the moving back procedure used here

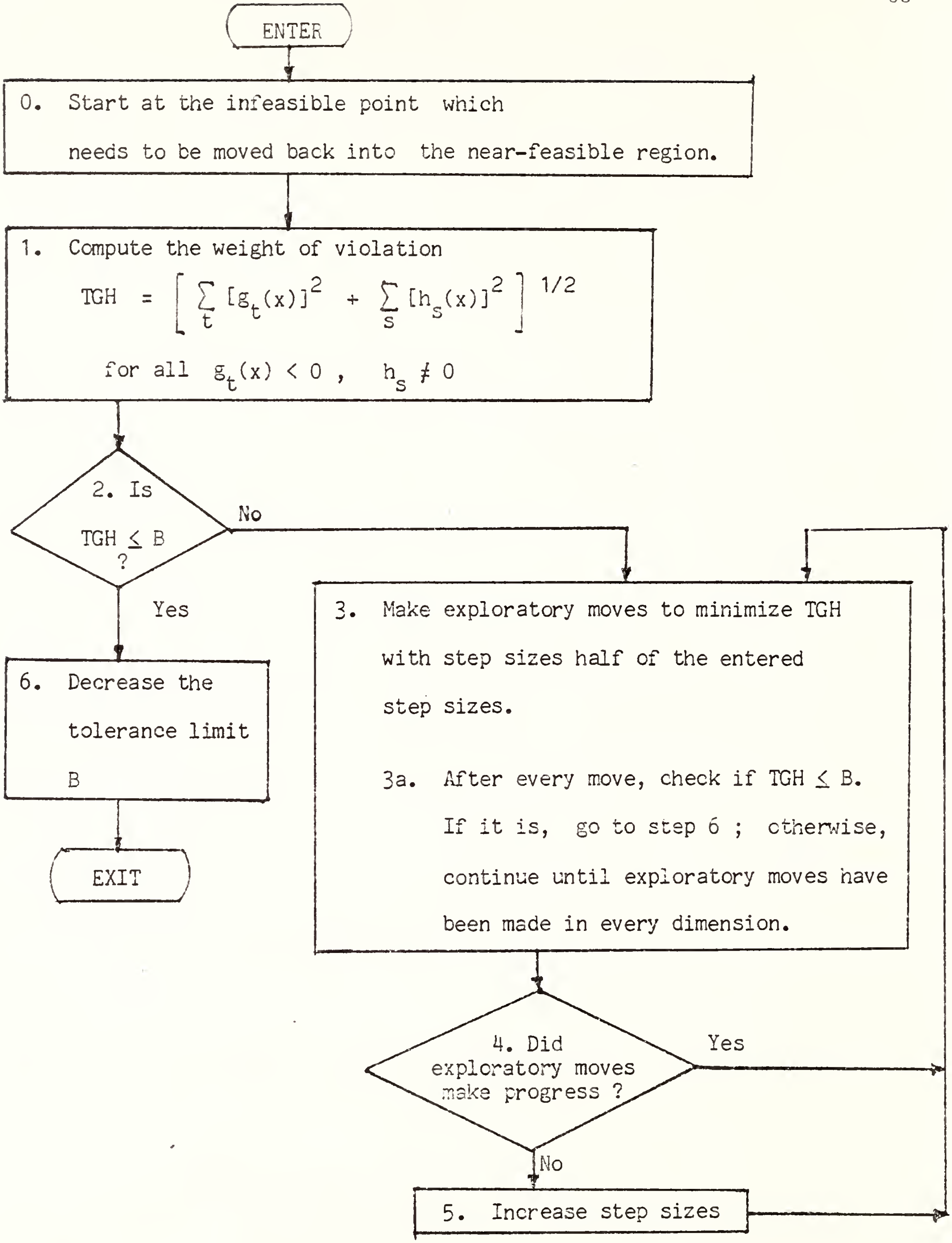

Fig. 3.4 Descriptive flow diagram for moving an infeasible point back into the near-feasible region.

consists of small step-size exploratory moves only. Pattern moves are not used.

The procedure is summarized below (refer to Fig. 3.4).

Step (0) Start at the infeasible point, $\underline{x}$, which is to be moved into the feasible or the near-feasible region bounded by inequaliy constraints.

Step (1) Compute the weight of violation, TGH, at $\underline{x}$

$$TGH = \left[\sum_{t \in T} [g_t(\underline{x})]^2 + \sum_{s \in R} [h_s(\underline{x})]^2 \right]^{1/2}$$

where $T = \{t | g_t(\underline{x}) < 0\}$ and $R = \{s | h_s(\underline{x}) \neq 0\}$.

Step (2) Check if $\underline{x}$ is in the near-feasible region defined as the region where all the points in the region satisfy the following condition [10]

$$TGH \leq B$$

where B is the tolerance limit of violation. If $TGH \leq B$, go to step 6; otherwise, go to step 3.

The starting tolerance limit, B^0, for the kth sub-optimum search is defined as [11]

$$B_k{}^0 = (0.5/n) \sum_{i=1}^{n} d_i$$

where d_i is the starting step-size for the ith dimension used in the kth sub-optimum search; n is the number of dimensions in the problem. This implies that the starting tolerance limit for the kth sub-optimum search is set to be half of the average starting step-sizes. After an infeasible point is moved back to the feasible or near-feasible region bounded by the inequality constraints, the size of the tolerance limit is decreased.

Step (3) Make exploratory moves to minimize TGH using step sizes which are half as large as the step sizes used before entering this routine.

Step (3a) After every move check if $TGH \leq B$. If it is, go to step 6; otherwise, continue until exploratory moves have been made in every dimension.

Step (4) Check if the exploratory moves have made progress in decreasing the value of TGH. If progress has been made, return to step 3; otherwise, go to step 5.

Step (5) Increase the step sizes used for finding a feasible point and return to step 3.

Step (6) Reduce the tolerance limit, B, to 3/4 of its current value. Set $\underline{x}$ to be the feasible or near feasible point found and exit the procedure.

3.7 PROCEDURE FOR MOVING THE NEAR-FEASIBLE kTH SUB-OPTIMUM POINT INTO THE FEASIBLE REGION

After the kth sub-optimum has been reached, it is desirable to have the optimum point in the feasible region subject to all the inequality constraints.

If the optimal point for $P(\underline{x},r_k)$ is in the near-feasible region but not in the feasible region, it will be moved back into the feasible region by the following procedure (refer to Figure 3.5).

Step (0) Start at the kth sub-optimum infeasible point, $\underline{x}_k^0$, which is to be moved into the feasible region.

Step (1) Move $\underline{x}_k^0$ toward $\underline{x}^0_{k-1}$, the feasible (k-1)st sub-optimum point using a step size which is equal to 1/3 of the distance between $\underline{x}_k^0$ and $\underline{x}^0_{k-1}$. Set the new point to be $\underline{x}_k^0$.

Step (2) Check if $\underline{x}_k^0$ is feasible. If $\underline{x}_k^0$ is feasible, exit the procedure; otherwise, go to step 3.

Step (3) If the pull back procedure has been repeated five times without finding a feasible point, go to step 4; otherwise, repeat from step 1.

Step (4) Set $\underline{x}_k^0 = \underline{x}^0_{k-1}$ and exit the procedure.

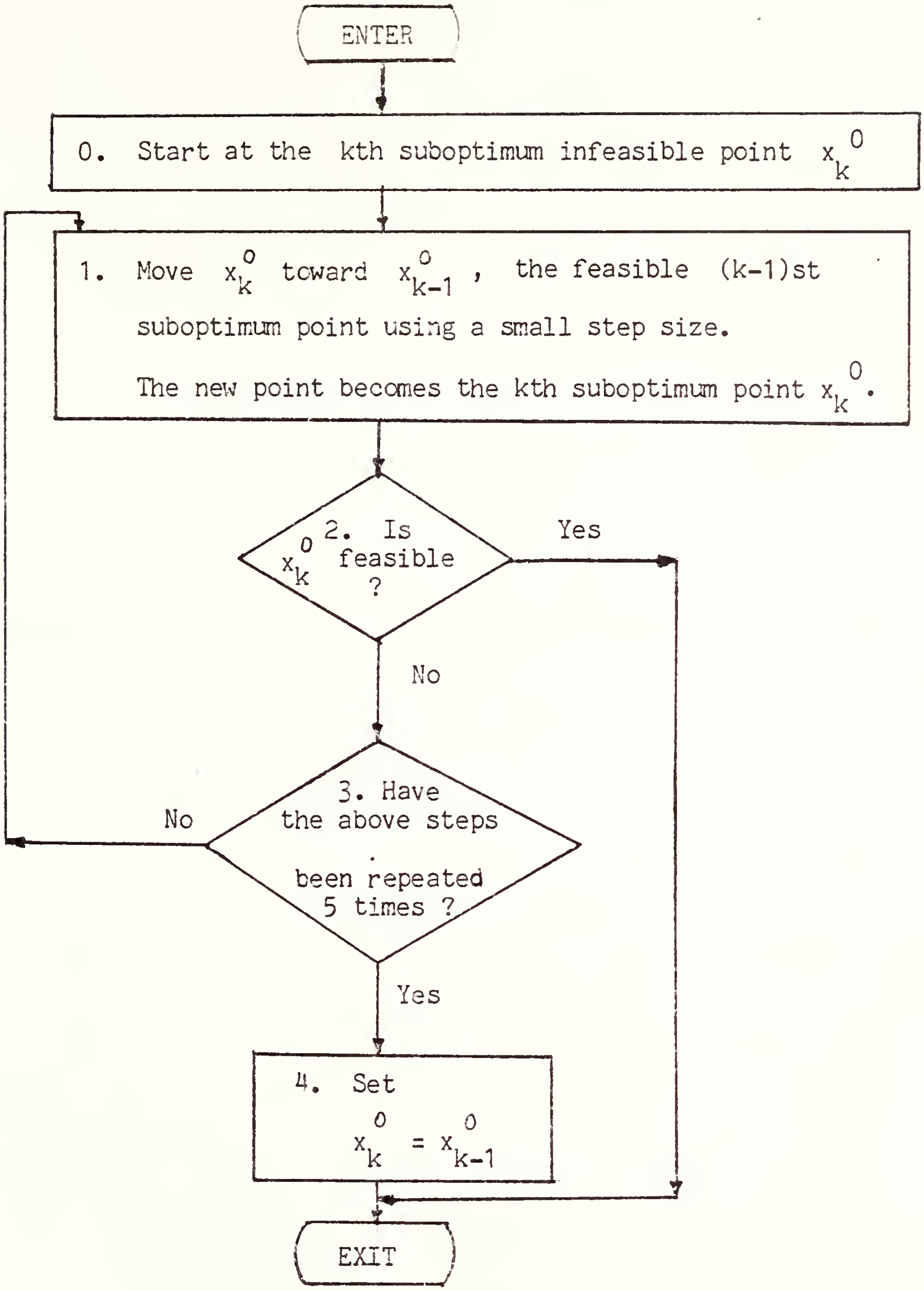

Fig. 3.5 Descriptive flow diagram for moving the near-feasible kth suboptimum point into the feasible region.

3.8 COMPUTER PROGRAM DESCRIPTION

3.8.1 DESCRIPTION OF SUBROUTINES

The main program is supplemented with 7 subroutines : BACK, CKVIOL, INPUT, PENAT, WEIGH, OBRES, OUTPUT.

SUBROUTINE BACK pulls the infeasible point back into the feasible or near-feasible region. The procedure is presented in Section 3.6.

SUBROUTINE CKVIOL checks for violation to inequality constraints and also updates the iteration count.

SUBROUTINE INPUT is used to enter data interactively from the terminal.

SUBROUTINE PENAT computes the penalty terms for SUMT formulation.

SUBROUTINE WEIGH computes the total weight of violation to the inequality and equality constraints as defined by equation 3.3.

SUBROUTINE OBRES defines the objective function and constraints for the problem to be solved. (User-defined).

SUBROUTINE OUTPUT prints out additional information desired by the user. (User-defined).

3.8.2 PROGRAM LIMITATIONS

The program will presently handle a problem with 20 variables, 20 inequality constraints and 20 equality constraints. To solve a larger problem, the dimensions of the arrays in the program must be changed. The key to the changes follows :

X, FX, BX, PX, OX, D, PD	----	N dimensions
FG	----	MG dimensions
FH	----	MH dimensions

The program requires at least 22K bytes of memory.

3.8.3 TABLE OF PROGRAM SYMBOLS AND EXPLANATION

Table 3.1. Program Symbols and Explanation

Program Symbols	Explanation	Mathematical Symbols
B	tolerance limit of constraint violation	
BX(I)	previous base point in Hooke and Jeeves pattern search	
D(I)	step size in Hooke and Jeeves pattern search	d_i
EXPSUC	Exploratory success flag. EXPSUC = TRUE when an exploratory move succeeds in one of the N dimensions ; otherwise EXPSUC = FALSE.	
FEAS	a logical variable indicating whether the current point is feasible or infeasible. FEAS = TRUE if the point is feasible.	
FG(J)	(j)th inequality constraint value at point FX(I)	g_j
FH(K)	(k)th equality constraint value at point FX(I)	h_k
FP	P function value at point FX(I)	P
FRAC	the fraction which is used to multiply the step sizes by in routine BACK.	
FX(I)	the current base point during the exploratory moves	
FY	f function value at point FX(I)	
FTGH	the intermediate least value of TGH during the pulling back procedure	
G(J)	(j)th inequality constraint value at point X(I)	g_j
H(K)	(k)th equality constraint value at point X(I)	h_k
ICONS	the logical unit number for console display	
ICUT	input option code for the starting step size values used at each subproblem search. ICUT = 0 means use input D(I). ICUT = 1 means use D(I)/K for kth stage.	
IDIFF	counts the number of consecutive exploratory move failures plus infeasible pattern moves. When IDIFF = INCUT, then simultaneous step size moves are made.	

Table 3.1. Program Symbols and Explanation

Program Symbols	Explanation	Mathematical Symbols
INCUT	the maximum number of step size reductions for a fixed r. It is used as a subproblem stopping criterion.	
IPRINT	the logical unit number for the printer.	
ISIZE	option for determining the starting step-size values for each subproblem search. (User supplied)	
ISKIP	program control code, ISKIP ← 1 when MXBACK is exceeded in routine BACK before a feasible point is found.	
ITER	number of f function values computed within a subproblem	
ITERB	equal to MXBACK + ITER . It is used in routine BACK to terminate the search for a feasible point.	
ITMAX	maximum number of f function values to be computed for a subproblem. It is used as a subproblem stopping criterion. (User-supplied)	
MG	number of inequality constriants	m
MCUT	program control code, MCUT = 3 when exploratory moves make progress in loop 101 of the main program.	
MH	number of equality constraints	
MAXP	maximum number of subproblems to be solved. It is used as a final stopping criterion.	
MXBACK	The maximum number of iterations (function evaluations) to be made in routine BACK.	
MXFEAS	The maximum number of iterations made in searching for an initial feasible point before terminating the search.	
N	number of decision variables	n
NOBP	It is also the number of times subroutine BACK is called.	
NOCUT	number of step size reductions made for a subproblem.	
NOEXP	number of successful exploratory moves made in the feasible region.	
NOIT	total number of f function values computed since the start of the program.	

Table 3.1. Program Symbols and Explanation

Program Symbols	Explanation	Mathematical Symbols
NOITB	number of exploratory moves made in the infeasible region (subroutine BACK).	
NOFEAS	number of exploratory and pattern moves made in the feasible region	
NOPAT	number of successful pattern moves made in a subproblem.	
NOPULL	number of times the pulling back procedure is executed in the process of moving the infeasible subproblem optimum point into the feasible region.	
NSTAGE	number of stages (subproblems) computed	k
OPTION	the option for using default values for input parameters in routine INPUT. (User-supplied).	
OX(I)	P optimum point of previous subproblem	x^0_{k-1}
P	P function value at point X(I)	P
PB	initial tolerance limit of constraint violation	
PD(I)	initial step size (User-supplied)	d^0_i
PENA1	penalty value to inequality constraints	$r_k \sum_j 1/g_j$
PENA2	penalty value to equality constraints	$r_k^{-1/2} \sum_j h^2_j$
PX(I)	pattern move point in Hooke and Jeeves pattern search	
PULL	a fraction used to pull back the kth suboptimum point into the feasible region	
R	penalty coefficient for SUMT formulation (User supplied or computed by formula)	$r_k = \dfrac{Y}{\sum_i \frac{1}{g_i} + \sum_j h_j}$
RATIO	reducing factor for R from one subproblem to the next. (ie. $r_k = r_{k-1}/C$) (User supplied)	C
STGH	intermediate least value of TGH during search for a feasible starting point	
TGH	weight of violation to constraints	$(\sum_k g_k^2 + \sum_s h_s^2)^{1/2}$

Table 3.1. Program Symbols and Explanation

Program Symbols	Explanation	Mathematical Symbols
THETA	value of the final stopping criterion (user supplied).	
TZER	tolerance of zero. It is used in the INPUT routine to make sure the computed step size values are not too small.	
X(I)	a trial point during the exploratory moves	x_i
XB(NB)	intermediate best point in pulling back procedure	x_i
XOLD	the value of the ith dimension of X before a step size is taken in that dimension. (subroutine BACK).	
Y	f function value at point X(I)	f
YSTOP	computed value of the final stopping criterion	$\left\lvert \left\lvert \dfrac{f}{f - r_k \sum_i \frac{1}{g_i} + r_k^{-1/2} \sum_j h_j^2} \right\rvert - 1 \right\rvert$

3.8.4 LISTING OF FORTRAN PROGRAM

```
      PROGRAM  KSUMT
C
C                   **  KSU SUMT PROGRAM **
C           ****************************************
C
C             THIS PROGRAM IS FOR OPTIMIZING A CONSTRAINED MINIMIZATION
C       PROBLEM BY A COMBINATIONAL USE OF HOOKE AND JEEVES PATTERN SEARCH
C       TECHNIQUE AND SUMT FORMULATION. WHEN THE SEARCH GETS OUT OF THE
C       FEASIBLE REGION, IT WILL BE PULLED BACK BY A HEURISTIC PROGRAMMING
C       TECHNIQUE EXECUTED BY THE SUBROUTINE BACK.
C             THE METHOD EMPLOYS ..
C                SEARCH TECHNIQUE .....  HOOKE AND JEEVES
C                SUMT FORMULATION .....  FIACCO AND MCCORMICK
C                PULL BACK TECHNIQUE ... PAVIANI AND HIMMELBLAU
C             THE ORIGINAL PROGRAM WAS
C              WRITTEN BY :  K. C. LAI  ,   I.E. ,  KSU  IN 1970
C             THE PROGRAM MODIFIED FOR THE MICROCOMPUTER IS
C              WRITTEN BY :  FRANK HWANG ,  I.E. ,  KSU  IN 1983
C
C           ****************************************
C
      EXTERNAL OBRES, OUTPUT
C
      LOGICAL EXPSUC, FEAS
C
      INTEGER  ICONS, ICUT, IDIFF, INCUT, IPRINT, ISIZE, ISKIP
      INTEGER  ITER, ITMAX, MG, MCUT, MH, MAXP, MXFEAS
      INTEGER  N, NOBP, NOCUT, NOEXP, NOFEAS, NOIT, NOITB, NOPAT
      INTEGER  NOPULL, NSTAGE
C
      REAL  X(20),FX(20),BX(20),PX(20),OX(20),PD(20),D(20)
      REAL  FG(20),FH(20)
      REAL  B, FP, FRAC, FY, FTGH, P, PB, PENA1, PENA2, PULL
      REAL  R, RATIO, STGH, TGH, THETA, TOLR, XOLD, Y, YSTOP
C
      COMMON  /BLOGY/ ITMAX, MG, MH, N
      COMMON  /INOUT/ ICONS, IPRINT
C
      DATA  ICONS,IPRINT /1,2/
      DATA  MAXP /50/, MXFEAS /500/
      DATA  TOLR /1.0E-3/
      DATA  NOEXP, NOPAT, NOCUT, NOBP, NOFEAS, NOITB /0,0,0,0,0,0/
      DATA  ITER, NOIT, NSTAGE /0,0,1/
C
1005  FORMAT (20X, 'INITIAL POINT' // 3X, 'Y = ',E11.4, ',   P = ',
     1        E11.4, ',   R = ',E11.4, ',   RATIO = ', E11.4,  /
     2        3X, 'B = ', E11.4, ',  INCUT = ', I4, ',   THETA = ',
     3        E11.4, ' .' /)
1006  FORMAT (10X,'X(',I2,') = ',E14.6, 5X, '  D(',I2,') = ',E14.6)
1007  FORMAT (/,38(' *') / )
1008  FORMAT (/,4X,'** P OPTIMUM..  (',I2, ')'  /
     1  3X, 'FY = ',E13.6, ',   FP = ',E13.6, ',   R = ',E11.4, 3X,
     2  'ITER =',I6 /
```

```
     3    20X, 'NOIT =',I6, ', NOITB =',I5, ',  NOFEAS =',I5,
     4    ',  NOBP =',I5 /
     5    20X, 'NOEXP =',I5, ',  NOPAT =',I5, ',  NOCUT =', I5,' .' /
     6    20X, 'YSTOP = ', E13.6, ' .' /)
 1011   FORMAT (5X, / 5X,'**CONSTRAINTS ..')
 1012   FORMAT (10X,'G(',I2,') = ',E14.6, ' ,' )
 1013   FORMAT (10X,'H(',I2,') = ',E14.6, ' ,' )
 1015   FORMAT (3X,'***** THE ABOVE RESULTS ARE THE FINAL OPTIMUM .')
 1016   FORMAT (3X,'**NO. OF P OPTIMUM EXCEEDED ',I5,' .')
 1020   FORMAT (/,6X,'** FEASIBLE STARTING POINT FOUND ..  ')
 1023   FORMAT (/,' A FEASIBLE STARTING POINT CANNOT BE FOUND AFTER',
     1          I5, ' ITERATIONS' / 1X, 'TRY A DIFFERENT STARTING ',
     2          'POINT AND/OR STEP SIZES')
 1025   FORMAT (2X, '** SUBPROBLEM SEARCH TERMINATED BECAUSE ',
     *          'ITERATION MAXIMUM EXCEEDED **'/)
 1027   FORMAT (3X, '** PROBLEM MAY BE TOO FLAT --- R VALUE REDUCED ',
     *          'AND INCUT VALUE INCREASED')
 1028   FORMAT (6X, 'EXPLORATORY MOVES TAKEN IN ALL DIRECTIONS ',
     *          'AT ONCE  FAILED'/)
 1029   FORMAT (6X, 'EXPLORATORY MOVES TAKEN IN ALL DIRECTIONS ',
     *          'AT ONCE  SUCCESSFUL'/)
C
C
C   *** READ IN PROBLEM NAME, DIMENSIONS, AND OTHER INPUT
C
    1   CALL INPUT ( R, RATIO, INCUT, THETA, ICUT, X, D )
C
        B = 0.0
C
        DO 4 I=1,N
          BX(I) = X(I)
          FX(I) = X(I)
          PD(I) = D(I)
          OX(I) = X(I)
          B = B + 0.5 * D(I)
   4    CONTINUE
C
C   **DECIDE THE STARTING VALUE OF TOLERANCE LIMIT FOR ( G(J) < 0  )
        B = B / N
        PB = B
        B = 2.0 * B
        CALL OBRES (FX,FY,FG,FH)
        CALL CKVIOL (FG,FEAS,ITER)
        CALL WEIGH (FG,FH,STGH)
   11   CALL PENAT (FG,FH, PENA1,PENA2)
C
C   **COMPUTE AN INITIAL VALUE OF R WHEN INPUT R VALUE IS .LE. 0
        IF (R) 12,12,15
   12      R =  ABS ( FY / (PENA1+PENA2) )
           IF (R.LE.TOLR) R=4.0
           R = R/4.0
C
C           * THE P-FUNCTION *
   15   FP = FY + R*PENA1 + R**(-0.5) * PENA2
C
```

```
C***************  OUTPUT THE VALUES AT THE STARTING POINT  **********
C
      WRITE (ICONS,1007)
      WRITE (ICONS,1005)  FY,FP,R,RATIO,B,INCUT,THETA
      WRITE (ICONS,1006)  ( I, FX(I), I, D(I), I=1,N )
      WRITE (ICONS,1011)
      IF (MG.GT.0)  WRITE (ICONS,1012)  ( I, FG(I), I = 1,MG )
      IF (MH.GT.0)  WRITE (ICONS,1013)  ( I, FH(I), I = 1,MH)
      WRITE (IPRINT,1007)
      WRITE (IPRINT,1005) FY,FP,R,RATIO,B,INCUT,THETA
      WRITE (IPRINT,1006) ( I, FX(I), I, D(I), I=1,N )
      WRITE (IPRINT,1011)
      IF (MG.GT.0)  WRITE (IPRINT,1012)  ( I, FG(I), I =1,MG )
      IF (MH.GT.0)  WRITE (IPRINT,1013)  ( I, FH(I), I =1,MH ).
C
      CALL OUTPUT (FX,FY,FG,FH)
C
      WRITE (IPRINT,1007)
C
C   * WHEN A FEASIBLE POINT CANNOT BE FOUND AFTER MXFEAS ITERATIONS,
C   * STOP THE PROGRAM AFTER PRINTING THE BEST POINT
      IF (ITER.GT.MXFEAS) STOP
C
C       * FIG. 1-2 *
C   IS THE INITIAL POINT FEASIBLE ?
      IF (FEAS) GO TO 50
C
C                              **  FIG. 2  **
C*****************  FIND A FEASIBLE STARTING POINT  ***************
C
C          * FIG. 2-2 *
C    **MAKE EXPLORATORY MOVES FOR FINDING A FEASIBLE STARTING POINT.
C
   16 EXPSUC = .FALSE.
C
      DO 28 I=1,N
         FX(I) = X(I) + 2.0 * D(I)
         CALL OBRES (FX, FY, FG, FH)
         CALL CKVIOL (FG,FEAS,ITER)
         CALL WEIGH (FG,FH,TGH)
         IF (FEAS) GO TO 44
         IF (STGH-TGH) 20,20,26
   20       FX(I) = X(I) - 2.0 * D(I)
            CALL OBRES (FX,FY,FG,FH)
            CALL CKVIOL (FG,FEAS,ITER)
            CALL WEIGH (FG,FH,TGH)
            IF (FEAS) GO TO 44
            IF (STGH-TGH) 24,24,26
   24          FX(I) = X(I)
               GO TO 28
C
   26       EXPSUC = .TRUE.
            STGH = TGH
            X(I) = FX(I)
   28 CONTINUE
```

```
C                * FIG. 2-3  *
C     ** DID EXPLORATORY MOVES MAKE PROGRESS ?
        IF (EXPSUC) GO TO 34
C
  29    IF (ITER.LE.MXFEAS) GO TO 30
           WRITE (ICONS,1023) MXFEAS
           WRITE (IPRINT,1023) MXFEAS
           GO TO 11
C
C                * FIG. 2-4  *
C   ** CUT STEP-SIZES FOR FINDING A FEASIBLE STARTING POINT.
   30   DO 32 I=1,N
           D(I) = D(I) * 0.5
   32   CONTINUE
        GO TO 16
C
C                * FIG. 2-5 *
C   ** MAKE PATTERN MOVE FOR FINDING A FEASIBLE STARTING POINT.
   34   DO 36 I=1,N
           PX(I) = FX(I) + ( FX(I) - BX(I) )
   36   CONTINUE
C
        CALL CBRES (PX,FY,FG,FH)
        CALL CKVIOL (FG,FEAS,ITER)
        CALL WEIGH (FG,FH,TGH)
C
C                * FIG. 2-6 *
C  ** DID PATTERN MOVE MAKE PROGRESS ?
        IF (STGH-TGH) 16,16,40
C
C                * FIG. 2-7 *
C   ** THE PATTERN MOVE POINT BECOMES THE NEW BASE POINT
   40   DO 42 I=1,N
           BX(I) = PX(I)
           X(I) = PX(I)
           FX(I) = PX(I)
   42   CONTINUE
C
C                * FIG. 2-8 *
C   ** IS THE NEW BASE POINT FEASIBLE ?
        IF (FEAS) GO TO 44
          STGH=TGH
          GO TO 16
C
   44     DO 46 I=1,N
            D(I) = FD(I)
            OX(I) = FX(I)
            BX(I) = FX(I)
   46     CONTINUE
          ITER = 0
          WRITE (IPRINT,1020)
          GO TO 11
C                    *  END OF PROCEDURE  *
C                FOR FINDING A FEASIBLE STARTING POINT
C*********************************************************************
```

```
C                        **  FIG. 3  **
C****************  MINIMIZING THE P-FUNCTION  ***********************
C
  50   IDIFF=0
       MCUT=1
  51   EXPSUC = .FALSE.
       IDIFF = IDIFF + 1
C
C                * FIG. 3-1 *
C    **MAKE EXPLORATORY MOVES FOR MINIMIZING THE P-FUNCTION
C
       DO 101 I=1,N
          X(I) = FX(I) + D(I)
          CALL OBRES (X,Y,FG,FH)
          CALL CKVIOL (FG,FEAS,ITER)
          IF (FEAS) GO TO 62
          IF (Y.GE.FY) GO TO 68
             CALL BACK (X,D,Y,FG,FH,NOITB,B,ISKIP,ITER)
             NCBP = NCBP + 1
             IF (ITER.GE.ITMAX) GO TO 140
C
C            * ISKIP = 1  MEANS MXBACK WAS REACHED WHILE IN ROUTINE BACK
C                        SO THE POINT IS STILL INFEASIBLE
             IF (ISKIP.EQ.1) GO TO 68
C
  62         NOFEAS = NOFEAS + 1
             CALL PENAT (FG,FH,PENA1,PENA2)
             P = Y + R * PENA1 + R**(-0.5) * PENA2
             IF (P.LT.FP) GO TO 88
C
  68      X(I) = FX(I) - D(I)
          CALL OBRES (X,Y,FG,FH)
          CALL CKVIOL(FG,FEAS,ITER)
          IF (FEAS) GO TO 80
          IF (Y.GE.FY) GO TO 86
             CALL BACK (X,D,Y,FG,FH,NOITB,B,ISKIP,ITER)
             NCBP = NCBP + 1
             IF (ITER.GE.ITMAX) GO TO 140
             IF (ISKIP.EQ.1) GO TO 86
C
  80         NOFEAS = NOFEAS + 1
             CALL PENAT (FG,FH,PENA1,PENA2)
             P = Y + R * PENA1 + R**(-0.5) * PENA2
             IF (P.LT.FP) GO TO 88
C
  86      X(I) = FX(I)
          GO TO 101
C
  88      EXPSUC = .TRUE.
          FY=Y
          FP=P
          FX(I) = X(I)
C
 101   CONTINUE
C
```

```
      IF (ITER.GE.ITMAX) GO TO 140
C
C                       * FIG. 3-2 *
C   ** DID THE EXPLORATORY MOVES MAKE PROGRESS ?
      IF (EXPSUC)  GO TO 111
C
C               * FIG. 3-10 *
C   ** IS STOPPING CRITERION SATISFIED ?
      IF (NOCUT.GE.INCUT) GO TO 150
C
C
C               * FIG. 3-11 *
C   ** CUT STEP-SIZES FOR MINIMIZING THE P-FUNCTION
      DO 105 I=1,N
        D(I) = 0.5 * D(I)
 105  CONTINUE
C
      NOCUT = NOCUT + 1
C
C               * FIG. 3-12 *
      IF (IDIFF.LT.INCUT) GO TO 51
      IF (MCUT.EQ.3) GO TO 51
C
C
C                    * FIG. 3-13 *
C              **  PROCEDURE  FOR TAKING  **
C*********  A STEP SIZE IN ALL DIRECTIONS SIMULTANEOUSLY  *******
C
      WRITE (IPRINT,1027)
      R = R / 2.0
      CALL PENAT (FG,FH,PENA1,PENA2)
      FP = FY + R * PENA1 + R**(-0.5) * PENA2
      INCUT = INCUT + 1
      NOCUT=0
C
      DO 109 I=1,N
        PD(I) = PD(I) * 4.0
        D(I) = PD(I)
 109  CONTINUE
C
      IF (ICUT) 2109,2109,102
2109  DO 2110 I=1,N
        D(I) = D(I) / NSTAGE
2110  CONTINUE
C
 102  DO 103 I=1,N
         X(I) = FX(I) + D(I)
 103  CONTINUE
C
      CALL OBRES (X,Y,FG,FH)
      CALL CKVIOL (FG,FEAS,ITER)
      IF (FEAS) GO TO 1106
      IF (Y.GT.FY) GO TO 1108
         CALL BACK (X,D,Y,FG,FH,NOITB,B,ISKIP,ITER)
         NOBP = NOBP + 1
```

```
          IF (ITER.GE.ITMAX) GO TO 140
          IF (ISKIP.EQ.1) GO TO 1108
C
 1106     NOFEAS = NOFEAS + 1
          CALL PENAT (FG,FH,PENA1,PENA2)
          P = Y + R * PENA1 + R**(-0.5) * PENA2
          IF (P-FP) 1115,1108,1108
C
C     * EXPLORATORY MOVE FAILED IN POSITIVE DIRECTIONS
C     * MAKE MOVE IN OPPOSITE DIRECTIONS
 1108   DO 1109 I=1,N
          X(I) = FX(I) - D(I)
 1109   CONTINUE
C
        CALL OBRES (X,Y,FG,FH)
        CALL CKVIOL (FG,FEAS,ITER)
        IF (FEAS) GO TO 1112
        IF (Y.GT.FY) GO TO 1114
          CALL BACK (X,D,Y,FG,FH,NOITB,B,ISKIP,ITER)
          NCBP = NCBP + 1
          IF (ITER.GE.ITMAX) GO TO 140
          IF (ISKIP.EQ.1) GO TO 1114
C
 1112     NOFEAS = NOFEAS + 1
          CALL PENAT (FG,FH,PENA1,PENA2)
          P = Y + R * PENA1 + R**(-0.5) * PENA2
          IF (P.LT.FP) GO TO 1115
C
C    * EXPLORATORY MOVE FAILED IN OPPOSITE DIRECTION
C    * FX(I) IS STILL THE BEST POINT FOUND SO FAR
 1114   MCUT = 3
        WRITE (IPRINT,1028)
        GO TO 51
C
C    ** EXPLORATORY MOVE MADE PROGRESS
 1115   FP=P
        FY=Y
C
C    * SET NEW BASE POINT *
        DO 1116 I=1,N
          FX(I) = X(I)
 1116   CONTINUE
        WRITE (IPRINT,1029)
        GO TO 50
C
C                          END OF PROCEDURE
C*************  FOR TAKING SIMULTANEOUS STEP SIZES  ************
C
C
C
C    *********** WHEN EXPLORATORY MOVES MADE PROGRESS  **************
C
  111   NOEXP=NOEXP + 1
        MCUT = 3
C
```

```
C                * FIG. 3-3 & 3-4 *
C   ** MAKE PATTERN MOVE FOR MINIMIZING THE P-FUNCTION
C   ** AND SET A NEW BASE POINT
        DO 112 I=1,N
          PX(I) = FX(I) + ( FX(I) - BX(I) )
          BX(I) = FX(I)
  112   CONTINUE
C
        CALL OBRES (PX,Y,FG,FH)
        CALL CKVIOL (FG,FEAS,ITER)
C
C                * FIG. 3-5 *
C  ** IS PATTERN MOVE POINT FEASIBLE ?
        IF (FEAS) GO TO 124
C
C                * FIG. 3-6 *
C  ** HAS THE OBJECTIVE FUNCTION IMPROVED ?
        IF (Y.GT.FY) GO TO 51
C
C                * FIG. 3-7 *
C  ** MOVE BACK INTO THE FEASIBLE OR NEAR FEASIBLE REGION
        CALL BACK (PX,D,Y,FG,FH,NOITB,B,ISKIP,ITER)
        NOBP = NOBP + 1
C
        IF (ITER.GE.ITMAX) GO TO 140
        IF (ISKIP.EQ.1) GO TO 50
C
C                * FIG. 3-8 *
C   ** DID PATTERN MOVE MAKE PROGRESS ?
  124   CALL PENAT (FG,FH,PENA1,PENA2)
        P = Y + R * PENA1 + R**(-0.5)  * PENA2
        IF (P.GE.FP) GO TO 50
C
        NOPAT = NOPAT + 1
        NOFEAS = NOFEAS + 1
C
C                * FIG. 3-9 *
C   ** SET NEW BASE POINT
        DO 129 I=1,N
          FX(I) = PX(I)
  129   CONTINUE
C
        FY=Y
        FP=P
        GO TO 50
C
C                  *  END OF PROCEDURE  *
C                FOR MINIMIZING THE P-FUNCTION
C**********************************************************************
C
C
C   *  BRANCH HERE WHEN ITMAX IS EXCEEDED
C
  140   WRITE (IPRINT,1025)
C
```

```
C
C   ** BRANCH HERE WHEN THE MAXIMUM NUMBER OF STEP SIZE
C   ** REDUCTIONS HAVE BEEN MADE
C
  150   CALL OBRES (FX,FY,FG,FH)
        CALL CKVIOL (FG,FEAS,ITER)
C
C  ** IS THE KTH SUB-OPTIMUM POINT FEASIBLE ?
  160   IF (FEAS) GO TO 170
C
C
C                        ** FIG. 5 **
C***********  PULL BACK THE INFEASIBLE STAGE-OPTIMUM  **************
C                INTO THE FEASIBLE REGION
C
  161   NOPULL=0
        PULL=0.63
C
C                * FIG. 5-1 *
C ** MOVE THE KTH SUB-OPTIMUM TOWARD THE (K-1)ST SUB-OPTIMUM
  162   DO 163 I=1,N
          FX(I) = PULL * ( FX(I)-OX(I) )  +  OX(I)
  163   CONTINUE
C
        NOPULL = NOPULL + 1
        CALL OBRES (FX,FY,FG,FH)
        CALL CKVIOL (FG,FEAS,ITER)
        NOITB = NOITB + 1
C
C                * FIG. 5-2 *
C  ** IS THE STAGE OPTIMUM POINT NOW FEASIBLE ?
        IF (FEAS) GO TO 170
C
C                * FIG. 5-3 *
        IF (NOPULL.LT.5) GO TO 162
C
C                * FIG. 5-4 *
C  **  SET THE KTH SUB-OPTIMUM EQUAL TO THE (K-1)ST SUB-OPTIMUM POINT
  165   DO 166 I=1,N
          FX(I) = OX(I)
  166   CONTINUE
C
        CALL OBRES (FX,FY,FG,FH)
        CALL CKVIOL (FG,FEAS,ITER)
C
C                 *  END OF PROCEDURE  *
C          FOR PULLING BACK THE INFEASIBLE STAGE OPTIMUM POINT
C*********************************************************************
C
C
C********  OUTPUT THE RESULTS AT THE KTH SUB-OPTIMUM POINT  *********
C
  170   CALL PENAT (FG,FH,PENA1,PENA2)
        FP = FY + R * PENA1 + R**(-0.5) * PENA2
        NOIT = NOIT + ITER
```

```
      YSTOP = ABS( FY / ( FY-R*PENA1 + R**(-0.5) * PENA2) )
      YSTOP = ABS( YSTOP-1.0 )
C
      WRITE (ICONS,1007)
      WRITE (ICONS,1008) NSTAGE,FY,FP,R,ITER,NOIT,NOITB,NOFEAS,NCBP,
     1    NOEXP,NOPAT,NOCUT,YSTOP
      WRITE (IPRINT,1008) NSTAGE,FY,FP,R,ITER,NOIT,NOITB,NOFEAS,NCBP,
     1    NOEXP,NOPAT,NOCUT,YSTOP
      WRITE (ICONS,1006)  ( I, FX(I), I, D(I), I=1,N )
      WRITE (IPRINT,1006) ( I, FX(I), I, D(I), I=1,N )
      WRITE (ICONS, 1011)
      WRITE (IPRINT,1011)
      IF (MG) 216,216,215
  215    WRITE (ICONS, 1012) ( J, FG(J), J=1,MG )
         WRITE (IPRINT,1012) ( J, FG(J), J=1,MG )
C
  216 IF (MH) 218,218,217
  217    WRITE (ICONS, 1013) ( K, FH(K), K=1,MH )
         WRITE (IPRINT,1013) ( K, FH(K), K=1,MH )
C
C   **OUTPUT  ADDITIONAL INFORMATION DESIRED BY USER
  218 CALL OUTPUT (FX,FY,FG,FH)
      WRITE (IPRINT,1007)
C
C   **CHECK IF THE FINAL STOPPING CRITERION IS  SATISFIED
      IF (YSTOP-THETA) 230,230,220
C
C   **CHECK IF MAXP IS EXCEEDED
  220 IF (NSTAGE-MAXP) 221,232,232
C
C  **STORE THE LAST SUB-OPTIMUM POINT
  221 DO 222 I=1,N
        D(I) = PD(I)
        OX(I) = FX(I)
  222 CONTINUE
C
C
C***********  SHIFT TO THE NEXT SUBPROBLEM SEARCH *****************
      R = R / RATIO
      FP = FY + R * PENA1 + R**(-0.5) * PENA2
      NSTAGE = NSTAGE + 1
      IF (NCBP.GT.0) INCUT = INCUT + 1
      NCBP = 0
      NOITB = 0
      NOFEAS=0
      NOEXP=0
      NOPAT=0
      NOCUT=0
      ITER=0
      B=0.0
C
C   **DECIDE THE INITIAL STEP-SIZES AND TOLERANCE LIMIT
      IF (ICUT) 227,227,229
  227    DO 228 I=1,N
            D(I) = PD(I) / NSTAGE
```

```
               B = B + 0.5 * D(I)
  228       CONTINUE
            B = B / N
            GO TO 50
C
  229     B = PB
          GO TO 50
C
  230     WRITE (ICONS,1015)
          WRITE (IPRINT,1015)
          GO TO 236
C
  232     WRITE (ICONS,1016) MAXP
          WRITE (IPRINT,1016) MAXP
C
  236     STOP
          END
C
C
C
C                         **  FIG. 4  **
C*****************  MOVE BACK PROCEDURE  ***************************
C
          SUBROUTINE BACK (X,D,Y,G,H,NOITB,B,ISKIP,ITER)
C
C            THIS SUBROUTINE PULLS THE INFEASIBLE POINT BACK INTO THE
C         FEASIBLE OR NEAR-FEASIBLE REGION.
C
C           **DEFINITION ..
C                FEASIBLE .. ALL G(I) .GE. 0
C                NEAR-FEASIBLE ..  TGH .LE. B
C
          LOGICAL  EXPSUC, FEAS
          INTEGER*1  NB
          INTEGER  ISKIP, ITER, ITERB, ITMAX
          INTEGER  MG, MH, MXBACK, N, NOITB
          REAL   D(20), G(20), H(20), X(20)
          REAL   B, FRAC, FTGH, TGH, XOLD, Y
          COMMON  /BLOGY/ ITMAX, MG, MH, N
C
C         MXBACK IS THE MAXIMUM NUMBER OF ITERATIONS TO BE MADE BEFORE
C         EXITING THIS ROUTINE.  IF MXBACK IS EXCEEDED, A PREMATURE EXIT
C         FROM THIS ROUTINE WILL BE MADE LEAVING THE POINT STILL
C         INFEASIBLE.  THE VARIABLE ISKIP WILL BE SET TO 1 TO FLAG THIS
C         CONDITION.
C
          MXBACK = 4*N
          ITERB = ITER + MXBACK
          ISKIP = 0
          FRAC = 0.5
C
C                * FIG. 4-1 *
C  ** COMPUTE THE WEIGHT OF VIOLATION
          CALL WEIGH (G,H,TGH)
C
```

```
C                 * FIG. 4-2 *
C  ** CHECK IF THE POINT IS IN THE NEAR-FEASIBLE REGION
    4    IF (TGH.LE.B)  GO TO 57
C
         FTGH = TGH
C
C                 * FIG. 4-3 *
C    **MAKE EXPLORATORY MOVES FOR MINIMIZING TGH
   22    EXPSUC = .FALSE.
C
         DO 38 NB=1,N
           XOLD = X(NB)
           X(NB) = XOLD - FRAC * D(NB)
           CALL CBRES (X,Y,G,H)
           CALL CKVIOL (G,FEAS,ITER)
           CALL WEIGH (G,H,TGH)
           IF (FEAS) GO TO 46
C
           NOITB = NOITB + 1
           IF (TGH-FTGH) 37,32,32
C
  32       X(NB) = XOLD +  FRAC * D(NB)
           CALL CBRES (X,Y,G,H)
           CALL CKVIOL (G,FEAS,ITER)
           CALL WEIGH (G,H,TGH)
           IF (FEAS)  GO TO 46
C
           NOITB = NOITB + 1
           IF (TGH-FTGH) 37,36,36
C
  36       X(NB) = XOLD
           GO TO 38
C
  37       EXPSUC = .TRUE.
           FTGH=TGH
           IF (TGH.LE.B)  GO TO 46
C
  38     CONTINUE
C
         IF (ITER.GE.ITMAX) GO TO 60
C
C                 * FIG. 4-4 *
C   ** DID EXPLORATORY MOVES MAKE PROGRESS ?
         IF (EXPSUC) GO TO 22
C
  42     IF (ITER - ITERB) 44,43,59
C
C                 * FIG. 4-5 *
C   ** INCREASE STEP SIZES
  43     FRAC = FRAC * 5.0
         GO TO 22
C
  44     FRAC = FRAC * 1.5
         GO TO 22
C
```

```
C    ** REDUCE STEP SIZE TO HELP PREVENT EXPLORATORY MOVES BACK INTO
C    ** INFEASIBLE REGION
  46     DO 50 I=1,N
           D(I) = D(I) * 0.55
  50     CONTINUE
C
C                  * FIG. 4-6 *
C    **DECREASE THE VALUE OF B
   57    IF ( TGH .LT. 0.7*B )  B = 0.75 * B
         GO TO 60
C
C   ** WHEN MXBACK IS EXCEEDED BEFORE A FEASIBLE POINT IS FOUND,
C      SET ISKIP = 1  BEFORE LEAVING THE SUBROUTINE
C
  59     ISKIP = 1
C
  60     RETURN
         END
C
C
         SUBROUTINE PENAT (G,H,PENA1,PENA2)
C
C           THIS SUBROUTINE COMPUTES THE PENALTY TERMS FOR SUMT FORMULATION
C                PENA1 FOR INEQUALITY CONSTRAINTS
C                PENA2 FOR EQUALITY CONSTRAINTS
C
C
         INTEGER  ITMAX, MG, MH, N
         REAL  G(20), H(20), PENA1, PENA2
         COMMON /BLOGY/ ITMAX, MG, MH, N
C
         PENA1 = 0.0
         PENA2 = 0.0
C
         IF (MG) 5,5,1
   1        DO 4 I=1,MG
              IF ( ABS( G(I) ) .LE. 0.1E-8 )  G(I) = 0.1E-08
              PENA1 = PENA1 + ABS (1.0 / G(I) )
   4        CONTINUE
C
   5     IF (MH) 10,10,6
   6        DO 9 K=1,MH
              PENA2 = PENA2 + H(K)**2
   9        CONTINUE
C
  10     RETURN
         END
C
C
```

```
      SUBROUTINE WEIGH (G,H,TGH)
C
C        THIS SUBROUTINE COMPUTES THE TOTAL WEIGHT OF VIOLATION
C     TO THE INEQUALITY AND EQUALITY CONTRAINTS.
C
      INTEGER*1  I
      INTEGER  ITMAX, MG, MH, N
      REAL  G(20), H(20), TGH
      COMMON /BLOGY/ ITMAX, MG, MH, N
C
      TGH = 0.0
      IF (MG.LE.0) GO TO 4
         DO 3 I=1,MG
           IF ( G(I).GE.0.0 ) GO TO 3
           TGH = TGH + G(I)**2
   3     CONTINUE
C
   4  IF (MH.LE.0) GO TO 8
         DO 7 I=1,MH
           IF ( H(I).EQ.0.0 ) GO TO 7
           TGH = TGH + H(I)**2
   7     CONTINUE
C
   8  IF (TGH.LT.0.0) TGH = 0.0
      TGH = SQRT(TGH)
C
      RETURN
      END
C
C
      SUBROUTINE CKVIOL (G,FEAS,ITER)
C
C         THIS SUBROUTINE CHECKS FOR ANY VIOLATION TO THE INEQUALITY
C     CONSTRAINTS AND ALSO UPDATES THE ITERATION COUNT. IT IS CALLED
C     AFTER EACH CALL TO SUBROUTINE CBRES.
C
      LOGICAL FEAS
      INTEGER*1  I
      INTEGER  ITER, ITMAX, MG, MH, N
      REAL  G(20)
      COMMON  /BLOGY/ ITMAX, MG, MH, N
C
      FEAS = .TRUE.
      ITER = ITER + 1
C
      IF (MG.EQ.0) GO TO 10
         DO 9 I=1,MG
            IF ( G(I).GE.0.0 ) GO TO 9
               FEAS = .FALSE.
               GO TO 10
    9    CONTINUE
C
   10 RETURN
      END
C
```

```
      SUBROUTINE INPUT ( R, RATIO, INCUT, THETA, ICUT, X, D )
C
C
      LOGICAL NAME(50)
      INTEGER*1  I
      INTEGER  ICONS, ICUT, INCUT, IPRINT, ISIZE, ITMAX
      INTEGER  MG, MH, N, OPTION
      REAL  X(20), D(20), R, RATIO, THETA, TZER
      COMMON /BLOGY/ ITMAX, MG, MH, N
      COMMON /INOUT/ ICONS, IPRINT
      DATA  TZER /1.0E-5/
C
      WRITE (ICONS,199)
      WRITE (IPRINT,199)
      WRITE (ICONS,198)
      WRITE (IPRINT,198)
      WRITE (ICONS,197)
      READ (ICONS,196) NAME
      WRITE (IPRINT,195) NAME
C
      WRITE (ICONS,194)
      READ (ICONS,193) N
      WRITE (IPRINT,190) N
C
      WRITE (ICONS,189)
      READ (ICONS,193)  MG
      WRITE (IPRINT,188) MG
C
      WRITE (ICONS,187)
      READ (ICONS,193)  MH
      WRITE (IPRINT,186) MH
C
C
         WRITE (ICONS,182)
      DO 50 I=1,N
         WRITE (ICONS,177) I
         READ (ICONS,176) X(I)
   50 CONTINUE
C
      WRITE (ICONS,175)
      READ (ICONS,174) ISIZE
      IF (ISIZE.EQ.1) GO TO 80
C
         DO 70 I=1,N
            D(I) = 0.02 * X(I)
            IF ( ABS( D(I) ).LE.TZER )  D(I) = 0.01
   70    CONTINUE
         GO TO 100
C
   80    WRITE (ICONS,171)
         DO 90 I=1,N
            WRITE (ICONS,173) I
            READ (ICONS,172) D(I)
   90    CONTINUE
C
```

```
C     DEFAULT VALUES OF THE INPUT PARAMETERS
C
  100   ITMAX = 100
        ICUT = 0
        R = 0.0
        RATIO = 4.0
        INCUT = 4
        THETA = 0.0001
C
        WRITE (ICONS,183)
        WRITE (ICONS,184)
        READ (ICONS,185) OPTION
        IF (OPTION.EQ.1) GO TO 130
           WRITE (ICONS,160)
           WRITE (IPRINT,160)
           WRITE (IPRINT,178) ITMAX
           RETURN
C
  130   WRITE (ICONS,180)
        READ (ICONS,179) ITMAX
        IF (ITMAX.LE.0)  ITMAX  = 100
        WRITE (IPRINT,178) ITMAX
C
        WRITE (ICONS,167)
        READ (ICONS,166) R
C
        WRITE (ICONS,165)
        READ (ICONS,166) RATIO
        IF (RATIO .LT. 2.0)  RATIO = 4.0
C
        WRITE (ICONS,164)
        READ (ICONS,163) INCUT
        IF (INCUT.LE.0)  INCUT = 4
C
        WRITE (ICONS,162)
        READ (ICONS,166) THETA
        IF (THETA.LE.0.0)  THETA = 0.0001
C
C
  199   FORMAT (/,31X,'KSU SUMT  PROGRAM')
  198   FORMAT (/,11X,30('* ') )
  197   FORMAT (/,9X, 'PROBLEM NAME : ')
  196   FORMAT (50A1)
  195   FORMAT (/,13X,50A1)
  194   FORMAT (/,9X,'NUMBER OF VARIABLES : ')
  193   FORMAT (I3)
  192   FORMAT (I1)
  191   FORMAT (I2)
  190   FORMAT (/,21X,'NO. OF X(I) ...    ',4X, I3)
  189   FORMAT (' ',8X,'NUMBER OF INEQUALITY CONSTRAINTS',
     1          ' ( G(X) >= 0 ) : ')
  188   FORMAT (' ',20X,'NO. OF G(J) >= 0  ...  ',I2)
  187   FORMAT (' ',8X,'NUMBER OF EQUALITY CONSTRAINTS ( H(X) = 0 ) : ')
  186   FORMAT (' ',20X,'NO. OF H(J) = 0   ...  ',I2)
C
```

```
  185   FORMAT (I3)
  184   FORMAT (/,8X,'TO USE ALL DEFAULT VALUES (ENTER 0) ' /
    1           8X,'TO SPECIFY OWN VALUES (ENTER 1)    :  ')
  183   FORMAT (' ',5X,'THE DEFAULT VALUES FOR THE FOLLOWING ',
    1           'PARAMETERS ARE SHOWN BELOW : ' //
    2           8X,'ITMAX --- THE MAX. NO. OF ITERATIONS AT EACH ',
    3           'STAGE   = 100' /
    4           8X, 'R --- PENALTY COEFFICIENT ',
    5                 ' = Y / SUM( 1.0 /G(I) ) ' /
    6           8X, 'RATIO --- REDUCING FACTOR = 4.0 ' /
    7           8X, 'INCUT --- NUMBER OF CUT-DOWN STEP SIZE ',
    8           'OPERATIONS   = 4 ' /
    9           8X, 'THETA ---  FINAL STOPPING CRITERION   = 0.0001 ')
C
  182   FORMAT (/,16X,'ENTER THE INITIAL POINT :' //)
  181   FORMAT (' ',8X,'X(',I2,') = ',G12.4)
  180   FORMAT (' ',7X,'MAX. NO. OF ITERATIONS AT EACH STAGE ' /
    1           8X,' ( PRESS RETURN FOR DEFAULT OF 100 ) ' /
    2           8X,'ITMAX =  ')
  179   FORMAT (I5)
  178   FORMAT (/,11X,'MAX. NO. OF ITERATIONS AT EACH STAGE   ... ',I5)
  177   FORMAT ('+',8X,' X(',I2,') =  ')
  176   FORMAT (F15.0)
  175   FORMAT (' ',8X,'WOULD YOU LIKE TO SPECIFY THE STEP-SIZE ',
    1            '( ENTER 1 ) ' /  5X,'OR  USE COMPUTED VALUE ',
    2           ' D(I) = 0.02 * X(I)   ( ENTER 2 ) :  ')
  174   FORMAT (I1)
  173   FORMAT ('+',8X,'D(',I2,') = ')
  172   FORMAT (F15.0)
  171   FORMAT (5X,' ')
  167   FORMAT (' ',7X,'R --- PENALTY COEFFICIENT FOR SUMT FORMULATION'
    1           /  8X,'PRESS RETURN TO USE A COMPUTED VALUE ',
    2           ' R = Y / SUM( 1.0/G(I) ) ' /  8X,'R = ')
  166   FORMAT (F15.0)
  165   FORMAT (' ',7X,'RATIO --- REDUCING FACTOR FOR R FROM STAGE ',
    1           'TO STAGE' /  8X, 'PRESS RETURN TO USE DEFAULT VALUE',
    2           ' OF 4.0  '/  8X,'RATIO = ')
  164   FORMAT (' ',7X,'INCUT --- NUMBER OF CUT-DOWN STEP-SIZE ',
    1      'OPERATIONS IN' /20X,'HOOKE AND JEEVES SEARCH TECHNIQUE'/
    2           8X,'PRESS RETURN FOR DEFAULT OF 4 ' /
    3           8X,'INCUT = ')
  163   FORMAT (I1)
  162   FORMAT (' '7X,'THETA --- FINAL STOPPING CRITERION ' /
    1           8X,' ( SUGGESTED VALUES ARE :  0.01, 0.001, 0.0001, ',
    2           '0.00001, 0.000001 )' /
    3           8X, 'PRESS RETURN FOR DEFAULT VALUE OF 0.0001' /
    4           8X,'THETA = ')
C
  160   FORMAT (/,9X,'DEFAULT VALUES CHOSEN')
C
        RETURN
        END
```

3.8.5 DESCRIPTION OF OUTPUT

The program title is printed followed by the name of the problem to be solved. Then the number of variables, inequality constraints and equality constraints are printed. The specified maximum number of iterations at each stage are printed last.

Following a row of asterisks the user supplied values of the parameters are printed along with the starting point and values of the constraints at the starting point. An explanation of the variables printed at the initial point follows.

Y --- F function value at the initial point

P --- P function value at the initial point

R --- penalty coefficient for SUMT formulation (computed or user supplied)

RATIO --- reducing factor for R; $r_{k+1} = r_k$ / RATIO. (User-supplied)

B --- tolerance limit of constraint violation.

INCUT --- maximum number of step size reductions for a fixed r. This is used as a subproblem stopping criterion. (User-supplied).

THETA --- final stopping criterion value. (User-supplied).

X(I) --- the starting point. (User-supplied).

D(I) --- the starting step size. (User-supplied).

G(I) --- the inequality constraint values at the starting point.

H(I) --- the equality constraint values at the starting point.

If the user supplied initial point was infeasible, the program will next print a feasible starting point if one can be found. If the input starting point was feasible, then the results at each of the subproblem optimum points are printed.

The first line tells how many subproblem (P optimum) points have been solved. The explanation of the varibles printed at each P optimum point follows.

FY --- the F function value at the P optimum point.

FP --- the minimum P function value for the subproblem.

R --- the penalty coefficient for SUMT formulation used at the subproblem.

ITER --- the number of F function values computed for the subproblem.

NOIT --- the total number of F function values computed since the start of the program. (the cumulative ITER count).

NOITB --- the number of exploratory moves made in the infeasible region.

NOFEAS --- the number of exploratory and pattern moves made in the feasible region.

NOBP --- number of times subroutine BACK is called.

NOEXP --- number of successful series of exploratory moves where a series of exploratory moves occurs when step sizes have been taken in all dimensions.

NOPAT --- number of successful pattern moves

NOCUT --- number of step size reductions for the subproblem. This may be less than the maximum specified if the maximum number of iterations is exceeded. It may also exceed the maximum specified if a subproblem is considered too flat in that more step size cuts are needed to get a more appropriate step size.

YSTOP --- computed value of the final stopping criterion. This value must be less than or equal to THETA to satisfy the final stopping criterion.

X(I) --- the P optimum point for the subproblem

D(I) --- the final step size used before terminating the subproblem search.

G(I) --- the inequality constraints at the P optimum point.

H(I) --- the equality constraint values at the P optimum point

In addition to the above values, a message is printed out if the subproblem search was stopped because the maximum number of iterations was reached.

3.8.6 SUMMARY OF USER REQUIREMENTS

1. Create a file on disk that contains both subroutine OBRES and subroutine OUTPUT.

2. Choose a point to be used as the starting point. A feasible point should be used if possible although the program will attempt to locate a feasible point if one is not given.

3. Determine the initial step size and the final step size. Compute INCUT as the number of times the initial step size must be reduced by 1/2 to get the final step size.

Note : The following steps will vary depending on the particular compiler used. The following applies if using Microsoft Fortran-80 for the North Star microcomputer.

4. Compile subroutine OBRES and OUTPUT using the F80 command

```
F80 =B:filename
```

where filename is the name of the file containing the two subroutines and the letter B is the disk drive where the file resides.

5. Run the program using the L80 command

```
L80 B:filename,B:KSUMT/G
```

Note : If several runs of the problem are to be made using different starting points and/or parameter values for each run, then the following two steps should be used instead of step 5.

6. Link edit the main program with the user supplied subroutines as follows

```
L80 B:filename,B:KSUMT/N,B:KSUMT/E
```

Note the order of the user supplied filename and the main program KSUMT. This order should not be reversed. The above statement link edits the two files and creates an executable file with a filename of KSUMT.COM.

7. Run the program by simply typing the filename of the executable file

```
B:KSUMT
```

To run the program again for a different starting point or parameter, simply repeat either step 5 or step 7 depending on which was used previously.

3.8.7 USER-SUPPLIED SUBROUTINES

Both of the user-supplied subroutines must contain a declaration statement :

```
REAL  X(20), G(20), H(20)
```

The following problem is used to show how to code the user-supplied subroutines.

Minimize $f(x) = x_1^2 + x_2^3 - x_1 x_2$

subject to

$$g_1(x) = 8x_1 + x_2^2 - 15 \geq 0$$

$$g_2(x) = 5x_1^4 + x_2^3 - 20 \geq 0$$

$$h_1(x) = x_1^2 + x_2^2 - 25 = 0$$

$$x_i \geq 0 \ , \quad i=1,2$$

OBRES (X,Y,G,H)

This subroutine defines the objective function Y (to be minimized), the inequality constraints ($g_j(x) \geq 0$), and the equality constraints ($h_j(x) = 0$). The equations are defined in terms of x_i. To transfer data from this subroutine to subroutine OUTPUT, blank COMMON may be used.

The OBRES routine for the example problem is shown below.

```
      SUBROUTINE OBRES (X,Y,G,H)
C
C     THIS ROUTINE DEFINES THE OBJECTIVE FUNCTION (TO BE MINIMIZED) AND
C     THE CONSTRAINTS ( >=0 AND =0 ).
C
      REAL  X(20), G(20), H(20), Y
      COMMON  VAL1
C
      VAL1 = X(1)*X(2)
      Y = X(1)**2 + X(2)**3 - VAL1
C
      G(1) = 8.*X(1) + X(2)**2 - 15.
      G(2) = 5.*X(1)**4 + X(2)**3 - 20.
      G(3) = X(1)
      G(4) = X(2)
      G(5) = X(3)
C
      H(1) = X(1)**2 + X(2)**2 - 25.
C
      RETURN
      END
```

OUTPUT (X,Y,G,H)

This subroutine is used to print out additional information desired by the user. If there is nothing to print out, simply code the subroutine name, the dimension statement, and a RETURN and END. This subroutine is called after printing out the results at each subproblem optimum point. To transfer data from subroutine OBRES to this routine, blank COMMON may be used.

The user must provide the WRITE and FORMAT statements necessary to print out the additional data desired. The logical unit number for the WRITE statement is a 1 for the CRT screen and a 2 for the printer. For example, to display information on the CRT screen, the following statements would be used

```
         WRITE (1,99) INFO
   99    FORMAT (2X,'INFO =',I2)
```

The logical unit number is different for different compilers. Please check the Fortran user manual for the proper values. The above values are appropriate for Microsoft's Fortran-80 for the North Star microcomputer.

To illustrate the above for the example problem, VAL1 has been passed into OUTPUT from subroutine OBRES using blank COMMON. VAL1 is then displayed on the CRT screen. VAL2 is computed in the routine and sent to the printer.

The OUTPUT routine for the example problem is shown below :

```
      SUBROUTINE OUTPUT (X,Y,G,H)
C
C     THIS SUBROUTINE PRINTS OUT ADDITIONAL INFORMATION
C     DESIRED BY THE USER.
C
      REAL  X(20), G(20), H(20), Y
      COMMON  VAL1
C
      WRITE (1,99) VAL1
C
      VAL2 = G(1) + G(2)
      WRITE (2,98) VAL2
C
   99 FORMAT (5X,'VAL1 =',F9.2)
   98 FORMAT (2X,'VAL2 =',F12.5)
C
      RETURN
      END
```

3.9 INPUT TO THE COMPUTER PROGRAM

3.9.1 CRT DISPLAY OF QUESTIONS

```
                          KSU SUMT  PROGRAM
     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

      PROBLEM NAME :

      NUMBER OF VARIABLES :

      NUMBER OF INEQUALITY CONSTRAINTS ( G(X) >= 0 ) :

      NUMBER OF EQUALITY CONSTRAINTS ( H(X) = 0 ) :

             ENTER THE INITIAL POINT :

       X( 1) =
       X( 2) =
          .
          .
       X( N) =


      WOULD YOU LIKE TO SPECIFY THE STEP-SIZE ( ENTER 1 )
   OR USE COMPUTED VALUE  D(I) = 0.02 * X(I)   ( ENTER 2 ) :  1

      D( 1) =
      D( 2) =
         .
         .
      D( N) =


    THE DEFAULT VALUES FOR THE FOLLOWING PARAMETERS ARE SHOWN BELOW :
      ITMAX ---- THE MAX. NO. OF ITERATIONS AT EACH STAGE  = 100
      R --- PENALTY COEFFICIENT  = Y / SUM( 1.0 /G(I) )
      RATIO --- REDUCING FACTOR = 4.0
      INCUT --- NUMBER OF CUT-DOWN STEP SIZE OPERATIONS  = 4
      THETA ---  FINAL STOPPING CRITERION  = 0.0001

      TO USE ALL DEFAULT VALUES (ENTER 0)
      TO SPECIFY OWN VALUES (ENTER 1)   :  1


      MAX. NO. OF ITERATIONS AT EACH STAGE
       ( PRESS RETURN FOR DEFAULT OF 100 )
      ITMAX =


      R --- PENALTY COEFFICIENT FOR SUMT FORMULATION
      PRESS RETURN TO USE A COMPUTED VALUE  R = Y / SUM( 1.0/G(I) )
      R =
```

```
RATIO --- REDUCING FACTOR FOR R FROM STAGE TO STAGE
PRESS RETURN TO USE DEFAULT VALUE OF 4.0
RATIO =


INCUT --- NUMBER OF CUT-DOWN STEP-SIZE OPERATIONS IN
             HOOKE AND JEEVES SEARCH TECHNIQUE
PRESS RETURN FOR DEFAULT OF 4
INCUT =


THETA --- FINAL STOPPING CRITERION
 ( SUGGESTED VALUES ARE :  0.01, 0.001, 0.0001, 0.00001, 0.000001 )
PRESS RETURN FOR DEFAULT VALUE OF 0.0001
THETA =
```

3.9.2 NOTES ABOUT THE INPUT

The maximum size problem that can be solved is 20 variables, 20 inequality constraints, and 20 equality constraints. To solve a larger problem, the dimensions in the main program must be modified. For the key to the changes, see section 3.8.2 PROGRAM LIMITATIONS.

3.10 TEST PROBLEMS

3.10.1 TEST PROBLEM 1 : NUMERIC EXAMPLE BY PAVIANI

3.10.1.1 SUMMARY

NO. OF VARIABLES : 3

NO. OF CONSTRAINTS : 1 nonlinear equality constraint
1 linear equality constraint
3 bounds on independent variables

OBJECTIVE FUNCTION :

Minimize $f(x) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$

CONSTRAINTS :

$$h_1(x) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$

$$h_2(x) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

$$x_i \geq 0 \qquad i = 1,2,3$$

STARTING POINT : $x_i = 2$ $\quad i = 1,2,3$

INITIAL STEP SIZE : $d_i = .05$ $\quad i = 1,2,3$

PARAMETERS : ITMAX = 200

r = 1.398 (computed value)

INCUT = 4

THETA = .1000E-04

RESULTS : $f(x) = 962.3$

$x_1 = 2.79$

$x_2 = 3.35$

$x_3 = 4.14$

$h_1(x) = 0.06$

$h_2(x) = 0.01$

NO. OF K ITERATED : 4

NO. OF FUNCTION EVALUATIONS : 432

	MICROCOMPUTER SINGLE PRECISION	LARGE COMPUTER DOUBLE PRECISION
EXECUTION TIME :	.42 min.	.02 min.

3.10.1.2 COMPUTER PRINTOUT OF RESULTS

```
                              KSU SUMT  PROGRAM

        * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

          TEST PROBLEM 1 : NUMERIC EXAMPLE BY PAVIANI

                    NO. OF X(I) ...          3
                    NO. OF G(J) >= 0   ...   3
                    NO. OF H(J) = 0    ...   2

        MAX. NO. OF ITERATIONS AT EACH STAGE  ...   200

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                    INITIAL POINT

 Y =    .9760E+03,  P =    .1124E+04,  R =    .1398E+01,  RATIO =    .4000E+01
 B =    .5000E-01,  INCUT =     4,  THETA =   .1000E-04 .

        X( 1) =    .200000E+01       D( 1) =    .500000E-01
        X( 2) =    .200000E+01       D( 2) =    .500000E-01
        X( 3) =    .200000E+01       D( 3) =    .500000E-01

    **CONSTRAINTS ..
        G( 1) =    .200000E+01 ,
        G( 2) =    .200000E+01 ,
        G( 3) =    .200000E+01 ,
        H( 1) =   -.130000E+02 ,
        H( 2) =    .200000E+01 ,

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

 ** PROBLEM MAY BE TOO FLAT --- R VALUE REDUCED AND INCUT VALUE INCREASED
    EXPLORATORY MOVES TAKEN IN ALL DIRECTIONS AT ONCE  SUCCESSFUL


  ** P OPTIMUM..  ( 1)
 FY =    .962096E+03,   FP =   .964700E+03,  R =    .6991E+00   ITER =   188
                    NOIT =   188, NOITB =     0,  NOFEAS =  177,  NOBP =    0
                    NOEXP =   21,  NOPAT =   12,  NOCUT =    5 .
                    YSTOP =   .232732E-02 .

        X( 1) =    .273750E+01       D( 1) =    .625000E-02
        X( 2) =    .350001E+00       D( 2) =    .625000E-02
        X( 3) =    .420625E+01       D( 3) =    .625000E-02

    **CONSTRAINTS ..
        G( 1) =    .273750E+01 ,
        G( 2) =    .350001E+00 ,
        G( 3) =    .420625E+01 ,
        H( 1) =    .308928E+00 ,
        H( 2) =    .243748E+00 ,
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

  ** P OPTIMUM..  ( 2)
 FY =   .962247E+03,   FP =   .963002E+03,  R =   .1748E+00   ITER =    63
                  NOIT =   251, NOITB =    0,  NOFEAS =   59,  NOBP =    0
                  NOEXP =    4,  NOPAT =    1,  NOCUT =    5 .
                  YSTOP =   .491858E-03 .

       X( 1) =    .272500E+01      D( 1) =    .312500E-02
       X( 2) =    .343751E+00      D( 2) =    .312500E-02
       X( 3) =    .420625E+01      D( 3) =    .312500E-02

    **CONSTRAINTS ..
       G( 1) =    .272500E+01 ,
       G( 2) =    .343751E+00 ,
       G( 3) =    .420625E+01 ,
       H( 1) =    .236311E+00 ,
       H( 2) =    .562477E-01 ,

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

  ** P OPTIMUM..  ( 3)
 FY =   .962292E+03,   FP =   .962515E+03,  R =   .4370E-01   ITER =   112
                  NOIT =   363, NOITB =    0,  NOFEAS =  105,  NOBP =    0
                  NOEXP =   12,  NOPAT =    6,  NOCUT =    5 .
                  YSTOP =   .931025E-04 .

       X( 1) =    .277708E+01      D( 1) =    .208333E-02
       X( 2) =    .335418E+00      D( 2) =    .208333E-02
       X( 3) =    .415833E+01      D( 3) =    .208333E-02

    **CONSTRAINTS ..
       G( 1) =    .277708E+01 ,
       G( 2) =    .335418E+00 ,
       G( 3) =    .415833E+01 ,
       H( 1) =    .116413E+00 ,
       H( 2) =    .208244E-01 ,

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

  ** P OPTIMUM..  ( 4)
 FY =   .962339E+03,   FP =   .962410E+03,  R =   .1092E-01   ITER =    69
                  NOIT =   432, NOITB =    0,  NOFEAS =   65,  NOBP =    0
                  NOEXP =    5,  NOPAT =    2,  NOCUT =    5 .
                  YSTOP =   .786781E-05 .

       X( 1) =    .278958E+01      D( 1) =    .156250E-02
       X( 2) =    .335418E+00      D( 2) =    .156250E-02
       X( 3) =    .414271E+01      D( 3) =    .156250E-02
```

```
**CONSTRAINTS ..
     G( 1) =    .278958E+01 ,
     G( 2) =    .335418E+00 ,
     G( 3) =    .414271E+01 ,
     H( 1) =    .562935E-01 ,
     H( 2) =    .114517E-01 ,

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

 ***** THE ABOVE RESULTS ARE THE FINAL OPTIMUM .
```

3.10.1.3 USER SUPPLIED SUBROUTINES

```
      SUBROUTINE OBRES (X,Y,G,H)
C
C       TEST PROBLEM 1 : NUMERIC EXAMPLE BY PAVIANI
C
      REAL  X(20), Y, G(20), H(20)
      REAL  X1, X2, X3
C
      X1 = X(1)
      X2 = X(2)
      X3 = X(3)
C
      Y =  1000.0  -  X1**2  - 2.0*X2**2  -  X3**2  - X1*X2  - X1*X3
C
      H(1) =  X1**2   + X2**2  + X3**2  - 25.0
      H(2) =  8.0 * X1  +  14.0 * X2  +  7.0 * X3  - 56.0
C
      G(1) = X1
      G(2) = X2
      G(3) = X3
C
      RETURN
      END
C
      SUBROUTINE  OUTPUT (X,Y,G,H)
      REAL  X(20), Y, G(20), H(20)
      RETURN
      END
```

3.10.2 TEST PROBLEM 2 : MAXIMIZING SYSTEMS RELIABILITY

3.10.1.2 SUMMARY

NO. OF VARIABLES : 4

NO. OF CONSTRAINTS : 1 inequality constraint
4 upper bounds on independent variables
4 lower bounds on independent variables

OBJECTIVE FUNCTION :

$$\text{Minimize} \quad f(x) = -1 + R_3[(1-R_1)(1-R_4)]^2 + (1-R_3)\{1 - R_2[1 - (1-R_1)(1-R_4)]\}^2$$

CONSTRAINTS :

$$g_1(x) = C - (2K_1R_1^{\alpha_1} + 2K_2R_2^{\alpha_2} + K_3R_3^{\alpha_3} + 2K_4R_4^{\alpha_4}) \geq 0$$

$$g_{i+1}(x) = 1 - R_i \geq 0 \qquad i = 1,2,3,4$$

$$g_{i+5}(x) = R_i - R_{i,min} \geq 0 \qquad i = 1,2,3,4$$

where $K_1 = 100 \quad K_2 = 100 \quad K_3 = 200 \quad K_4 = 150$

$C = 800 \quad \alpha_i = 0.6 \quad i = 1,2,3,4$

$R_{i,min} = 0.5 \quad i = 1,2,3,4$

STARTING POINT : $R_i = 0.6 \quad i = 1,2,3,4$

INITIAL STEP SIZE : $d_i = 0.05 \quad i = 1,2,3,4$

PARAMATERS : ITMAX = 200

r = .4412E-02 (computed value)

INCUT = 4

THETA = .1000E-03

RESULTS : $f(x) = 0.9955$

$R_1 = 0.7928$

$R_2 = 0.9172$

$R_3 = 0.8068$

$R_4 = 0.7882$

NO. OF K ITERATED : 6

NO. OF FUNCTION EVALUATIONS : 1048

	MICROCOMPUTER SINGLE PRECISION	LARGE COMPUTER DOUBLE PRECISION
EXECUTION TIME :	2.4 min.	.03 min.

3.10.2.2 DESCRIPTION OF THE PROBLEM

The problem of maximizing the reliability of the complex system given in Fig. 3.3 which is subject to a single constraint can be stated as follows [11,12,13]

Maximize the system reliability

$$\begin{aligned} R_s &= 1 - Q_s \\ &= 1 - R_3[(1 - R_1)(1 - R_4)]^2 \\ &\quad - (1 - R_3)\{1 - R_2[1 - (1 - R_1)(1 - R_4)]\}^2 \end{aligned}$$

subject to

$$C_s = \sum_i C_i \leq C \tag{3.5}$$

$$R_i \geq R_{i,min}$$

where

$$C_i = K_i R_i^{\alpha_i} \qquad i = 1,2,3,4 \tag{3.6}$$

The constraint given by eq. (3.5) can be interpreted as follows. C_i can represent the weight, cost, or volume of each unit or component of the system, and the total weight, cost, or volume of the system must be less than C. Each of these is a function of reliability that can be expressed by eq. (3.6) where K_i is a proportionality constant and α_i the exponential factor that relates C_i and the reliability. That is, K_i is the weight, cost, or volume of the component when $R_i = 1$ and $K_i R_i^{\alpha_i}$ is the reduced cost, weight, or volume when $R_i < 1$. Usually α_i is less than one. The following values are assigned to the constants K_1, K_2, K_3, and K_4, the constraint C, the exponential constant α_i, and the minimum reliability for each component $R_{i,min}$, $i = 1,2,3,4$.

$$K_1 = 100, \quad K_2 = 100, \quad K_3 = 200, \quad K_4 = 150,$$

$$C = 800, \quad \alpha_i = 0.6, \quad R_{i,min} = 0.5 \quad i = 1,2,3,4.$$

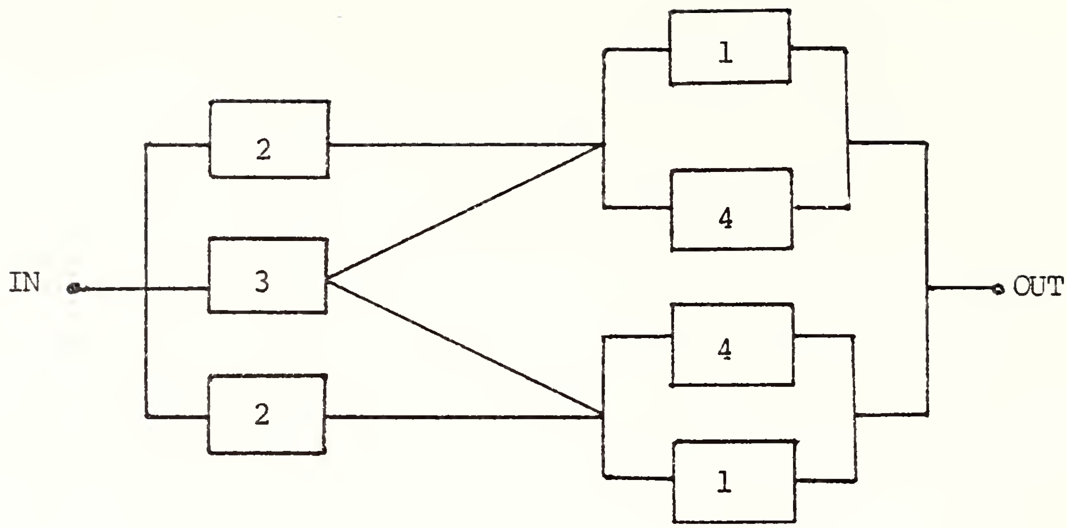

Figure 3.3 A schematic diagram of a complex system.

3.10.2.3 COMPUTER PRINTOUT OF RESULTS

```
                    KSU SUMT  PROGRAM

     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

       TEST PROBLEM 2 : MAXIMIZING SYSTEMS RELIABILITY

               NO. OF X(I) ...          4
               NO. OF G(J) >= 0   ...   9
               NO. OF H(J) = 0    ...   0

     MAX. NO. OF ITERATIONS AT EACH STAGE  ...   200

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

              INITIAL POINT

 Y =  -.8862E+00,  P =  -.6647E+00,  R =   .4431E-02,  RATIO =   .4000E+01
 B =   .5000E-01,  INCUT =     4,  THETA =   .1000E-03 .

       X( 1) =    .600000E+00       D( 1) =    .500000E-01
       X( 2) =    .600000E+00       D( 2) =    .500000E-01
       X( 3) =    .600000E+00       D( 3) =    .500000E-01
       X( 4) =    .600000E+00       D( 4) =    .500000E-01

  **CONSTRAINTS ..
       G( 1) =    .137580E+03 ,
       G( 2) =    .400000E+00 ,
       G( 3) =    .400000E+00 ,
       G( 4) =    .400000E+00 ,
       G( 5) =    .400000E+00 ,
       G( 6) =    .100000E+00 ,
       G( 7) =    .100000E+00 ,
       G( 8) =    .100000E+00 ,
       G( 9) =    .100000E+00 ,

   COST  =   662.42

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


  ** P OPTIMUM..  ( 1)
 FY =  -.987505E+00,   FP =  -.841031E+00,  R =    .4431E-02   ITER =   124
              NOIT =   124, NOITB =    6,  NOFEAS =  107,  NCBP =    1
              NOEXP =   10,  NOPAT =    2,  NCCUT =    4 .
              YSTOP =   .129168E+00 .

       X( 1) =    .777500E+00       D( 1) =    .171875E-02
       X( 2) =    .817187E+00       D( 2) =    .171875E-02
       X( 3) =    .787969E+00       D( 3) =    .171875E-02
       X( 4) =    .777656E+00       D( 4) =    .171875E-02
```

```
**CONSTRAINTS ..
      G( 1) =    .195078E+02 ,
      G( 2) =    .222500E+00 ,
      G( 3) =    .182813E+00 ,
      G( 4) =    .212031E+00 ,
      G( 5) =    .222344E+00 ,
      G( 6) =    .277500E+00 ,
      G( 7) =    .317187E+00 ,
      G( 8) =    .287969E+00 ,
      G( 9) =    .277656E+00 ,

  COST  =   780.49

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


 ** P OPTIMUM..  ( 2)
FY =  -.993208E+00,   FP =  -.953826E+00,  R =   .1108E-02   ITER =   100
                NOIT =   224, NOITB =     5,  NOFEAS =    87,  NCBP =    1
                NOEXP =     6,  NOPAT =    0,  NOCUT =     5 .
                YSTOP =   .381396E-01 .

      X( 1) =    .806797E+00       D( 1) =    .429687E-03
      X( 2) =    .866250E+00       D( 2) =    .429687E-03
      X( 3) =    .809531E+00       D( 3) =    .429687E-03
      X( 4) =    .788906E+00       D( 4) =    .429687E-03

 **CONSTRAINTS ..
      G( 1) =    .427661E+01 ,
      G( 2) =    .193203E+00 ,
      G( 3) =    .133750E+00 ,
      G( 4) =    .190469E+00 ,
      G( 5) =    .211094E+00 ,
      G( 6) =    .306797E+00 ,
      G( 7) =    .366250E+00 ,
      G( 8) =    .309531E+00 ,
      G( 9) =    .288906E+00 ,

  COST  =   795.72

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

** SUBPROBLEM SEARCH TERMINATED BECAUSE ITERATION MAXIMUM EXCEEDED **


 ** P OPTIMUM..  ( 3)
FY =  -.993752E+00,   FP =  -.983683E+00,  R =   .2769E-03   ITER =   203
                NOIT =   427, NOITB =   164,  NOFEAS =    25,  NCBP =   14
                NOEXP =     1,  NOPAT =    0,  NOCUT =     2 .
                YSTOP =   .100304E-01 .

      X( 1) =    .817297E+00       D( 1) =    .319257E-05
      X( 2) =    .866250E+00       D( 2) =    .319257E-05
      X( 3) =    .820031E+00       D( 3) =    .319257E-05
      X( 4) =    .788906E+00       D( 4) =    .319257E-05
```

```
**CONSTRAINTS ..
     G( 1) =    .153925E+01 ,
     G( 2) =    .182703E+00 ,
     G( 3) =    .133750E+00 ,
     G( 4) =    .179969E+00 ,
     G( 5) =    .211094E+00 ,
     G( 6) =    .317297E+00 ,
     G( 7) =    .366250E+00 ,
     G( 8) =    .320031E+00 ,
     G( 9) =    .288906E+00 ,

  COST  =   798.46

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

 ** PROBLEM MAY BE TOO FLAT --- R VALUE REDUCED AND INCUT VALUE INCREASED
    EXPLORATORY MOVES TAKEN IN ALL DIRECTIONS AT ONCE  SUCCESSFUL

 ** SUBPROBLEM SEARCH TERMINATED BECAUSE ITERATION MAXIMUM EXCEEDED **


 ** P OPTIMUM..  ( 4)
FY =  -.995522E+00,   FP =  -.993988E+00,  R =   .3461E-04   ITER =   204
                 NOIT =   631, NOITB =   22,  NOFEAS = 164,  NCBP =   12
                 NOEXP =   14,  NCPAT =   10,  NOCUT =     0 .
                 YSTOP =   .153822E-02 .

     X( 1) =    .792833E+00       D( 1) =    .761217E-03
     X( 2) =    .917194E+00       D( 2) =    .761217E-03
     X( 3) =    .806850E+00       D( 3) =    .761217E-03
     X( 4) =    .788186E+00       D( 4) =    .761217E-03

**CONSTRAINTS ..
     G( 1) =    .201355E+00 ,
     G( 2) =    .207167E+00 ,
     G( 3) =    .828061E-01 ,
     G( 4) =    .193150E+00 ,
     G( 5) =    .211814E+00 ,
     G( 6) =    .292833E+00 ,
     G( 7) =    .417194E+00 ,
     G( 8) =    .306850E+00 ,
     G( 9) =    .288186E+00 ,

  COST  =   799.80

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

 ** SUBPROBLEM SEARCH TERMINATED BECAUSE ITERATION MAXIMUM EXCEEDED **


 ** P OPTIMUM..  ( 5)
FY =  -.995522E+00,   FP =  -.995138E+00,  R =   .8653E-05   ITER =   207
                 NOIT =   838, NOITB =  181,  NOFEAS =    9,  NCBP =    9
                 NOEXP =    1,  NOPAT =     0,  NOCUT =     1 .
```

```
                    YSTOP =   .384986E-03 .

          X( 1) =     .792833E+00       D( 1) =     .167468E-03
          X( 2) =     .917194E+00       D( 2) =     .167468E-03
          X( 3) =     .806850E+00       D( 3) =     .167468E-03
          X( 4) =     .788186E+00       D( 4) =     .167468E-03

     **CONSTRAINTS ..
          G( 1) =     .201355E+00 ,
          G( 2) =     .207167E+00 ,
          G( 3) =     .828061E-01 ,
          G( 4) =     .193150E+00 ,
          G( 5) =     .211814E+00 ,
          G( 6) =     .292833E+00 ,
          G( 7) =     .417194E+00 ,
          G( 8) =     .306850E+00 ,
          G( 9) =     .288186E+00 ,

      COST  =   799.80

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 

 ** SUBPROBLEM SEARCH TERMINATED BECAUSE ITERATION MAXIMUM EXCEEDED **


    ** P OPTIMUM..  ( 6)
  FY =  -.995522E+00,   FP =  -.995426E+00,  R =   .2163E-05   ITER =   210
                    NOIT =  1048, NOITB =  180,  NOFEAS =   14,  NOBP =   10
                    NOEXP =    2,  NOPAT =    0,  NOCUT =    0 .
                    YSTOP =   .962615E-04 .

          X( 1) =     .792833E+00       D( 1) =     .153512E-03
          X( 2) =     .917194E+00       D( 2) =     .153512E-03
          X( 3) =     .806850E+00       D( 3) =     .153512E-03
          X( 4) =     .788186E+00       D( 4) =     .153512E-03

     **CONSTRAINTS ..
          G( 1) =     .201355E+00 ,
          G( 2) =     .207167E+00 ,
          G( 3) =     .828061E-01 ,
          G( 4) =     .193150E+00 ,
          G( 5) =     .211814E+00 ,
          G( 6) =     .292833E+00 ,
          G( 7) =     .417194E+00 ,
          G( 8) =     .306850E+00 ,
          G( 9) =     .288186E+00 ,

      COST  =   799.80

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 

  ***** THE ABOVE RESULTS ARE THE FINAL OPTIMUM .
```

3.10.2.4 USER SUPPLIED SUBROUTINES

```
      SUBROUTINE OBRES (X,Y,G,H)
C
C     TEST PROBLEM 2  ---  MAXIMIZING SYSTEMS RELIABILITY
C
      REAL  X(20), Y, G(20), H(20)
      REAL  C, COST
      REAL  R1, R2, R3, R4
      REAL  K1, K2, K3, K4
      REAL  A1, A2, A3, A4
      REAL  RMIN1, RMIN2, RMIN3, RMIN4
C
      COMMON  COST
      DATA  C /800.0/
      DATA  K1, K2, K3, K4 / 100.0, 100.0, 200.0, 150.0 /
      DATA  A1, A2, A3, A4 / 0.6, 0.6, 0.6, 0.6/
      DATA  RMIN1, RMIN2, RMIN3, RMIN4 / 0.5, 0.5, 0.5, 0.5 /
C
      R1 = X(1)
      R2 = X(2)
      R3 = X(3)
      R4 = X(4)
C
      Y =  - 1.0 +  R3*( (1.-R1)*(1.-R4) )**2
     1    +   (1.-R3) * (1. - R2*(1. - (1.-R1)*(1.-R4) ) )**2
C
C
      COST =   2*K1*R1**A1 +  2*K2*R2**A2
     1           + K3*R3**A3  +  2*K4*R4**A4
C
      G(1)  =  C - COST
      G(2)  =  1.0 - R1
      G(3)  =  1.0 - R2
      G(4)  =  1.0 - R3
      G(5)  =  1.0 - R4
C
      G(6)  =  R1 - RMIN1
      G(7)  =  R2 - RMIN2
      G(8)  =  R3 - RMIN3
      G(9)  =  R4 - RMIN4
C
      RETURN
      END
C
C
      SUBROUTINE OUTPUT (X,Y,G,H)
C
      INTEGER  ICONS, IPRINT
      REAL  X(20), Y, G(20), H(20)
      COMMON  COST
      COMMON  /INOUT/ ICONS, IPRINT
C
      WRITE (ICONS,199) COST
```

```
      WRITE (IPRINT,199) COST
  199 FORMAT (/, 6X,'COST  =',F9.2)
C
      RETURN
      END
```

3.11 REFERENCES

1. Carrol, C. W., "An Operations Research Approach to the Economic Optimization of a Kraft Pulping Process", Ph.D. Dissertion, Institute of Paper Chemistry, Appletown, Wisc., 1959.

2. Carroll, C. W., "The Created Response Surface Technique for Optimizing Nonlinear Restrained Systems", Operations Research, 9, 169-184, 1961.

3. Fiacco, A. V., and G. P. McCormick, "The Sequential Unconstrained Minimization Technique for Nonlinear Programming : A Primal-Dual Method", Management Sci., 10, 360-366, 1964.

4. Fiacco, A. V., and G. P. McCormick, "Computational Algorithm for the Sequential Unconstrained Minimizatin Technique for Nonlinear Programming", Management Sci., 10, 601-617, 1964.

5. Fiacco, A. V., and G. P. McCormick, "SUMT without parameters", Systems Research Memorandum No. 121, Technical Institute, Northwestern University, Evanston, Illinois, 1965.

6. Fiacco, A. V., and G. P. McCormick, "Extension of SUMT for Nonlinear Programming : Equality Constraints and Extrapolation", Management Sci., 12 (11) : 816-829, 1966.

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

8. Hooke, R., and T. A. Jeeves, "Direct Search Solution of Numerical and Statistical Problems", J. Assoc. Comp. Mach., 8, p. 212, 1961.

9. Hwang, C. L., L. T. Fan, and S. Kumar, "Hooke and Jeeves Pattern Search Solution to Optimal Production Planning Problems", Report No. 18, Insititute of Systems Design and Optimization, Kansas State University, 1969.

10. Paviari, D. A., and D. M. Himmelblau, "Constrained Nonlinear Optimization by Heuristic Programming", AICHE meeting in New Orleans, March, 1969.

11. Lai, K. C., "Optimization of Industrial Management Systems by the Sequential Unconstrained Minimization Technique", M.S. Report, Dept. of Industrial Engineering, Kansas State University, 1970.

12. Hwang, C. L., K. C. Lai, F. A. Tillman, and L. T. Fan, "Optimization of System Reliability by the Sequential Unconstrained Minimization Technique", IEEE Trans. on Reliability, vol. R-24., pp. 133-135., June 1975.

13. Tillman, F. A., C. L. Hwang, and W. Kuo, Optimization of Systems Reliability, Marcel Dekker, New York, 1980.

CHAPTER 4

RAC - SEQUENTIAL UNCONSTRAINED MINIMIZATION TECHNIQUE

4.1 INTRODUCTION

The general nonlinear programming problem with nonlinear (and/or linear) inequality and/or equality constraints is to choose x to

minimize $f(x)$

subject to

$$g_i(x) \geq 0 \ , \quad i=1,2,\ldots,m$$

and

$$h_j(x) = 0 \ , \quad j=1,2,\ldots,\ell$$

where x is an n-dimensional vector $(x_1,x_2,\ldots,x_n)$. A number of techniques have been developed to solve this problem. The method presented here is the sequential unconstrained minimization technique (SUMT) as implemented by Fiacco and McCormick [1,2,3,4,5]. The basic SUMT algorithm was introduced in Chapter 3.

The major differences between the RAC-SUMT and the KSU-SUMT computer program is described below.

4.2 METHOD

4.2.1 MAJOR DIFFERENCES BETWEEN RAC-SUMT AND KSU-SUMT COMPUTER PROGRAM

Although both the RAC-SUMT and KSU-SUMT computer programs use the basic SUMT algorithm, there are a few major differences in the implementation of the algorithm. The first major difference is in the formulation of the P-function. The KSU-SUMT formulation of the P-function is

$$P(x,r_k) = f(x) + r_k \sum_{i=1}^{m} 1/g_i(x) + r_k^{-1/2} \sum_{j=1}^{\ell} h_j^{\,2}(x)$$

The RAC-SUMT formulation of the P-function is [6]

$$P(x,r_k) = f(x) - r_k \sum_{i=1}^{m} \ln g_i(x) + r_k^{-1} \sum_{j=1}^{\ell} h_j^2(x)$$

Whereas the KSU-SUMT program uses $\sum_i 1/g_i(x)$ as the added barrier for inequality constraints, the RAC-SUMT program uses $-\sum_i \ln g_i(x)$. In addition, instead of using $r^{-1/2}$ as the penalty factor for the equality constraints, the term r^{-1} is used.

A second major difference between the two programs is in the method used to minimize the P-function. Whereas the KSU-SUMT program uses the Hooke and Jeeves pattern search technique to minimize the P-function, the RAC-SUMT program uses one of four methods : two versions of a second order gradient method, a first order gradient method, or a conjugate gradient method. The four methods are actually only used to determine the search direction; the Golden Section method determines the step size.

A third difference is the use of extrapolation in the RAC-SUMT program to speed up convergence to the optimum point. The extrapolation is carried out using the previous two or three suboptimum points. The new point computed by extrapolation is then used as a starting point for the next subproblem search.

The details of the unconstrained minimization techniques and the extrapolation technique are explained in [5]. In the next section, a summary of the basic logic of the method is presented.

4.2.2 SUMMARY OF COMPUTATIONAL PROCEDURE

The computational procedure for RAC-SUMT is summarized below (see Fig. 4.1).

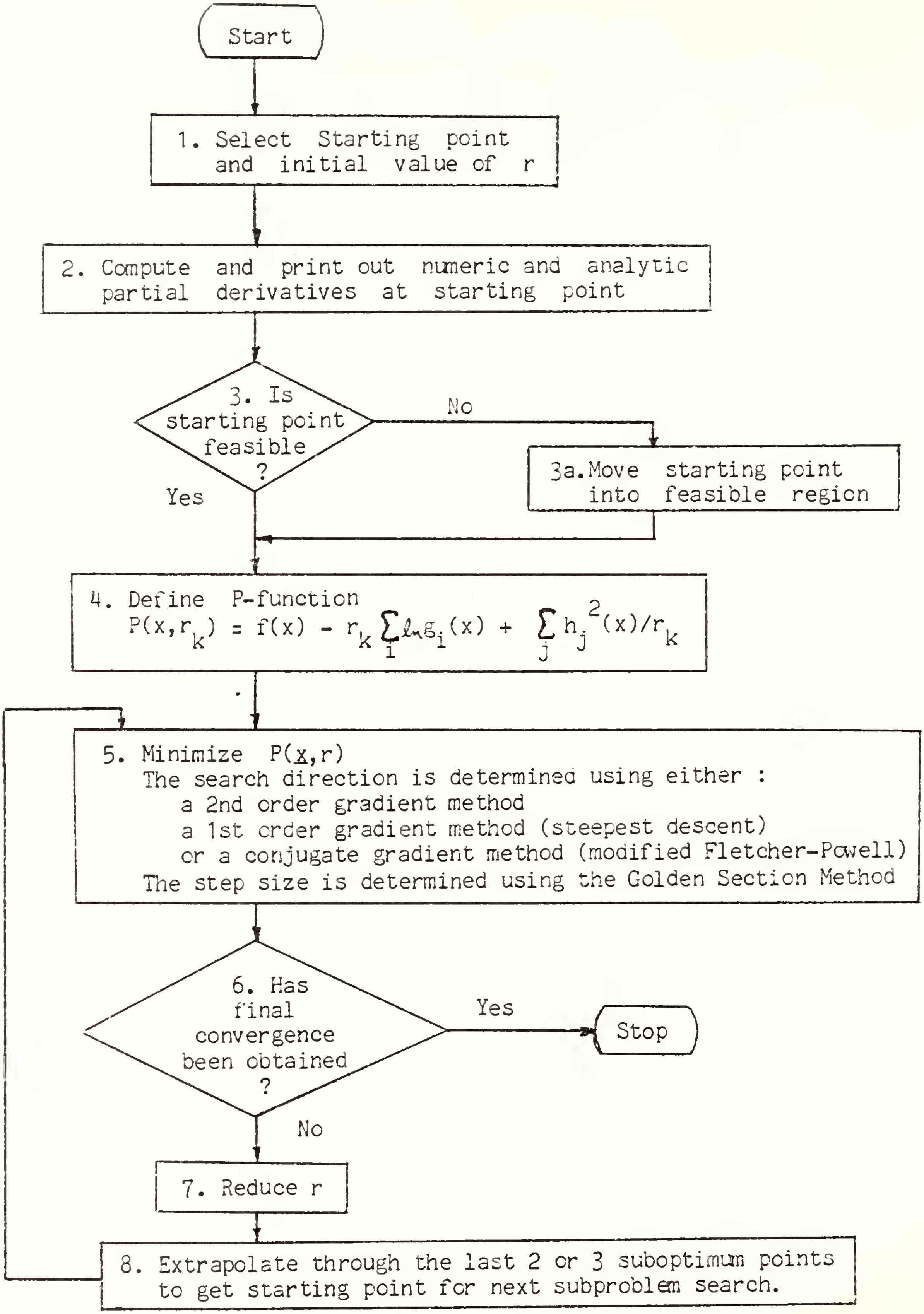

Fig. 4.1 Descriptive flow diagram for RAC-SUMT method

Step (1) Select a starting point $x^0 = (x_1, x_2, \ldots, x_n)$ and the initial value of the penalty coefficient r.

Step (2) If the user requests it, print out the values of both the numeric and analytic first and second partial derivatives at the starting point. This enables the user to check the user-supplied analytic derivatives by comparing them with the computed numeric derivatives.

Step (3) Check if the initial point is feasible subject to the inequality constraints. If it is, go to step 4; otherwise, go to step 3a.

Step (3a) Locate a feasible point by minimizing the negative of the sum of the violated inequality constraints.

Step (4) Define the P function as

$$P(x,r_k) = f(x) - r_k \sum_{i=1}^{m} \ln g_i(x) + r_k^{-1} \sum_{j=1}^{\ell} h_j^2(x)$$

where $g_i(x) \geq 0$, $i=1,2,\ldots,m$, are inequality constraints and $h_j(x) = 0$, $j=1,2,\ldots,\ell$, are equality constraints.

Step (5) Minimize the P function for the current value r_k. The direction of search is obtained by using either a second order gradient method, a first order gradient method (Steepest descent) or a conjugate gradient method (modified Fletcher-Powell); the method is chosen by the user. The step size is determined using the Golden Section method.

Step (6) Check if the final convergence has been obtained. If it has, then stop; otherwise, go to step 7. The criteria for determining convergence is one of the following :

$$\left| \frac{G - f(x)}{G} \right| < \Theta$$

$$\text{or} \quad \left| r \sum_{j=1}^{m} \ln g_j(x) \right| < \Theta$$

where G is the dual value, $G = f(x) + (2/r) \sum_{j=1}^{\ell} h^2(x) - m \cdot r - n \cdot r$

Step (7) Reduce the r value, $r_k = r_{k-1}/C$, where C is a constant greater than 1.

Step (8) Extrapolate through the last two or three suboptimum points to get the starting point for the next subproblem search. Then return to step 5.

4.3 COMPUTER PROGRAM DESCRIPTION

The RAC-SUMT computer program is actually two programs : a READIN program and a RACSUMT program. The READIN program is used to input the data and the RACSUMT program does the computations to get the solution. The reason why two separate programs are used instead of one is that both programs could not fit into the computer memory at the same time.

The microcomputer used was a North Star Horizon II which has 64K bytes of memory but only 37K bytes of it is available for the program and data; the other 27K is reserved for the operating system and other functions. The software used was Microsoft's Fortran-80 for the NorthStar microcomputer which was run under the CP/M (version 2.26) operating system.

Using Microsoft's North Star Fortran compiler, the size of the READIN program was 14K bytes while the size of the RACSUMT program depended on the size of the problem : 32K bytes was needed for test problem 1 (N=3, M=2) while 34K bytes was needed for test problem 2 (N=4, M=9). Therefore, both programs will not fit into memory at the same time. But since the READIN program is needed only to input the data, it can be removed from the computer's memory once it is through executing and the RACSUMT program can then be brought into memory. This process is done automatically with a CALL FCHAIN statement which loads the RACSUMT program into memory and begins to execute it. This statement is the last statement in the READIN program.

The only problem with the above procedure is that when the RACSUMT program is loaded into memory, the data from the READIN program is lost. In order to save the data, the READIN program must store the data on disk and the RACSUMT program must then read the data back from disk. This is what is done in the two programs.

IF the FORTRAN compiler does not have a program chaining statement (CALL FCHAIN ('filename',drive)), it is still possible to run the program.

Simply remove the statement CALL FCHAIN ('RACSUMT COM',2) from the READIN program and add a step 6 which is simply to type

B:RACSUMT

which loads and executes the RACSUMT program manually. This step is performed after the READIN program is finished executing, which occurs when a STOP and then an A> is displayed on the CRT screen.

4.3.1 DESCRIPTION OF SUBROUTINES

The READIN program consists of a main program which allows the user to interactively enter the data needed for the RACSUMT program.

The RACSUMT program consists of a main program, two control subroutines (BODY,FEAS), sixteen special purpose subroutines (CONVRG, EVALU, GRAD, INPUT, INVERS, OPT, OUTPUT, PEVALU, REJECT, RHOCOM, SECORD, STORE, XMOVE, DIFF1, DIFF2, CHCKER) and three user supplied subroutines (RESTNT, GRAD1, MATRIX). Input is coordinated by the READIN program and subroutine INPUT. Output is from the main program and subroutines BODY, CHECKER, CONVRG, FEAS, INVERS, OPT, OUTPUT. The relationship among the subroutines is shown in Fig. 4.2 and Fig. 4.3.

The description of each subroutine follows.

SUBROUTINE BODY coordinates all subroutines.

SUBROUTINE CHCKER is used to check the correctness of the user-supplied first and second partial derivatives by printing the values of both the user-supplied analytic derivatives and the computed numeric derivatives.

SUBROUTINE CONVRG (N1) checks for convergence to the subproblem.

SUBROUTINE DIFF1 (IN) computes numeric first derivatives by central difference.

* indicates user-supplied subroutines

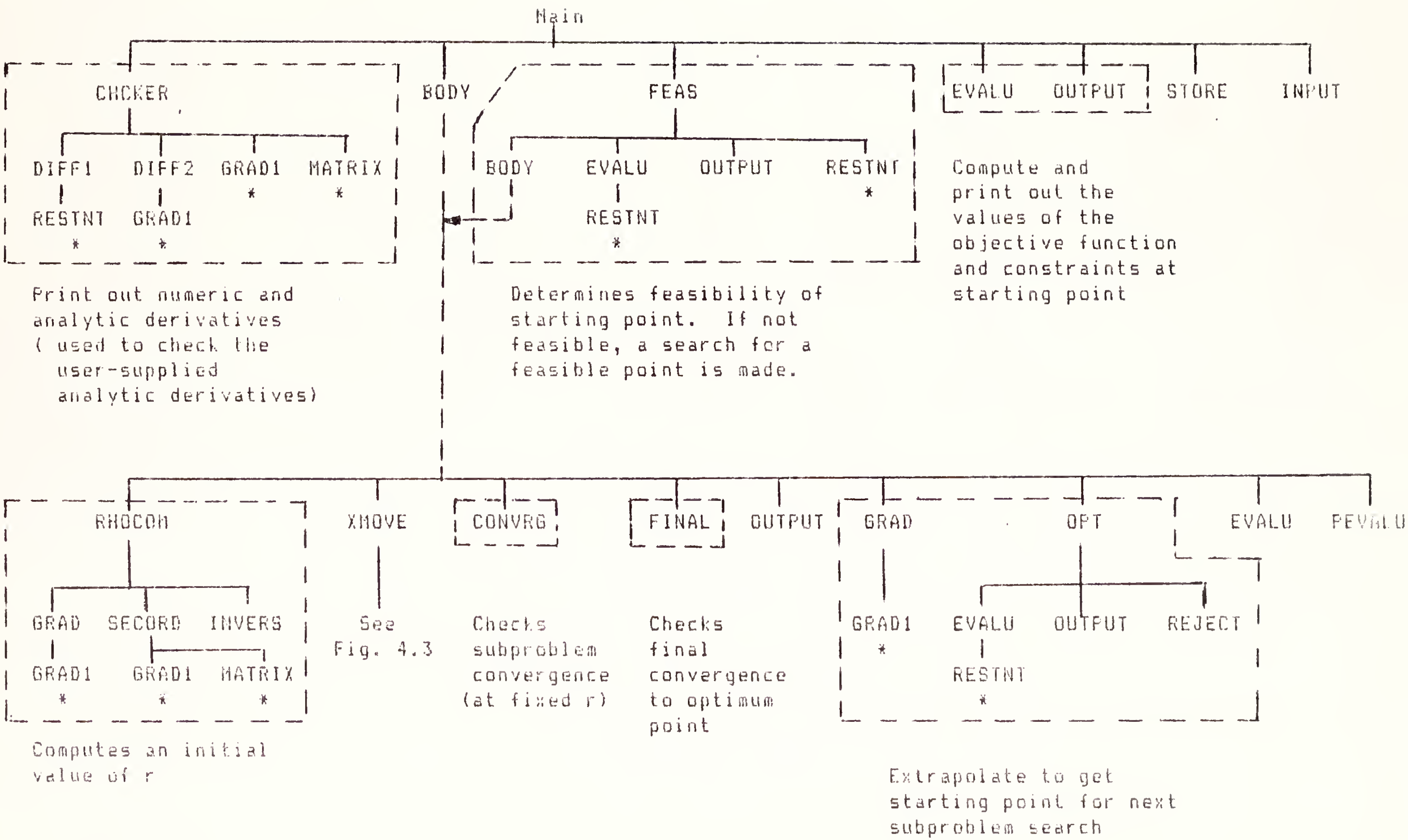

Fig. 4.2 Hierarchy of Subroutines

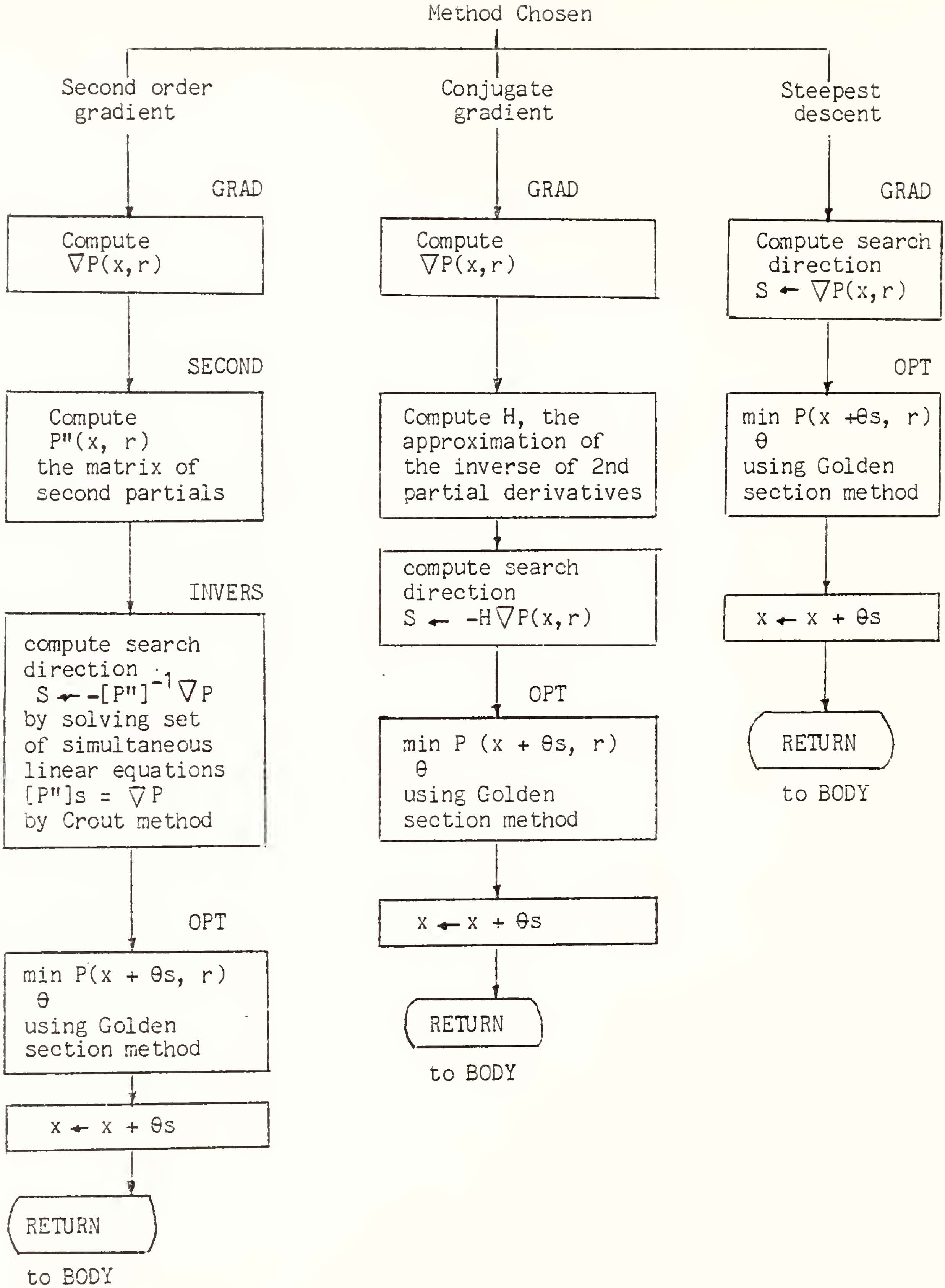

Fig. 4.3 Descriptive flow diagram for minimizing P(x, r) function in XMOVE subroutine

SUBROUTINE DIFF2 (IN) computes numeric second partial derivatives by central difference.

SUBROUTINE EVALU evaluates the P-function, the dual value G, and the constraints.

SUBROUTINE FEAS determines the feasibility of the starting point; if it is not feasible, a feasible point is sought; if no feasible point is possible, an error message is printed.

SUBROUTINE FINAL (N2) checks for final convergence to the optimum point.

SUBROUTINE GRAD (IS) computes the gradient of the P-function.

SUBROUTINE INPUT reads in the input data which was saved on disk by the READIN program.

SUBROUTINE INVERS (NSME) solves the set of equations to determine the search direction.

SUBROUTINE OPT performs a one dimensional search for the optimal step size using the Golden Section method.

SUBROUTINE OUTPUT (K) prints out the results at each suboptimum point.

SUBROUTINE PEVALU computes the P-function value and dual value using the previously computed values of f(x) and g(x).

SUBROUTINE REJECT returns stored values to their normal locations.

SUBROUTINE RHOCOM computes an initial value of r.

SUBROUTINE SECORD (IS) computes second partial derivatives of the P-function.

SUBROUTINE STORE stores the values of the current point.

SUBROUTINE XMOVE determines the search direction and then calls OPT to find the step size. The user has the option of specifying which method to use to compute the search direction (two versions of a second order gradient method, the steepest descent method, or a modified Fletcher-Powell method).

SUBROUTINE RESTNT (I,VAL) specifies the objective function and constraints (user supplied).

SUBROUTINE GRAD1 (I) specifies the first partial derivatives of the objective function and constraints (user supplied).

SUBROUTINE MATRIX (J,L) specifies the second partial derivatives of the objective function and constraints (user supplied).

4.3.2 PROGRAM LIMITATIONS

The program will presently handle a problem with 20 variables and 40 constraints (inequality + equality). To solve a larger problem, the dimensions of the arrays in the program must be changed. The key to the changes are as follows :

X, DEL, A, X1, X2, X3, DELX, DELX0,
XR1, XR2, PGRAD, DIAG, SIG, XXX, YY, DELL --- N dimensions
RJ, RJ1 --- M + MZ dimensions

The READIN program requires 14K bytes of memory and the RACSUMT program requires at least 32K bytes of memory. The smallest problems require 32K bytes; larger problems like test problem 2 (4 variables, 9 constraints) require 34K bytes; larger problems will require even more memory. Note that even though a microcomputer may have 64K bytes of memory, usually only 30-40 K bytes of it may actually be used for the program; the rest is taken up by the operating system or reserved for special purposes. Thus, the North Star Horizon microcomputer with 64K bytes of memory has only 37K bytes available for the program and will not be able to solve a problem very much larger than test problem 2.

4.3.3 LISTING OF FORTRAN PROGRAM

```
      PROGRAM RSUMT
C
C                  **  RAC SUMT  PROGRAM ---  VERSION 4  **
C          ******************************************************
C
C          THIS PROGRAM IS FOR OPTIMIZING THE GENERAL NONLINEAR
C     PROGRAMMING PROBLEM WITH NONLINEAR (AND/OR LINEAR) INEQUALITY
C     AND/OR EQUALITY CONSTRAINTS.
C
C     THE METHOD EMPLOYS :
C            SUMT FORMULATION ......... FIACCO AND MCCORMICK
C            SEARCH TECHNIQUE ........  THE USER HAS THE OPTION OF
C                 SPECIFYING WHICH OF THE FOLLOWING METHODS TO USE
C                 TO DETERMINE THE DIRECTION OF SEARCH.
C                      CONJUGATE GRADIENT METHOD
C                      FIRST ORDER GRADIENT METHOD
C                      SECOND ORDER GRADIENT METHOD
C                 THE OPTIMUM STEP SIZE IS DETERMINED USING THE
C                 GOLDEN SECTION METHOD.
C
C       THE PROGRAM IS WRITTEN BY :
C            W.C. MYLANDER , R. L. HOLMES  AND  G. P. MCCORMICK
C            RESEARCH ANALYSIS CORPORATION,  MCLEAN, VA.,  1971.
C
C    ****************************************************************
C
C
      EXTERNAL  RESTNT, GRAD1, MATRIX
C
      INTEGER  CONSOL, PRINTR
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /EQAL/ H, H1, MZ
      COMMON /OPTNS/ NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
      COMMON /VALUE/ F,G,PO,RSIGMA,RJ(20),RHO
      COMMON /CRST/ DELX(20),DELXO(20),RHOIN,RATIO,EPSI,THETAO,
     1 RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2 PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3 PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
      COMMON /EXPOPT/ NEXOP1, NEXOP2, XEP1, XEP2
      COMMON /DEVC/ CONSOL, PRINTR, NP
C
      DATA  CONSOL, PRINTR /1,2/
      DATA  XEP1, XEP2 / 0.0001, 0.0/
C
C
      CALL INPUT
C
      NTCTR = 0
      NP1 = N+1
      NM1 = N-1
C*   * CALL TIMEC
      NPHASE = 4
```

```
C         JUST TO GET AN INITIAL PRINTOUT
        CALL EVALU
        PO = 0.0
        G=0.0
        H=0.0
        RSIGMA = 0.0
        CALL OUTPUT (2)
        CALL STORE
        IF (NEXOP1.GT.1) CALL CHCKER
        IF (NEXOP1.EQ.3) STOP 01072
        IF (NEXOP1.EQ.5) STOP 01104
        CALL FEAS
C
C          NPHASE = 5  INDICATES NO FEASIBLE POINT WAS FOUND
        GO TO (30,30,30,30,40), NPHASE
   30      NPHASE = 2
           NTCTR=0
           CALL BODY
C
        WRITE (PRINTR,181)
        WRITE (PRINTR,189) F
        WRITE (PRINTR,187)
        WRITE (PRINTR,186) (I, X(I), I=1,N)
        WRITE (PRINTR,180)
C
  189   FORMAT (//,2X,19HFINAL VALUE OF F =  ,1PE15.6)
  187   FORMAT (//,2X,14HFINAL X VALUES  )
  186   FORMAT (1X, 3(2X,2HX(, I2, 3H) = ,1PE14.6) )
  181   FORMAT (//,1X,38('* ') )
  180   FORMAT ('1',' ')
C
   40   STOP
        END




        SUBROUTINE BODY
C
C       BODY COORDINATES THE FLOW AMONG THE SUBROUTINES THAT ACTUALLY DO
C       THE CALCULATIONS REQUIRED BY THE VARIOUS PARTS OF THE ALGORITHM.
C
        INTEGER  CONSOL, PRINTR
        COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
        COMMON /OPTNS/ NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
        COMMON /VALUE/ F,G,PO,RSIGMA,RJ(20),RHO
        COMMON /CRST/ DELX(20),DELXO(20),RHOIN,RATIO,EPSI,THETAO,
     1  RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2  PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3  PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
        COMMON /CONPAR/ NF1, NF2, NF3
        COMMON /DEVC/ CONSOL, PRINTR, NP
C
        NF2=2
```

```
      NF3=2
      MN=0
      NUMINI=0
C       OPTION OF GETTING INITIAL RHO
      CALL RHOCOM
      CALL EVALU
   10 CALL XMOVE
      GO TO (30,20), NT3
C
C*   * 20 CALL TIMEC
   20 CALL OUTPUT (1)
      GO TO 40
C
C*  *  30  CALL TCHECK
   30 CONTINUE
C
C          IN FEASIBILITY PHASE,  4  MEANS FEASIBILITY ACHIEVED
   40 GO TO (50,50,50,200), NSATIS
C
   50 CALL CONVRG (N1)
      GO TO (60,10,125), N1
C
C        MINIMUM ACHIEVED IF  N1 = 1
   60 GO TO (70,80), NT3
C
C*  *  70   CALL TIMEC
   70    CALL OUTPUT(1)
C
C          NUMBER OF MINIMA ACHIEVED INCREASED BY 1
   80    NUMINI = NUMINI + 1
         MN = 0
         GO TO (190,90,90), NPHASE
C
C*   90       CALL ESTIM
C
C               FINAL MIGHT HAVE BEEN CALLED BY ESTIM
C               --- CONVERGED IF N2 = 1
C*       GO TO (100,110,120), NT4
C
C             NT4=1  FINAL CONVERGENCE ON 0 ORDER ESTIMATES
C             NT4=2  CONVERGE ON FIRST ORDER ESTIMATES
C             NT4=3  CONVERGE ON SECOND ORDER ESTIMATES
   90    CALL FINAL (NF1)
         GO TO (130,140), NF1
  110    GO TO (130,140), NF2
  120    GO TO (130,140), NF3
  125 NPHASE = 5
  130 RETURN
C
  140 RHO = RHO / RATIO
C
C         USING PREVIOUSLY COMPUTED VALUES FOR F, AND RJ
C         P IS RECOMPUTED WITH THE NEW VALUE OF RHO.
      CALL PEVALU
C
```

```
CC          A VECTOR IS LEFT IN DELX(I) BY ESTIM
       IF (NUMINI-2) 10,150,150
  150  GO TO (10,160,160), NT7
  160     CALL GRAD(2)
          CALL OPT
          GO TO (180,170), NT3
  170        WRITE (PRINTR,210)
  210        FORMAT (//,2X,30HMOVED ON EXTRAPOLATION VECTOR    )
             CALL OUTPUT (1)
  180        GO TO 50
C
C         DUAL VALUE GREATER THAN 0  MEANS NO FEASIBLE POINT EXISTS
  190  IF (G) 90,90,200
C
C
  200  RETURN
       END



       SUBROUTINE CHCKER
C
C      CHCKER COMPUTES AND LIST THE FIRST PARTIAL DERIVATIVES USING GRAD1
C      AND THEN USING NUMERICAL DIFFERENCING (DIFF1).  IF REQUESTED, THE
C      SECOND PARTIAL DERIVATIVES ARE COMPUTED AND LISTED USING MATRIX
C      AND DIFF2.
C
       INTEGER  CONSOL, PRINTR
       COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
       COMMON /EQAL/ H, H1, MZ
       COMMON /EXPOPT/ NEXOP1, NEXOP2, XEP1, XEP2
       COMMON /DEVC/ CONSOL, PRINTR, NP
C
       MMZ = 1 + M + MZ
       DO 5 J=1,N
         DEL(J) = 1.2345678
    5  CONTINUE
C
       DO 10 I=1,MMZ
          IN = I-1
          IF (IN) 170,170,180
  170        WRITE (PRINTR,1)
             GO TO 190
C
  180        WRITE (PRINTR,2) IN
  190     CALL GRAD1 (IN)
          WRITE (PRINTR,3)
          WRITE (PRINTR,4) (J, DEL(J), J=1,N )
          CALL DIFF1 (IN)
          WRITE (PRINTR,6)
          WRITE (PRINTR,4) (J, DEL(J), J=1,N)
   10  CONTINUE
C
C      SOMETIMES FIRST DERIVATIVES ARE  TO BE CHECKED
       IF (NEXOP1.LT.4) GO TO 160
```

```
C
       DO 150 I=1,MMZ
          IN = I-1
          IF (IN) 200,200,210
 200         WRITE (PRINTR,1)
             GO TO 220
C
 210         WRITE (PRINTR,2) IN
 220      IT = 2
          DO 30 K=1,N
          DO 30 J=1,N
            A(K,J) = 0.0
  30      CONTINUE
C
          CALL MATRIX (IN,IT)
          IF (IT.EQ.1) GO TO 150
          DO 50 K=2,N
             KM1 = K-1
             DO 40 J=1,KM1
                IF ( A(K,J).EQ.0.0 ) GO TO 40
                NEXOP1 = 5
                WRITE (PRINTR,7) K,J
                GO TO 60
  40         CONTINUE
  50      CONTINUE
C
  60      WRITE (PRINTR,9)
          DO 90 K=1,N
             DO 70 J=K,N
               IF ( A(K,J).NE.0.0 )  GO TO 80
  70         CONTINUE
  80         WRITE (PRINTR,8) (K, J, A(K,J), J=1,N)
  90      CONTINUE
C
          DO 110 K=1,N
          DO 110 J=1,N
             A(K,J) = 0.0
 110      CONTINUE
C
          WRITE (PRINTR,11)
          CALL DIFF2 (IN)
          DO 140 K=1,N
             DO 120 J=K,N
                IF ( A(K,J).NE.0.0 )  GO TO 130
 120         CONTINUE
             GO TO 140
 130         WRITE (PRINTR,8) (K, J, A(K,J), J=1,N)
 140      CONTINUE
 150   CONTINUE
C
 160   CONTINUE
C
   1   FORMAT (//, 2X, 38HVALUES OF OBJECTIVE FUNCTION PARTIALS  )
   2   FORMAT (/, 2X, 29HVALUES OF CONSTRAINT NUMBER  ,I2 )
   3   FORMAT (/, 2X, 25HANALYTICAL FIRST PARTIALS   )
```

```
    4   FORMAT (1X, 3(2X,4HDEL(, I2, 3H) = ,E14.7) )
    6   FORMAT (/, 2X, 24HNUMERICAL FIRST PARTIALS   )
    7   FORMAT (/, 2X, 2HA(, I2,1H, ,I2, 10H) .NE. 0.0 )
    8   FORMAT (1X, 3(2X, 2HA(, I2,1H,,I2,4H) = ,E12.6) )
    9   FORMAT (/, 2X, 26HANALYTICAL SECOND PARTIALS  )
   11   FORMAT (/, 2X, 25HNUMERICAL SECOND PARTIALS   )
C
        RETURN
        END


        SUBROUTINE CONVRG (N1)
C
C       AFTER EACH ITERATION OF THE ALGORITHM TO LOCATE THE MINIMUM OF THE
C       PENALTY FUNCTION, CONVRG DETERMINES IF THE CURRENT POINT IS CLOSE
C       ENOUGH TO THE POINT GIVING THE MINIMUM VALUE OF THE P FUNCTION.
C       N1 SET EQUAL TO 1 IF MINIMUM HAS BEEN FOUND.
C       N1 SET EQUAL TO 2 IF MINIMUM HAS NOT BEEN FOUND (AND TIME IS NOT UP).
C       N1 SET EQUAL TO 3 OTHERWISE.
C
        INTEGER  CONSOL, PRINTR
        COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
        COMMON /OPTNS/ NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
        COMMON /VALUE/ F,G,PO,RSIGMA,RJ(20),RHO
        COMMON /CRST/ DELX(20),DELXO(20),RHOIN,RATIO,EPSI,THETAO,
     1  RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2  PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3  PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
        COMMON /EXPOPT/ NEXOP1, NEXOP2, XEP1, XEP2
        COMMON /TSW/ NSWW
        COMMON /DEVC/  CONSOL, PRINTR, NP
C
        N1=2
        IF (NT8.LE.1) Q1=PO
        NT8=2
        IF (MN.LE.1) Q1=PO
C
        GO TO (10,20,30), NT9
   10   IF ( ABS(DOTT).LT.EPSI ) GO TO 70
        GO TO 40
C
   20   IF ( ABS(DOTT).LT.(P1-PO)/5.0 )  GO TO 70
        GO TO 40
C
   30   IF (ADELX.LT.EPSI) GO TO 70
C
   40   GO TO (50,60), NSWW
   50   IF (MN.LE.1) RETURN
C
        IF (PO+XEP2 .LT. Q1) GO TO 75
        GO TO 70
C
   60   WRITE (PRINTR,90)
        N1=3
```

```
C
C          FOUND THE MINIMUM TO THE SUBPRCBLEM
      RETURN
C
   70 N1=1
   75 Q1 = P0
C
   90 FORMAT (///, 10X, 37H**** TIME LIMIT.   CALLING EXIT FROM  ,
     1         13HCONVRG ***** )
C
      RETURN .
      END



      SUBROUTINE DIFF2 (IN)
C
C     DIFF2 COMPUTES THE SECOND DERIVATIVES BY NUMERICAL DIFFERENCING
C
      COMMON /SHARE/ X(20),DEL(20),A(20,20), N,M,MN,NP1,NM1
      COMMON /EXPOPT/ NEXOP1, NEXOP2, XEP1, XEP2
      COMMON /STIRX/ XSTR(20), XSSS(20), DDLL(20)
C
      DO 10 J=1,N
         XSSS(J) = X(J)
   10 CONTINUE
C
      DO 50 J=1,N
        IF (J.EQ.1) GO TO 20
           JM1 = J-1
           X(JM1) = XSSS(JM1)
C
   20      X(J) = XSSS(J) + XEP1
           CALL GRAD1 (IN)
           DO 30 I =1,N
             DDLL(I) = DEL(I)
   30      CONTINUE
           X(J) = XSSS(J) - XEP1
           CALL GRAD1 (IN)
           DO 40 I=J,N
             A(J,I) = (DDLL(I)-DEL(I) ) / (2.0*XEP1)
   40      CONTINUE
   50 CONTINUE
C
      X(N) = XSSS(N)
C
      RETURN
      END



      SUBROUTINE DIFF1 (IN)
C
C     DIFF1 COMPUTES THE FIRST DERIVATIVES BY NUMERICAL DIFFERENCING.
C     USER CAN CALL FOR DIFFERENCING OF SELECTED FUNCTIONS.
```

```
C
      COMMON /SHARE/ X(20),DEL(20),A(20,20), N,M,MN,NP1,NM1
      COMMON /EXPOPT/ NEXOP1, NEXOP2, XEP1, XEP2
      COMMON /STIRX/ XSTR(20), XSSS(20), DDLL(20)
C
      DO 10 J=1,N
        XSTR(J) = X(J)
   10 CONTINUE
C
      DO 30 J=1,N
         IF (J.EQ.1) GO TO 20
           JM1=J-1
           X(JM1) = XSTR(JM1)
C
   20      X(J) = XSTR(J) + XEP1
           CALL RESTNT (IN,ZZ2)
           X(J) = XSTR(J) - XEP1
           CALL RESTNT (IN,ZZ1)
           DEL(J) = (ZZ2-ZZ1) / (2.0 * XEP1)
   30 CONTINUE
C
      X(N) = XSTR(N)
C
      RETURN
      END



      SUBROUTINE EVALU
C
C     IN THE NORMAL PHASE EVALU CALLS THE USER-SUPPLIED ROUTINES TO
C     EVALUATE THE OBJECTIVE FUNCTION AND THE CONTRAINT FUNCTIONS
C     AT THE CURRENT POINT.  IN THE FEASIBILITY PHASE  THIS ROUTINE
C     PUTS THE NEGATIVE SUM OF THE VIOLATED CONSTRAINTS IN LOCATION F.
C
      INTEGER  CONSOL, PRINTR
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /EQAL/  H, H1, MZ
      COMMON /OPTNS/ NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
      COMMON /VALUE/ F,G,P0,RSIGMA,RJ(20),RHO
      COMMON /CRST/ DELX(20),DELX0(20),RHOIN,RATIO,EPSI,THETA0,
     1 RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2 PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3 PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
C
      H = 0.0
      RSIGMA = 0.0
      F = 0.0
      NSATIS = 2
C
C     NPHASE DETERMINES THE PHASE OF PROGRAM
C       1  PROBLEM IN FEASIBILITY PHASE
C       2  PROBLEM IN REGULAR PHASE
C       3  PROBLEM IN GUESS PHASE
C       4  EVALUATE ALL FUNCTIONS REGARDLESS OF PHASE
```

```
C
      GO TO (10,100,190,200), NPHASE
C
C     ** FEASIBILITY PHASE
   10 GO TO (20,40), NT2
C
C     NON-NEGATIVES INCLUDED
   20 DO 30 I=1,N
         IF ( X(I).LE.0.0 ) GO TO 260
         RSIGMA = RSIGMA - RHO * ALOG ( X(I) )
   30 CONTINUE
C
   40 IF (M.EQ.0)  GO TO 90
C
      DO 80 J=1,M
         CALL RESTNT ( J, RJ(J) )
         IF ( RJ1(J).LE.0.0 ) GO TO 50
         IF ( RJ(J).GT.0.0 ) GO TO 60
C        VIOLATION OF A PREVIOUSLY SATISFIED CONSTRAINT
         GO TO 260
C
   50    IF ( RJ(J).GT.0.0 )  GO TO 70
C     ALL VIOLATED CONSTRAINTS ADDED INTO OBJECTIVE FUNCTION
         F = F - RJ(J)
         GO TO 80
C
   60    RSIGMA = RSIGMA  -  RHO * ALOG ( RJ(J) )
         GO TO 80
C
C     INDICATES SATISFACTION OF CONSTRAINT  ( 1 OR MORE )
   70    NSATIS = 1
         RSIGMA = RSIGMA  -  RHO * ALOG( RJ(J) )
C
   80 CONTINUE
C
   90 CONTINUE
C     EQUALITIES NOT COMPUTED IN FEASIBILITY PHASE
      PO = F + RSIGMA
      G = F - RHO * FLOAT(M)
      IF (NT2.EQ.1) G = G - RHO * FLOAT(N)
      RETURN
C
C     REGULAR PHASE
  100 GO TO (110,130), NT2
C
C     NON NEGATIVITIES INCLUDED
  110    DO 120 I=1,N
            IF ( X(I).LE.0.0 ) GO TO 260
               RSIGMA = RSIGMA - RHO * ALOG( X(I) )
  120    CONTINUE
C
  130    IF (M.EQ.0) GO TO 150
         DO 140 J=1,M
            CALL RESTNT ( J, RJ(J) )
            IF ( RJ(J).LE.0.0) GO TO 260
```

```
          RSIGMA = RSIGMA - RHO * ALOG( RJ(J) )
  140   CONTINUE
C
C       EVALUATE AND ADD IN EQUALITY CONSTRAINTS
  150   CONTINUE
        CALL RESTNT (0,F)
        IF (MZ) 180,180,160
  160      DO 170 I=1,MZ
              J=I+M
              CALL RESTNT ( J, RJ(J) )
C             ADD INTO THIRD TERM OF P FUNCTION
              H = H + ( RJ(J) )**2
  170      CONTINUE
           H = H / RHO
C
  180   PO = RSIGMA + H
        PO = F + PO
        G = 2.0 * H - RHO * FLOAT(M)
        G = G + F
        IF ( NT2.EQ.1)  G = G - RHO * FLOAT(N)
C        DUAL VALUE
        RETURN
C
C       GUESS PHASE NOT YET CODED
  190   RETURN
C
C       STRAIGHT FUNCTION EVALUATION ( MAIN + FEASIBLE ONLY )
  200   CONTINUE
        IF (M.EQ.0) GO TO 220
        DO 210 I=1,M
           CALL RESTNT ( I, RJ(I) )
  210   CONTINUE
C
  220   CALL RESTNT ( 0,F )
C       EQUALITY CONSTRAINTS
        IF (MZ) 250,250,230
  230      DO 240 I=1,MZ
              KZ = M + I
              CALL RESTNT ( KZ, RJ(KZ) )
  240      CONTINUE
C
  250   RETURN
C
C       CONSTRAINTS VIOLATED NOT SO BEFORE
  260   NSATIS = 3
        PO = 10.0E35
C
        RETURN
        END
```

```
      SUBROUTINE FEAS
C
C     FEAS DETERMINES WHETHER THE STARTING POINT IS FEASIBLE.
C     IF IT IS NOT, FEAS LOOKS FOR A FEASIBLE ONE.
C     IF NONE EXISTS, A MESSAGE IS PRINTED AND CONTROL RETURNS
C     TO MAIN.
C
      INTEGER  CONSOL, PRINTR
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /OPTNS/ NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
      COMMON /VALUE/ F,G,PO,RSIGMA,RJ(20),RHO
      COMMON /CRST/ DELX(20),DELXO(20),RHOIN,RATIO,EPSI,THETAO,
     1 RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2 PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3 PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
      COMMON /DEVC/ CONSOL, PRINTR, NP
C
      NPHASE = 1
      GO TO (10,50), NT2
C
   10    NFIX =1
         DO 30 I=1,N
            IF ( X(I) ) 20,20,30
   20          NFIX = 2
               X(I) = 1.0E-05
   30    CONTINUE
C
         GO TO (50,40), NFIX
C
   40       NPHASE = 4
            CALL EVALU
            NPHASE = 1
            WRITE (PRINTR,130)
  130 FORMAT (//, 2X, 43HMADE VARIABLES WHICH VIOLATED NON-NEGATIVE   ,
     1          30HCONSTRAINTS SLIGHTLY POSITIVE   )
            CALL OUTPUT (2)
C
   50       IF (M) 90,90,60
C
   60          DO 70 I=1,M
                  IF ( RJ(I) ) 100,100,70
   70          CONTINUE
               IF (NPHASE.EQ.1) GO TO 90
C
C*    *  80      CALL TIMEC
   80          WRITE (PRINTR,140)
  140       FORMAT (//,2X,38HTHE FEASIBLE STARTING POINT AND VALUES  )
               G = 0.0
               CALL RESTNT(0,F)
               CALL OUTPUT (2)
C
   90       RETURN
C
  100 CALL BODY
      IF (NPHASE.EQ.5) RETURN
```

```
C
      DO 110 I=1,M
         IF ( RJ(I) ) 120,120,110
  110 CONTINUE
      GO TO 80
C
  120 WRITE (PRINTR,150)
  150 FORMAT (/////,2X,43HTHIS PROBLEM POSSESSES NO FEASIBLE STARTING,
     1  7H POINT.  /  2X, 36HWILL LOOK FOR DATA TO NEXT PROBLEM.  )
C
C     TO INDICATE TO MAIN TO START ON NEXT PROBLEM
      NPHASE = 5
      GO TO 90
C
      END



      SUBROUTINE FINAL (N2)
C
C     FINAL CONTAINS THE TESTS USED TO DETERMINE WHETHER A POINT
C     SATISFIES THE FINAL CONVERGENCE CRITERION CHOSEN TO DETERMINE
C     IF THE NLP PROBLEM HAS BEEN SOLVED.
C     N2 SET EQUAL TO 1  IF CONERGENCE CRITERION IS SATISFIED.
C     N2 SET EQUAL TO 2 OTHERWISE.
C
      INTEGER  CONSOL, PRINTR
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /OPTNS/ NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
      COMMON /VALUE/ F,G,PO,RSIGMA,RJ(20),RHO
      COMMON /CRST/ DELX(20),DELXO(20),RHOIN,RATIO,EPSI,THETAO,
     1 RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2 PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3 PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
      COMMON /DEVC/ CONSOL, PRINTR, NP
C
      GO TO (10,20,30), NT5
C
   10    EPSIL = ABS( F/G-1.0 )
         IF (EPSIL-THETAO) 50,50,70
C
   20    IF ( ABS(RSIGMA) - THETAO ) 50,50,70
C
   30    IF (NUMINI-1) 50,40,40
   40       PEST = PR1 - (PR1-PO) / ( 1.0 - 1.0 / SQRT(RATIO) )
            EPSIL = ABS (PEST/G-1.0)
            IF (EPSIL-THETAO) 50,70,70
C
  50     N2 = 1
         GO TO 80
C
   70    N2=2
   80 RETURN
      END
```

```
      SUBROUTINE GRAD (IS)
C
C     GRAD COMPUTES THE GRADIENT OF THE PENALTY FUNCTION AND THE
C     OUTER PRODUCT FACTORS OF THE MATRIX OF SECOND PARTIALS  OF P.
C     IF (IS=1) ACCUM. MATRIX OF 2ND PARTIALS
C     IF (IS=2) DON'T
C
      INTEGER  CONSOL, PRINTR
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /EQAL/ H, H1, MZ
      COMMON /OPTNS/ NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
      COMMON /VALUE/ F,G,PO,RSIGMA,RJ(20),RHO
      COMMON /CRST/ DELX(20),DELXO(20),RHOIN,RATIO,EPSI,THETAO,
     1 RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2 PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3 PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
      COMMON /DEVC/ CONSOL, PRINTR, NP
C
      GO TO (10,30), IS
C
   10 DO 20 I=1,N
      DO 20 J=1,I
         A(I,J) = 0.0
   20 CONTINUE
C
   30 DO 40 I=1,N
         DELXO(I) = 0.0
   40 CONTINUE
C
C     THIS SECTION WORKS CORRECTLY IN FEASIBILITY PHASE AS WELL AS
C     NORMAL PHASE
C
      GO TO (50,80), NT2
C
   50 DO 70 I=1,N
         DELXO(I) = - RHO / X(I)
         GO TO (60,70), IS
   60    A(I,J) = ( -DELXO(I) / X(I) )
   70 CONTINUE
C
   80 CONTINUE
      IF (M.LE.0) GO TO 180
      DO 170 K=1,M
         CALL GRAD1(K)
         IF ( RJ(K).GT.0.0 ) GO TO 110
C
C        ALL VIOLATED CONSTRAINT GRADS ADDED TO OBJECTIVE FUNCTION
         DO 100 I=1,N
            IF (DEL(I) ) 90,100,90
   90       DELXO(I) = DELXO(I) - DEL(I)
  100    CONTINUE
         GO TO 170
C
  110    TT = RHO / RJ(K)
         DO 160 I=1,N
```

```
            IF ( DEL(I) ) 120,160,120
C             IF DEL(I) = 3  SKIP ALL THE FOLLOWING COMPUTATION
C                                  INVOLVING  *  BY DEL(I)
  120       T = TT * DEL(I)
            DELXO(I) = DELXO(I) - T
            GO TO (130,160), IS
  130       T = T / RJ(K)
            DO 150 JJ=1,I
               IF (DEL(JJ) ) 140,150,140
  140          A(I,JJ) = A(I,JJ)  +  T * DEL(JJ)
  150       CONTINUE
  160    CONTINUE
  170 CONTINUE
C
C     EQUALITY CHANGES FOR GRAD
  180 IF (MZ.LE.0) GO TO 250
      GO TO (250,190,250), NPHASE
C
  190 RQ = 2.0 / RHO
      DO 240 J=1,MZ
         K = M + J
         CALL GRAD1(K)
         TT = RQ * RJ(K)
         DO 230 I=1,N
            IF (DEL(I).EQ.0.0 ) GO TO 230
            DELXO(I) = DELXO(I) + DEL(I) * TT
            GO TO (200,230), IS
  200       T = RQ * DEL(I)
            DO 220 JJ=1,I
               IF ( DEL(JJ) ) 210, 220, 210
  210          A(I,JJ) = A(I,JJ)  +  T * DEL(JJ)
  220       CONTINUE
  230    CONTINUE
  240 CONTINUE
C
  250 GO TO (260,280), IS
C
  260 DO 270 I=1,N
         DIAG(I) = A(I,I)
  270 CONTINUE
C
  280 GO TO (290,330,290), NPHASE
C     LEAVES NEGATIVE GRADIENT IN DELP
  290 DO 300 I=1,N
         DELXO(I) =  - DELXO(I)
  300 CONTINUE
C
  310 ADELX = 0.0
      DO 320 I=1,N
         ADELX = ADELX + DELXO(I)**2
  320 CONTINUE
C
      ADELX = SQRT(ADELX)
      RETURN
C
```

```
  330   CALL GRAD1(0)
        DO 340 I=1,N
           DELXO(I) =  - DELXO(I) - DEL(I)
  340   CONTINUE
C
C       LEAVES THE NEGATIVE GRADIENT OF P IN DELXO
        GO TO 310
C
        END


        SUBROUTINE INPUT
C
        INTEGER  CONSOL, PRINTR
        COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
        COMMON /EQAL/ H, H1, MZ
        COMMON /OPTNS/ NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
        COMMON /VALUE/ F,G,PO,RSIGMA,RJ(20),RHO
        COMMON /CRST/ DELX(20),DELXO(20),RHOIN,RATIO,EPSI,THETAO,
     1   RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2   PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3   PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
        COMMON /EXPOPT/ NEXOP1, NEXOP2, XEP1, XEP2
        COMMON /DEVC/ CONSOL, PRINTR, NP
C
        CALL OPEN (6,'OPTIONS DAT',2)
           READ (6) N,M,MZ
           READ (6) ( X(I), I=1,N )
           READ (6) RHOIN, RATIO, EPSI, THETAO
           READ (6) NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
           READ (6) NEXOP1, NEXOP2
        ENDFILE 6
C
        RETURN
        END
```

```
      SUBROUTINE INVERS (NSME)
C
C     INVERS  SOLVES THE SET OF EQUATION FOR THE MOVE-VECTOR USING
C     THE CROUT PROCEDURE.  IF THE MATRIX IS NOT POSITIVE DEFINITE,
C     A DIFFERENT METHOD IS USED.
C     PERFORMING A L-U DECOMPOSITION OF THE MATRIX A, TAKING ADVANTAGE
C     OF THE SYMMETRY OF THE A MATRIX.
C     IF A NON-POSITIVE PIVOT CANDIDATE IS GENERATED, THEN MCCORMICK'S
C     PROCEDURE IS USED ( SEE PP. 167-168 IN FIACCO AND MCCORMICK ).
C     IF NSME =1 WORKING WITH A NEW A MATRIX
C     IF NSME =2 USING PREVIOUS A MATRIX, BUT HAVE A NEW RIGHT-HAND SIDE.
C     NINV IS THE NUMBER OF NON-POSITIVE PIVOT CANDIDATES GENERATED.
C
      INTEGER  CONSOL, PRINTR
      DIMENSION B(20)
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /OPTNS/ NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
      COMMON /CRST/ DELX(20),DELXO(20),RHOIN,RATIO,EPSI,THETAO,
     1 RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2 PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3 PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
      COMMON /EXPOPT/ NEXOP1, NEXOP2, XEP1, XEP2
      COMMON /DEVC/ CONSOL, PRINTR, NP
C
      GO TO (20,170), NSME
C
   20 NINV=0
      IF ( A(1,1) ) 40,30,50
   30    NINV=1
         GO TO 70
C
   40 NINV=1
   50 A(1,1) = 1.0 / A(1,1)
      DO 60 I=2,N
         A(1,I) = A(1,I) * A(1,1)
   60 CONTINUE
C
   70 DO 160 J=2,N
         JM1=J-1
         T=0.0
         DO 90 I=1,JM1
            IF ( A(I,J)) 80,90,80
   80          T = T + A(J,I) * A(I,J)
   90    CONTINUE
C
         A(J,J) = A(J,J) - T
         IF ( A(J,J) ) 110,100,120
  100       NINV = NINV + 1
            GO TO 170
C
  110    NINV = NINV + 1
  120    A(J,J) = 1.0 / A(J,J)
         IF (J.EQ.N) GO TO 170
         JP1 = J+1
         DO 150 L=JP1,N
```

```
            T=0.0
            DO 140 I=1,JM1
               IF ( A(I,J) ) 130,140,130
  130             T = T + A(L,I) * A(I,J)
  140       CONTINUE
            A(L,J) = A(L,J) - T
            A(J,L) = A(L,J) * A(J,J)
  150    CONTINUE
  160 CONTINUE
C
  170 CONTINUE
C
      IF (NINV) 180,180,290
C
  180    B(1) = B(1) * A(1,1)
         DO 210 J=2,N
            T = 0.0
            JM1=J-1
            DO 200 I=1,JM1
               IF ( A(J,I) ) 190,200,190
  190             T = T + A(J,I) * B(I)
  200       CONTINUE
            B(J) = ( B(J)-T ) * A(J,J)
  210    CONTINUE
         DO 240 I=1,NM1
            NMK=N-I
            DO 230 J=1,I
               L = NP1 - J
               IF ( A(NMK,L) ) 220,230,220
  220             B(NMK) = B(NMK) - A(NMK,L) * B(L)
  230       CONTINUE
  240    CONTINUE
C
  250    GO TO (280,260), NT3
  260       WRITE (PRINTR,430)
  430       FORMAT (/,2X, 12HDEL P VECTOR   )
            WRITE (PRINTR,420) ( I, DELXO(I), I=1,N )
  420       FORMAT (/, 3(2X,4HDEL(, I2, 3H) = , E15.8) )
  270       WRITE (PRINTR,440)
  440       FORMAT (/, 2X, 24HSECOND ORDER MOVE VECTOR    )
            WRITE (PRINTR,420) ( I, DELX(I), I=1,N )
  280       RETURN
C
C     COMPUTE ORTHOGONAL MOVE
  290 CONTINUE
      DO 350 II=1,N
         I = N - II + 1
         IF ( A(I,I) ) 310,300,320
  300       B(I) = 0.0
            GO TO 350
C
  310       B(I) =  1.0
            GO TO 330
C
  320       B(I) = 0.0
```

```
  330         IP1 = I+1
              IF ( IP1.GT.N ) GO TO 350
              DO 340 J=IP1,N
                 B(I) = B(I) - A(I,J) * B(J)
  340         CONTINUE
  350      CONTINUE
           GO TO 360
C
C       CHECK MAYBE DO DIFF FOR P.S.D.
  360   ZC2 = 0.0
        DO 370 I=1,N
           ZC2 = ZC2 + DELXO(I) * B(I)
  370   CONTINUE
C
        IF (ZC2) 380,400,400
  380      DO 390 I=1,N
             B(I) = - B(I)
  390      CONTINUE
C
  400   IF (NEXOP2.NE.2) GO TO 250
C
           DO 410 K=1,N
              B(K) = B(K) + DELXO(K)
  410      CONTINUE
           GO TO 250
C
        END



        SUBROUTINE OPT
C
C       OPT LOOKS FOR A MINIMUM ALONG THE SEARCH VECTOR USING THE
C       GOLDEN SECTION SEARCH METHOD.
C
        INTEGER  CONSOL, PRINTR
        COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
        COMMON /VALUE/ F,G,PO,RSIGMA,RJ(20),RHO
        COMMON /CRST/ DELX(20),DELXO(20),RHOIN,RATIO,EPSI,THETAO,
     1  RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2  PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3  PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
        COMMON /DEVC/ CONSCL, PRINTR, NP
C
        KSW=1
        N405=1
        P31=PO
        ISW=1
        DOTT=0.0
        DO  10 J=1,N
           DOTT = DOTT + DELX(J) * DELXO(J)
   10   CONTINUE
        GO TO 40
C
   20   DO 30 I=1,N
```

```
         DELX(I) =  - DELX(I)
   30  CONTINUE
C
   40  CONTINUE
       N404 = 0
       MN=MN+1
C        MN IS NOW NUMBER OF POINTS AFTER MINIMUM ACHIEVED
       NTCTR = NTCTR + 1
       DO 50 I=1,N
          X2(I) = X(I)
   50  CONTINUE
C
       PX1=PO
       N401=0
   60  N401 = N401 + 1
       DO 70 I=1,N
          X(I) = X2(I) + DELX(I)
   70  CONTINUE
C
       CALL EVALU
C
C        1  MEANS SATISFIED A CONSTRAINT NOT PREVIOUSLY SATISFIED.
C        2  MEANS NO CHANGE
C        3  MEANS VIOLATION
C      IF POINT IS NOT FEASIBLE GIVE IT AN ARBITRARILY HIGH VALUE.
C
       GO TO (540,90,80), NSATIS
   80  PX2 = 10.0E35
       PO = 10.0E35
       GO TO 100
C
   90  CONTINUE
       PX2 = PO
       IF (PX1-PX2) 100,100,150
  100  IF (N401-2) 130,110,110
  110  DO 120 I=1,N
          X1(I) = X(I)
  120  CONTINUE
C
       P1 = PX2
       GO TO 430
C
C      ONLY ONE POINT SO FAR COMPUTED
  130  DO 140 I=1,N
          X3(I) = X2(I)
  140  CONTINUE
C
       PREV3=PX1
       GO TO 180
C
  150  DO 160 I=1,N
          X3(I) = X2(I)
          X2(I) = X(I)
          DELX(I) = 1.61803399 * DELX(I)
  160  CONTINUE
```

```
C
        PREV3 = PX1
        PX1 = PX2
        GO TO 60
C
C       THE GOLDEN SECTION SEARCH METHOD.
C
C       B VECTOR GOES TO X1(I)
  170   P0=1.0E36
        N404 = N404 + 1
  180   DO 190 I=1,N
           X1(I) = X(I)
  190   CONTINUE
C
        P1 = P0
        DO 200 I=1,N
           X(I) = 0.38196601 * ( X1(I)-X3(I) ) + X3(I)
           X2(I) = X(I)
  200   CONTINUE
C
        CALL EVALU
C
        GO TO (540,270,210), NSATIS
C
  210   IF (N404.LT.30) GO TO 170
C
C       IT IS POSSIBLE NO FEASIBLE POINT EXISTS, IF NOT, TRY MOVING ON
C       DELXO.  IF IT IS NOT POSSIBLE TO MOE ON DELXO  THEN WE MUST BE
C       AT A SOLUTION OF THE NLP PROBLEM.
C
        IF (N404.GT.100) GO TO 240
  220   DO 230 I=1,N
           IF ( ABS( ABS(X3(I)/X1(I) ) - 1.0 ) .GT. 1.0E-07 ) GO TO 170
  230   CONTINUE
C
  240   GO TO (250,260), N405
  250   N405=2
C
C       TRY TO MOVE ON GRADIENT
        NTCTR = NTCTR - 1
        MN = MN - 1
        GO TO 20
C
  260   WRITE (PRINTR,580)
  580   FORMAT (//, 2X, 42HOPT CAN'T FIND A FEASIBLE POINT THAT GIVES
     1          ,33H A LOWER VALUE OF THE P-FUNCTION  )
C*    * CALL TIMEC
        CALL OUTPUT (1)
        CALL REJECT
        STOP 22042
C
  270   CONTINUE
        N404 = 0
        PX1 = P0
        DO 280 I=1,N
```

```
         X(I) = 0.38196601 * ( X1(I)-X2(I) ) + X2(I)
  280   CONTINUE
C
        CALL EVALU
        GO TO (540,290,220), NSATIS
C
  290   PX2 = PO
        N401 = 1
  300   N401 = N401 + 1
        IF ( N401-25) 340,310,310
  310   KSW=2
C
        IF (N401-40) 320,460,460
  320      DO 330 I=1,N
              IF ( ABS(X2(I)/X(I)-1.0 ).GE.1.0E-7 )  GO TO 340
  330      CONTINUE
           GO TO 460
C
  340      IF ( ABS( PX1/PX2-1.0 ) .LE. 1.0E-7 )  GO TO 460
           IF ( PX1-PX2 )  350,460,400
C
C       THROW AWAY RIGHT PART
  350          DO 360 I=1,N
                  X1(I) = X(I)
  360          CONTINUE
C
                P1 = PX2
                DO 370 I=1,N
C           POINT XP1 BECOMES XP2 TEMPORARILY IN X STORAGE
                   X(I) = 0.38196601 * ( X1(I)-X3(I) ) + X3(I)
  370           CONTINUE
C
                CALL EVALU
                GO TO (540,380,170), NSATIS
C
  380           CONTINUE
                PX2 = PX1
C
C           SWITCH VECTORS TO PROPER POSITION
                PX1=PO
                DO 390 I=1,N
                   XX = X2(I)
                   X2(I) = X(I)
                   X(I) = XX
  390           CONTINUE
                GO TO 300
C
C       LEFT SIDE TOSSED AWAY
C       CHANGES FOR NONUNIMODAL FUNCTION.  GO TO THROW AWAY RIGHT
C       IN CASE INITIAL VALUE LESS THAN FEASIBLE POINT.
  400   IF (PREV3-PX2) 350,350,410
  410   DO 420 I=1,N
           X3(I) = X2(I)
           X2(I) = X(I)
  420   CONTINUE
```

```
C
      PREV3=PX1
      PX1=PX2
  430 DO 440 I=1,N
         X(I) = 0.38196601 * ( X1(I)-X2(I) ) + X2(I)
  440 CONTINUE
C
      CALL EVALU
      GO TO (540,450,170), NSATIS
C
  450 CONTINUE
      PX2=P0
      GO TO 300
C
C     THE INTERIOR POINTS NOW GIVE EQUAL VALUE FOR P.  COMPUTE MIDPOINT.
  460 DO 470 I=1,N
         DELX0(I) = X(I)
         X(I) = ( DELX0(I) + X2(I) ) * 0.5
  470 CONTINUE
C
      CALL EVALU
      GO TO (480,490), KSW
C
  480 IF ( ABS( P0/PX1-1.0 ) .GT.1.0E-07) GO TO 520
  490 GO TO (500,510), ISW
  500 IF (P0.LT.P31) GO TO 510
      ISW=2
C     IF P-FUNCTION DIDN'T GO DOWN, TRY NEGATIVE VECTOR.
      GO TO 20
C
  510 RETURN
C
  520 DO 530 I=1,N
         X(I) = DELX0(I)
  530 CONTINUE
      GO TO 350
C
C     WE ARE NOW IN FEASIBILITY PHASE
  540 DO 550 I=1,M
         IF ( RJ(I) ) 560,560,550
  550 CONTINUE
C
      NSATIS = 4
      RETURN
C
C     PROBLEM HAS BECOME FEASIBLE
C     P - FUNCTION CHANGES IF A CONSTRAINT BECOMES FEASIBLE
  560 MN=0
      DO 570 I=1,M
         RJ1(I) = RJ(I)
  570 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE OUTPUT (K)
C
C     OUTPUT PRINTS OUT INFORMATION ON THE RESULTS OF EACH ITERATION
C
      INTEGER  CONSOL, PRINTR
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /EQAL/ H, H1, MZ
      COMMON /OPTNS/ NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
      COMMON /VALUE/ F,G,P0,RSIGMA,RJ(20),RHO
      COMMON /CRST/ DELX(20),DELX0(20),RHOIN,RATIO,EPSI,THETA0,
     1 RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2 PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3 PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
      COMMON /DEVC/ CONSOL, PRINTR, NP
C
      NZ = M + MZ
      GO TO (10,20), K
C
   10 WRITE (PRINTR,1) NTCTR
      WRITE (PRINTR,2) RHO, RSIGMA
   20 WRITE (PRINTR,3) F,P0,G
      WRITE (PRINTR,4)
      WRITE (PRINTR,5)  ( J, X(J), J=1,N )
      WRITE (PRINTR,6)
      GO TO (30,40), NT2
C
   30    WRITE (PRINTR,8) ( I, RJ(I), I=1,NZ )
         GO TO 50
C
   40    WRITE (PRINTR,8) ( I, RJ(I), I=1,NZ )
C
    1 FORMAT (///, 8X, 18H *** POINT NUMBER  ,I5, 8H    ***    )
    2 FORMAT (/, 2X, 6HRHO = ,E14.7, 4X, 9HRSIGMA = ,E14.7 )
    3 FORMAT (/, 2X, 3HF =,E14.7, 4X, 3HP =,E14.7, 4X, 3HG =,E14.7)
    4 FORMAT (/, 2X, 18HVALUES OF X VECTOR )
    5 FORMAT (1X, 3(2X,2HX(, I2, 3H) =,E14.7) )
    6 FORMAT (/, 2X, 25HVALUES OF THE CONSTRAINTS  )
    8 FORMAT (1X, 3(3X, 2HG(, I2, 3H) = ,E14.7) )
C
   50 RETURN
      END



      SUBROUTINE PEVALU
C
C     PEVALU COMPUTES THE VALUE OF THE PENALTY FUNCTION AND THE VALUE
C     OF THE DUAL USING PREVIOUSLY COMPUTED VALUES FOR F AND RJ.
C
      INTEGER  CONSOL, PRINTR
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /EQAL/ H, H1, MZ
      COMMON /OPTNS/ NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
      COMMON /VALUE/ F,G,P0,RSIGMA,RJ(20),RHO
      COMMON /CRST/ DELX(20),DELX0(20),RHOIN,RATIO,EPSI,THETA0,
```

```
     1   RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2   PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3   PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
        COMMON /DEVC/ CONSOL, PRINTR, NP
C
        H=0.0
        RSIGMA=0.0
C         NONNEGS IF INCLUDED ARE ADDED TO P-- ARE POSITIVE IN ALL PHASES
        GO TO (10,30), NT2
C
   10      DO 20 I=1,N
              RSIGMA = RSIGMA - RHO*ALOG(X(I))
   20      CONTINUE
C
   30      GO TO (40,50,150), NPHASE
C
C       OBJECTIVE FUNCTION  -  SIGMA VIOLATED CONSTRAINTS
   40         F = 0.0
   50         IF (M) 100,100,60
   60            DO 90 J=1,M
                    IF (RJ(J)) 80,80,70
   70                  RSIGMA = RSIGMA - RHO*ALOG( RJ(J))
                       GO TO 90
C
   80                  F = F - RJ(J)
   90            CONTINUE
C
C       EQUALITIES NOT ADDED IN FEASIBILITY PHASE
  100            CONTINUE

                 IF (MZ) 140,140,110
  110               GO TO (140,120,150), NPHASE
C
  120                  DO 130 I=1,MZ
                          K=M+I
                          H = H + RJ(K)**2
  130                  CONTINUE
                       H = H / RHO
C
  140                  HS = H + RSIGMA
                       PO = F + HS
                       HMS = 2.0 * H  -  RHO*FLOAT(M)
                       G = F + HMS
                       IF (NT2.EQ.1) G = G - RHO*FLOAT(N)
C
  150   RETURN
        END
```

```
      SUBROUTINE REJECT
C
C     REJECT RETURNS THE STORED VALUES OF THE OBJECTIVE FUNCTION, THE
C     CONSTRAINT FUNCTION AND THE PENALTY FUNCTION TO THEIR NORMAL
C     LOCATION.
C
      INTEGER  CONSOL, PRINTR
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /EQAL/ H, H1, MZ
      COMMON /VALUE/ F,G,PO,RSIGMA,RJ(20),RHO
      COMMON /CRST/ DELX(20),DELXO(20),RHOIN,RATIO,EPSI,THETAO,
     1 RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2 PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3 PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
      COMMON /DEVC/ CONSOL, PRINTR, NP
C
      DO 10 I=1,N
         X(I) = X1(I)
   10 CONTINUE
C
      MMZ=M+MZ
      DO 20 J=1,MMZ
         RJ(J) = RJ1(J)
   20 CONTINUE
C
      PO=P1
      RSIGMA = RSIG1
      G=G1
      F=F1
      H=H1
C
      RETURN
      END
```

```
      SUBROUTINE RHOCOM
C
C     RHOCOM COMPUTES THE INITIAL R VALUE IF DESIRED
C
      INTEGER  CONSOL, PRINTR
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /OPTNS/ NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
      COMMON /VALUE/ F,G,P0,RSIGMA,RJ(20),RHO
      COMMON /CRST/ DELX(20),DELX0(20),RHOIN,RATIO,EPSI,THETA0,
     1 RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2 PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3 PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
      COMMON /DEVC/ CONSOL, PRINTR, NP
C
      GO TO (110,50,10,190), NT1
   10    RHO = RHOIN
   20    IF (RHO) 30,30,40
   30    RHO = 1.0
   40    RETURN
C
   50 NPAR1 = 1
   60 RHO = 1.0
C       NT1=2  MEANS RHO WHICH MINIMIZES GRADIENT MAGNITUDE
      CALL GRAD (2)
      DO 70 I=1,N
         PGRAD(I) = DELX0(I)
   70 CONTINUE
      RHO = 2.0
      CALL GRAD (2)
      DO 80 I=1,N
         DELX0(I) = DELX0(I) - PGRAD(I)
         PGRAD(I) = PGRAD(I) - DELX0(I)
   80 CONTINUE
C
      GO TO (90,130), NPAR1
   90 DOT1 = 0.0
      DOT2 = 0.0
      DO 100 I=1,N
         DOT1 = DOT1 + DELX0(I) * PGRAD(I)
         DOT2 = DOT2 + DELX0(I)**2
  100 CONTINUE
      RHO = ABS(DOT1/DOT2)
      GO TO 20
C
C       NT1=3  MEANS COMPUTE RHO SO AS TO MINIMIZE DELP (/DDP/1.) DEL P
  110 NPAR2 = 1
  120 NPAR1 = 2
      GO TO 60
  130 RHO = 1.0
C       ASSUME SIGMA TERM IS CONSIDERABLE GREATER THAN F TERM
      CALL SECORD (2)
      DO 140 I=1,N
         DELX(I) = PGRAD(I)
  140 CONTINUE
      CALL INVERS (1)
```

```
      DO 150 I=1,N
         X1(I) = DELX(I)
         DELX(I) = DELXO(I)
  150 CONTINUE
      CALL SECORD (2)
      CALL INVERS (1)
      DO 160 I=1,N
         XR2(I) = DELX(I)
  160 CONTINUE
      GO TO (170,200), NPAR2
  170 DOT1 = 0.0
      DOT2 = 0.0
      DO 180 I=1,N
         DOT1 = DOT1 + PGRAD(I) * X1(I)
         DOT2 = DOT2 + DELXO(I) * XR2(I)
  180 CONTINUE
      RHO = SQRT( ABS(DOT1/DOT2) )
      GO TO 20
C
C        RHO MINIMIZES 2ND ORDER MOVE
  190 NPAR2 = 2
      GO TO 120
C
  200 DOT1 = 0.0
      DOT2 = 0.0
      DO 210 I=1,N
         DOT1 = X1(I)**2 + DOT1
         DOT2 = X1(I)*XR2(I) + DOT2
  210 CONTINUE
      RHO = ABS(DOT1/DOT2)
      GO TO 20
C
      END



      SUBROUTINE SECORD (IS)
C
C     SECORD EVALUATES THE MATRIX OF SECOND PARTIALS OF THE PENALTY
C     FUNCTION.
C     (1)  MEANS DON'T COMPUTE GRADIENT OUTER PRODUCT ( IN SECORD).
C
      INTEGER  CONSOL, PRINTR
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /EQAL/ H, H1, MZ
      COMMON /OPTNS/ NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
      COMMON /VALUE/ F,G,PO,RSIGMA,RJ(20),RHO
      COMMON /CRST/ DELX(20),DELXO(20),RHOIN,RATIO,EPSI,THETAO,
     1 RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2 PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3 PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
      COMMON /DEVC/ CONSOL, PRINTR, NP
C
      DO 10 I=1,N
      DO 10 J=1,N
```

```
            A(I,J) = 0.0
   10    CONTINUE
C
         GO TO (230,20), IS
C
C        GRADIENT TERM NOT PREVIOUSLY COMPUTED.
   20    DO 30 I=1,N
         DO 30 J=1,I
            A(I,J) = 0.0
   30    CONTINUE
C
         GO TO (40,60), NT2
C
   40    DO 50 I=1,N
            A(I,I) = RHO / X(I)**2
   50    CONTINUE
C
   60    CONTINUE
         IF (M.LE.0) GO TO 130
         DO 120 IN=1,M
            IF ( RJ(IN)) 120,120,70
   70          CALL GRAD1(IN)
               TT = RHO / RJ(IN)**2
               DO 110 I=1,N
                  IF ( DEL(I)) 80,110,80
   80                T = TT * DEL(I)
                     DO 100 J=1,I
                        IF ( DEL(J)) 90,100,90
   90                      A(I,J) = A(I,J) + T * DEL(J)
  100                CONTINUE
  110          CONTINUE
  120    CONTINUE
C
C           EQUALITY CONSTRAINTS
  130    IF (MZ) 210,210,140
  140    GO TO (210,150,230), NPHASE
C
  150       RQ = 2.0 / RHO
            DO 200 JJ=1,MZ
               IN = M + JJ
               CALL GRAD1 (IN)
               DO 190 I=1,N
                  IF ( DEL(I)) 160,190,160
  160                T = RQ * DEL(I)
                     DO 180 J=1,I
                        IF ( DEL(J)) 170,180,170
  170                      A(I,J) = A(I,J) + T*DEL(J)
  180                CONTINUE
  190          CONTINUE
  200       CONTINUE
C
  210       DO 220 I=1,N
               DIAG(I) = A(I,I)
               A(I,I) = 0.0
  220       CONTINUE
```

```
C
C        READY NOW FOR MATRIX OF 2ND PARTIALS OF RESTRAINTS
  230       GO TO (240,510,520), NT10
C
  240          IF (M.LE.0) GO TO 340
               DO 330 IN=1,M
                  LORN = 2
C                  CONSTRAINT ASSUMED NONLINEAR
                  CALL MATRIX (IN,LORN)
                  IF (LORN.LT.2) GO TO 330
                  IF ( RJ(IN).GT.0.0 )  GO TO 280
                  DO 261 I=2,N
                     IM1 = I - 1
                     DO 260 J=1,IM1
                        IF ( A(J,I)) 250,260,250
  250                      A(I,J) = A(I,J) + A(J,I)
                           A(J,I) = 0.0
  260                CONTINUE
  261             CONTINUE
C
                  DO 270 I=1,N
                     DIAG(I) = DIAG(I) - A(I,I)
                     A(I,I) = 0.0
  270             CONTINUE
                  GO TO 330
C
  280             T = - RHO / RJ(IN)
                  DO 301 I=2,N
                     IM1 = I - 1
                     DO 300 J=1,IM1
                        IF ( A(J,I)) 290,300,290
  290                      A(I,J) = A(I,J) + T*A(J,I)
                           A(J,I) = 0.0
  300                CONTINUE
  301             CONTINUE
C
                  DO 320 I=1,N
                     IF ( A(I,I) ) 310,320,310
  310                   DIAG(I) = DIAG(I) +  T*A(I,I)
                        A(I,I) = 0.0
  320             CONTINUE
  330       CONTINUE
C
  340    CONTINUE
         GO TO (520,350,520), NPHASE
  350       IF (MZ.EQ.0) GO TO 420
C
C             EQUALITY SECOND PARTIALS HERE
            IF (NT10.GE.2) GO TO 420
            DO 410 II=1,MZ
               IN = M + II
               LORN=2
               CALL MATRIX (IN,LORN)
               IF (LORN.LT.2) GO TO 410
               T = 2.0 * RJ(IN) / RHO
```

```
         DO 380 I=2,N
            IM1 = I-1
            DO 370 J=1,IM1
               IF ( A(J,I)) 360,370,360
 360              A(I,J) = A(I,J) + T*A(J,I)
                  A(J,I) = 0.0
 370        CONTINUE
 380     CONTINUE
C
         DO 400 I=1,N
            IF ( A(I,I)) 390,400,390
 390           DIAG(I) = DIAG(I) + T*A(I,I)
               A(I,I)=0.0
 400     CONTINUE
C
 410   CONTINUE
C
C      GET MATRIX OF 2ND PARTIALS OF OBJECTIVE FUNCTION
 420  LLL=2
      CALL MATRIX (0,LLL)
      IF (LLL.LT.2) GO TO 490
      DO 441 I=2,N
         IM1=I-1
         DO 440 J=1,IM1
            IF ( A(J,I)) 430,440,430
 430           A(I,J) = A(I,J) + A(J,I)
 440     CONTINUE
 441  CONTINUE
C
      DO 470 I=1,N
         IF ( A(I,I)) 450,460,450
 450        A(I,I) = DIAG(I) + A(I,I)
            GO TO 470
C
 460        A(I,I) = DIAG(I)
 470  CONTINUE
 480  RETURN
C
 490  DO 501 I=1,N
         A(I,I) = DIAG(I)
         DO 500 J=I,N
            A(I,J) = A(J,I)
 500     CONTINUE
 501  CONTINUE
      GO TO 480
C
 510  GO TO (520,350,350), NPHASE
 520     DO 531 I=2,N
            IM1=I-1
            DO 530 J=1,IM1
               A(J,I) = A(I,J)
 530        CONTINUE
 531     CONTINUE
         DO 540 I=1,N
            A(I,I) = DIAG(I)
```

```
  540     CONTINUE
          GO TO 480
C
      END



      SUBROUTINE STORE
C
C     STORE STORES THE VALUES OF THE CURRENT POINT AND THE
C     ASSOCIATED VALUES OF THE FUNCTION IN A TEMPORARY AREA.
C
      INTEGER  CONSOL, PRINTR
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /EQAL/ H, H1, MZ
      COMMON /VALUE/ F,G,PO,RSIGMA,RJ(20),RHO
      COMMON /CRST/ DELX(20),DELXO(20),RHOIN,RATIO,EPSI,THETAO,
     1 RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2 PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3 PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
      COMMON /DEVC/ CONSOL, PRINTR, NP
C
      DO 10 I=1,N
         X1(I) = X(I)
   10 CONTINUE
C
      MMZ = M + MZ
      DO 20 J=1,MMZ
         RJ1(J) = RJ(J)
   20 CONTINUE
C
      P1=PO
      F1=F
      G1=G
      RSIG1=RSIGMA
      H1=H
C
      RETURN
      END
```

```
      SUBROUTINE XMOVE
C
C     XMOVE DETERMINES THE VECTOR ALONG WHICH THE SEARCH FOR A MINIMUM
C     IS USING OPT.
C     NEXOP2  DETERMINES HOW MOVE IS TO BE MADE
C        1  USE MODIFIED  NEWTON RAPHSON METHOD.
C        2  USE MODIFIED  NEWTON RAPHSON METHOD, BUT ADD DELXO TO
C              ORTHOGONAL MOVE VECTOR IF HESSIAN IS INDEFINITE.
C        3  USE  STEEPEST DESCENT METHOD.
C        4  USE  MCCORMICK'S MODIFICATION OF THE FLETCHER-POWELL METHOD.
C
      INTEGER  CONSOL, PRINTR
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /CRST/ DELX(20),DELXO(20),RHOIN,RATIO,EPSI,THETAO,
     1 RSIG1, G1, X1(20), X2(20), X3(20), XR2(20), XR1(20), PR1,
     2 PR2, P1, F1, RJ1(40), DOTT, PGRAD(20), DIAG(20),
     3 PREV3, ADELX, NTCTR, NUMINI, NPHASE, NSATIS
      COMMON /EXPOPT/ NEXOP1, NEXOP2, XEP1, XEP2
      COMMON /XVE/ SIG(20), YY(20), XXX(20), DELL(20)
      COMMON /DEVC/ CONSOL, PRINTR, NP
C
      GO TO (10,10,180,30), NEXOP2
C
C          NEWTON-RAPHSON WITH WHATEVER METHOD IS IN INVERSE
   10    CALL GRAD(1)
C          ONE (1) MEANS ACCUMULATE MATRIX OF SECOND PARTIAL DERIVATIVES
         CALL SECORD(1)
         DO 20 I=1,N
            DELX(I) = DELXO(I)
   20    CONTINUE
         CALL INVERS(1)
C           IF A NONPOSITIVE PIVOT IS ENCOUNTERED IN INVERSE, AN ATTEMPT
C             IS MADE TO COMPUTE A VECTOR HAVING A POSITIVE DOT PRODUCT
C             WITH A NEGATIVE EIGENVECTOR AND THE NEGATIVE OF DEL P.
         CALL STORE
         CALL OPT
         RETURN
C
   30 CALL GRAD (2)
C       MN IS NO. OF MOVES FOR THIS VALUE OF RHO
      IF (MN.NE.0) GO TO 70
   40 IREP=0
      IT=0
C       SET INITIAL GUESS INVERSE MATRIX OF SECOND PARTIAL DERIVATIVES
C       USE PARTIAL INVERSE IF KNOWN
      DO 50 I=1,N
      DO 50 J=1,N
         A(I,J) = 0.0
   50 CONTINUE
C
      DO 60 I=1,N
         A(I,I) = 1.0
   60 CONTINUE
C
```

```
   70   DO 80 I=1,N
           DELX(I) = DELXO(I)
   80   CONTINUE
C
        IF (IREP.GT.N) GO TO 40
        IF (IT.EQ.0) GO TO 130
C
        DO 90 I=1,N
           SIG(I) = X(I) - XXX(I)
           YY(I) = DELL(I) -DELXO(I)
   90   CONTINUE
C
C       NEGATIVE GRADIENT STORED AND COMPUTED.  COMPUTE HY.
        DO 101 I=1,N
           DELX(I) = 0.0
           DO 100 J=1,N
              DELX(I) = DELX(I) + A(I,J)*YY(J)
  100      CONTINUE
  101   CONTINUE
C
C          COMPUTE Y(SIG-HY) - 1
        ZCON=0.0
        DO 110 I=1,N
           ZCON = ZCON  +  YY(I) * ( SIG(I) - DELX(I) )
  110   CONTINUE
C
        IF (ZCON.EQ.0.0) GO TO 130
        IREP = IREP + 1
        ZC = 1.0 / ZCON
C
C          UPDATE H MATRIX USING MCC FORMULA WHEN SCALAR NOT EQUAL TO ZERO
        DO 121 I=1,N
           T1 = ZC * ( SIG(I) - DELX(I) )
           DO 120 J=1,N
              A(I,J) = A(I,J)  +  T1 * ( -DELX(J)+SIG(J) )
              A(J,I) = A(I,J)
  120      CONTINUE
  121   CONTINUE
C
C          STORE CURRENT POINT AND CURRENT GRADIENT (NEG)
  130   DO 140 I=1,N
           XXX(I) = X(I)
           DELL(I) = DELXO(I)
  140   CONTINUE
C
        DO 151 I=1,N
           DELX(I) = 0.0
           DO 150 J=1,N
              DELX(I) = DELX(I)  +  A(I,J) * DELXO(J)
  150      CONTINUE
  151   CONTINUE
C
        ZC1 = 0.0
        DO 160 I=1,N
           ZC1 = DELX(I)**2 + ZC1
```

```
  160   CONTINUE
C
        ZC1 = SQRT(ZC1)
        DO 170 I=1,N
           DELX(I) = DELX(I) / ZC1
  170   CONTINUE
C
        CALL STORE
        CALL OPT
        IT = IT + 1
        RETURN
C
  180   CONTINUE
C
C          STEEPEST DESCENT
        CALL GRAD(2)
        DO 190 I=1,N
           DELX(I) = DELXO(I)
  190   CONTINUE
C
        CALL STORE
        CALL OPT
C
        RETURN
        END
```

```
      PROGRAM READIN
C
C                ** RAC SUMT  INPUT PROGRAM  **
C              ****************************************
C
C           THIS INPUT PROGRAM IS USED TO ENTER ALL DATA NEEDED BY THE
C        MAIN PROGRAM.  IT ALLOWS INPUT TO BE ENTERED FROM THE KEYBOARD
C        IN AN INTERACTIVE MANNER.
C
C          THE PROGRAM IS WRITTEN BY :  FRANK HWANG,  I.E.,  KSU,  1983.
C
C        ***************************************************************
C
C
      LOGICAL  NAME(60)
      INTEGER  OPTION, CONSOL, PRINTR
      REAL  X(20)
C
      DATA  CONSOL,PRINTR /1,2/
      DATA  NT1,NT2,NT3,NT4,NT5 / 3,1,1,1,2/
      DATA  NT6,NT7,NT8,NT9,NT10 /1,1,1,1,1/
C
      WRITE (CONSOL,199)
      WRITE (PRINTR,199)
      WRITE (CONSOL,197)
      READ  (CONSOL,196) NAME
      WRITE (PRINTR,195) NAME
C
      WRITE (CONSOL,194)
      READ (CONSOL,193) N
C
      WRITE (CONSOL,189)
      READ (CONSOL,193) M
      WRITE (CONSOL,187)
      READ (CONSOL,193) MZ
      WRITE (CONSOL,185)  N, M, MZ
      WRITE (PRINTR,185) N, M, MZ
C
      WRITE (CONSOL,182)
      DO 50 I=1,N
         WRITE (CONSOL,181) I
         READ (CONSOL,180) X(I)
   50 CONTINUE
C
C    * ECHO CHECK INITIAL POINT
      WRITE (CONSOL,178)  ( I, X(I), I=1,N)
C
C    * DEFAULT VALUES OF THE PARAMETERS
C
      RHO = 1.0
      RHOIN = RHO
      RATIO = 4.0
      EPSI = 0.1E-4
      THETA0 = 0.1E-2
C
```

```
      NT1=3
      NT2=1
      NT3=1
      NT4=1
      NT5=2
      NT6=1
      NT7=1
      NT8=1
      NT9=1
      NT10=1
C
      NEXOP1 = 1
      NEXOP2 = 1
C
      WRITE (CONSOL,175)
   60 WRITE (CONSOL,174)
      READ (CONSOL,173) OPTION
      IF (OPTION.LE.0) GO TO 70
C
      GO TO (1,2,3,4,5,6,7,8,9,10), OPTION
C
    1 WRITE (CONSOL,170)
      READ  (CONSOL,169) NT1
      IF (NT1.NE.3) GO TO 21
C
      WRITE (CONSOL,168)
      READ (CONSOL,167) RHOIN
      IF (R.LE.0.0) RHOIN = 1.0
   21 IF (NT1.LE.0) NT1 = 3
      IF (OPTION.NE.99) GO TO 60
C
    2 WRITE (CONSOL,160)
      READ (CONSOL,167) RATIO
      IF (RATIO.LE.1.0) RATIO = 4.0
      IF (OPTION.NE.99) GO TO 60
C
    3 WRITE (CONSOL,159)
      READ  (CONSOL,167) EPSI
      IF (EPSI.LE.0.0) EPSI = 0.1E-4
      IF (OPTION.NE.99) GO TO 60
C
    4 WRITE (CONSOL,158)
      READ (CONSOL,167) THETAO
      IF (THETAO.LE.0) THETAO = 0.1E-2
      IF (OPTION.NE.99) GO TO 60
C
    5 WRITE (CONSOL,155)
      READ (CONSOL,154) NT2
      IF ( (NT2.LE.0).OR.(NT2.GT.2) )  NT2=1
      IF (OPTION.NE.99) GO TO 60
C
    6 WRITE (CONSOL,150)
      READ (CONSOL,154) NT5
      IF ( (NT5.LE.0).OR.(NT5.GT.2) )  NT5 = 2
      IF (OPTION.NE.99) GO TO 60
```

```
C
    7   WRITE (CONSOL,149)
        READ (CONSOL,154) NT9
        IF ( (NT9.LE.0).OR.(NT9.GT.3) )  NT9 = 1
        IF (OPTION.NE.99) GO TO 60
C
    8   WRITE (CONSOL,147)
        READ (CONSOL,154) NT7
        IF ( (NT7.LE.0).OR.(NT7.GT.3) )  NT7 = 1
        IF (OPTION.NE.99) GO TO 60
C
    9   WRITE (CONSOL,145)
        READ (CONSOL,154) NEXOP1
        IF ( (NEXOP1.LE.0).OR.(NEXOP1.GT.5) ) NEXOP1 = 1
        IF (OPTION.NE.99) GO TO 60
C
   10   WRITE (CONSOL,144)
        READ (CONSOL,154) NEXOP2
        IF ( (NEXOP1.LE.0).OR.(NEXOP2.GT.4) )  NEXOP2 = 1
C
C     * ECHO CHECK OPTIONS CHOSEN
   70   WRITE (CONSOL,143)
        WRITE (PRINTR,143)
C
   75   GO TO (76,77,78), NT1
   76     WRITE (CONSOL,109)
          WRITE (PRINTR,109)
          GO TO 79
C
   77     WRITE (CONSOL,108)
          WRITE (PRINTR,108)
          GO TO 79
C
   78     WRITE (CONSOL,110) RHOIN
          WRITE (PRINTR,110) RHOIN
C
   79   WRITE (CONSOL,140) RATIO, EPSI, THETAO
        WRITE (PRINTR,140) RATIO, EPSI, THETAO
C
        GO TO (80,81), NT2
   80     WRITE (CONSOL,138)
          WRITE (PRINTR,138)
          GO TO 82
C
   81     WRITE (CONSOL,137)
          WRITE (PRINTR,137)
C
   82   GO TO (83,84), NT5
   83      WRITE (CONSOL,135)
           WRITE (PRINTR,135)
           GO TO 85
C
   84      WRITE (CONSOL,134)
           WRITE (PRINTR,134)
C
```

```
   85   GO TO (86,87,88), NT9
   86      WRITE (CONSOL,132)
           WRITE (PRINTR,132)
           GO TO 89
C
   87      WRITE (CONSOL,131)
           WRITE (PRINTR,131)
           GO TO 89
C
   88      WRITE (CONSOL,130)
           WRITE (PRINTR,130)
C
   89   GO TO (90,91,92), NT7
   90      WRITE (CONSOL,128)
           WRITE (PRINTR,128)
           GO TO 93
C
   91      WRITE (CONSOL,127)
           WRITE (PRINTR,127)
           GO TO 93
C
   92      WRITE (CONSOL,126)
           WRITE (PRINTR,126)
C
   93   GO TO (94,95,96,97,98), NEXOP1
   94      WRITE (CONSOL,125)
           WRITE (PRINTR,125)
           GO TO 99
C

   95      WRITE (CONSOL,124)
           WRITE (PRINTR,124)
           GO TO 99
C
   96      WRITE (CONSOL,123)
           WRITE (PRINTR,123)
           GO TO 99
C
   97      WRITE (CONSOL,122)
           WRITE (PRINTR,122)
           GO TO 99
C
   98      WRITE (CONSOL,121)
           WRITE (PRINTR,121)
C
   99   GO TO (100,101,102,103), NEXOP2
C
  100      WRITE (CONSOL,119)
           WRITE (PRINTR,119)
           GO TO 105
C
  101      WRITE (CONSOL,118)
           WRITE (PRINTR,118)
           GO TO 105
C
```

```
  102      WRITE (CONSCL,117)
           WRITE (PRINTR,117)
           GO TO 105
C
  103      WRITE (CONSOL,116)
           WRITE (PRINTR,116)
C
  105   CALL OPEN (6,'OPTIONS DAT',2)
           WRITE (6) N,M,MZ
           WRITE (6) ( X(I), I=1,N )
           WRITE (6) RHOIN, RATIO, EPSI, THETA0
           WRITE (6) NT1,NT2,NT3,NT4,NT5,NT6,NT7,NT8,NT9,NT10
           WRITE (6) NEXOP1,NEXOP2
        ENDFILE 6
C
        CALL FCHAIN ('RACSUMT COM',2)
C
C
  199   FORMAT (//,20X,'RAC-SUMT --- VERSION 4.1'/)
  197   FORMAT (' ',5X,'PROBLEM NAME : ')
  196   FORMAT (60A1)
  195   FORMAT ('0',12X,60A1)
  194   FORMAT ('0',5X,'NUMBER OF VARIABLES : ')
  193   FORMAT (I2)
  189   FORMAT (' ',5X,'NUMBER OF INEQUALITY CONSTRAINTS',
    1            ' ( G(X) >= 0 ) : ')
  187   FORMAT (' ',5X,'NUMBER OF EQUALITY CONSTRAINTS',
    1            ' ( H(X) = 0 ) : ')
  185   FORMAT ('0',' N =',I3, 4X, 'M =',I3, 4X, 'MZ =',I3)
  182   FORMAT ('0',15X,'ENTER THE INITIAL POINT : '/)
  181   FORMAT (' ',5X, ' X(',I2,') = ')
  180   FORMAT (G15.4)
  178   FORMAT (1X,3(2X,'X(',I2,') =',E14.7) )
C
  175   FORMAT (/,8X,'The default values for the ',
    1           ' parameters follow :'/
    1   5X,'1)  R = 1.0 '/
    2   5X,'2)  C = 4.0 '/
    3   5X,'3)  EPSI = 0.1E-4'/
    4   5X,'4)  THETA = 0.1E-2'/
    5   5X,'5)  Constraint option --- include X(I) >= 0 constraints'/
    6   5X,'6)  Final convergence criterion :  RSIGMA < THETA '/
    7   5X,'7)  Subproblem convergence criterion #1: DELP < EPSI'/
    8   5X,'8)  No extrapolation'/
    9   5X,'9)  No checking for derivatives'/
    1   5X,'10) Unconstrained minimization technique : Second order',
    1            ' gradient method'/
    2   5X,'    press RETURN to use all default values'/
    3   5X,'    Enter option number (1,2,...,10) to change one or more',
    3      ' options')
  174   FORMAT (/,5X,'ENTER option number (RETURN if finished) : ')
  173   FORMAT (I2)
C
```

```
170   FORMAT (/,5X,'1)  R -- penalty factor '/
   1             5X,'      ( RETURN  for  R = 1.0 )'/
   1             5X,'          1    R computed by formula 1',
   1                                 ' (see User''s guide)'/
   2             5X,'          2    R computed by formula 2',
   2                                 ' (see User''s guide)'/
   3             5X,'          3    specify own value of R'/
   2             5X,'    R option code = ' )
169   FORMAT (I1)
168   FORMAT (/,5X,'1)  R = ')
167   FORMAT (G15.7)
165   FORMAT (/,5X,'ENTER option number (RETURN if finished) : ')
164   FORMAT (I2)
160   FORMAT (/,5X,'2)  C -- Reducing factor for R from stage ',
   1            'to stage '/
   1             5X,'      ( RETURN  for C = 4.0 )'/
   2             5X,'    C = ' )
159   FORMAT (/,5X,'3)  EPSI --- subproblem stopping value '/
   1             5X,'      ( RETURN  for EPSI = 0.1E-4 )'/
   2             5X,'    EPSI = ' )
158   FORMAT (/,5X,'4)  THETA --- final stopping value '/
   1             5X,'       ( RETURN  for THETA = 0.1E-2 )'/
   2             5X,'    THETA = ' )
155   FORMAT (/,5X,'5)  Constraint option '/
   1             5X,'        1   include  X(I) >= 0  constraints'/
   2             5X,'        2   do not include X(I) >= 0',
   2                                           ' constraints'/
   3             5X,'     ENTER option : ')
154   FORMAT (I1)
150   FORMAT (/,5X,'6)  Final convergence criterion '/
   1             5X,'       1   ABS[ F(X)/G ] - 1 < THETA '/
   2             5X,'       2   RSIGMA < THETA '/
   3             5X,'    Final convergence criterion = ')
149   FORMAT (/,5X,'7)  Subproblem convergence criterion'/
   1             5X,'        1    see User''s guide'/
   2             5X,'        2    see User''s guide'/
   3             5X,'        3    gradient of P < EPSI'/
   4             5X,'    Subproblem convergence criterion = ')
147   FORMAT (/,5X,'8)  Extrapolation option'/
   1             5X,'       1   No extrapolation'/
   2             5X,'       2   Extrapolate through last 2 minima'/
   3             5X,'       3   Extrapolate through last 3 minima'/
   4             5X,'    Extrapolation option = ')
145   FORMAT (/,5X,'9)  Key for checking derivatives '/
   1             5X,'       1  Do not check derivatives '/
   2             5X,'       2  Solve problem after checking',
   2                         ' first derivatives'/
   3             5X,'       3  Check first derviatives but ',
   3                          'do not solve problem'/
   4             5X,'       4  Solve problem after checking ',
   4                          '1st and 2nd derivatives'/
   5             5X,'       5  Check 1st and 2nd derivatives but ',
   5                          'do not solve problem'/
   6             5X,'     Key = ')
```

```
  144   FORMAT (/,5X,'10) Unconstrained minimization technique used'/
     1           5X,'        1    2nd order gradient method'/
     2           5X,'        2    same as 1 with modification'/
     3           5X,'        3    Steepest descent method'/
     4           5X,'        4    Modified Fletcher - Powell method'/
     5           5X,'     Method = ')
C
  143   FORMAT (/,2X,'OPTIONS SELECTED')
  140   FORMAT (2X,'2)  C =',E11.4 / 2X,'3)  EPSI =',E11.4 /
     1          2X,'4)  THETA =',E11.4 )
  138   FORMAT (2X,'5)  CONSTRAINT OPTION --- INCLUDE X(I) >= 0 ',
     1          'CONSTRAINTS')
  137   FORMAT (2X,'5)  CONSTRAINT OPTION --- DO NOT INCLUDE X(I) >= 0',
     2          ' CONSTRAINTS')
  135   FORMAT (2X,'6)  FINAL CONVERGENCE CRITERION --- ',
     1          'ABS[ F(X)/G ] - 1 < THETA')
  134   FORMAT (2X,'6)  FINAL CONVERGENCE CRITERION --- ',
     2          'RSIGMA < THETA')
  132   FORMAT (2X,'7)  SUBPROBLEM CONVERGENCE CRITERION #1 ')
  131   FORMAT (2X,'7)  SUBPROBLEM CONVERGENCE CRITERION #2 ')
  130   FORMAT (2X,'7)  SUBPROBLEM CONVERGENCE CRITERION #3 ')
  128   FORMAT (2X,'8)  NO EXTRAPOLATION')
  127   FORMAT (2X,'8)  EXTRAPOLATE THROUGH LAST 2 MINIMA')
  126   FORMAT (2X,'8)  EXTRAPOLATE THROUGH LAST 3 MINIMA')
  125   FORMAT (2X,'9)  NO CHECKING FOR DERIVATIVES')
  124   FORMAT (2X,'9)  SOLVE PROBLEM AFTER CHECKING FIRST DERIVATIVES')
  123   FORMAT (2X,'9)  CHECK FIRST DERIVATIVES BUT DO NOT SOLVE',
     1          ' PROBLEM')
  122   FORMAT (2X,'9)  SOLVE PROBLEM AFTER CHECKING 1ST AND 2ND ',
     1          'DERIVATIIVES')
  121   FORMAT (2X,'9)  CHECK 1ST AND 2ND DERIVATIVES ',
     2          'BUT DO NOT SOLVE PROBLEM')
  119   FORMAT (2X,'10) UNCONSTRAINED MINIMIZATION TECHNIQUE -- ',
     1          '2ND ORDER GRADIENT METHOD')
  118   FORMAT (2X,'10) UNCONSTRAINED MINIMIZATION TECHNIQUE -- ',
     2          'MODIFIED 2ND ORDER GRADIENT METHOD')
  117   FORMAT (2X,'10) UNCONSTRAINED MINIMIZATION TECHNIQUE -- ',
     3          'STEEPEST DESCENT METHOD')
  116   FORMAT (2X,'10) UNCONSTRAINED MINIMIZATION TECHNIQUE -- ',
     4          'MODIFIED FLETCHER - POWELL METHOD ')
  110   FORMAT (2X,'1)  R =',E11.4, 5X, '(USER SPECIFIED)' )
  109   FORMAT (2X,'1)  R TO BE COMPUTED BY FORMULA 1')
  108   FORMAT (2X,'1)  R TO BE COMPUTED BY FORMULA 2')
C
        STOP
        END
```

4.3.4 DESCRIPTION OF OUTPUT

The program title is printed followed by the name of the problem to be solved. Then the dimensions of the problem are printed where

N = the number of decision variables, M = the number of inequality constraints, and MZ = the number of equality constraints.

A list of options selected is next printed out. The options printed are :

1) R -- penalty factor

2) C -- reducing factor

3) EPSI -- subproblem stopping value

4) THETA -- final stopping value

5) Constraint option

6) Final convergence criterion

7) Subproblem convergence criterion

8) Extrapolation option

9) Key for checking derivatives

10) Unconstrained minimization technique chosen.

Following the list of options, the objective function value F is printed. Note that although the variables P and G are printed, they will always show a value of zero because they have not been computed. After the value of F, the initial point is printed followed by the values of the constraints at the initial point. Then the values of the user supplied analaytic and the computed numeric derivatives at the starting point are printed if the user specified it on option 9 (Key for checking derivatives).

After printing the derivatives, the program checks if the initial point is feasible and if necessary, it attempts to locate a feasible point. The feasible starting point is then printed along with the values of the objective function and constraints at the feasible starting point.

At each suboptimum point, the following results are printed. First the iteration counter identified as "Point Number" is printed. Then the value of r (RHO) and the value of the penalty term (RSIGMA) is printed where RSIGMA $= - r \sum_i \ln[g_i(x)] + r^{-1} \sum_j h_j^2(x)$. The next line contains the objective function value F, the P-function value P, and the dual value G at the suboptimum point. The values of the decision variable x is then printed followed by the values of the constraints.

At the optimum point, the value of the objective function F and the decision variable x are printed.

4.3.5 SUMMARY OF USER REQUIREMENTS

1. Create a file on disk that contains subroutines RESTNT, GRAD1 and MATRIX. (see the following section for a description of how to code these routines.)
2. Make an estimate of the optimum point which is to be used as the starting point for the search.

NOTE : The following steps will vary depending on the particular compiler used. The following applies if using Microsoft FORTRAN-80.

3. Compile subroutines RESTNT, GRAD1, AND MATRIX using the F80 command.

```
F80 =B:filename
```

where the letter B refers to the disk drive where the file resides and the filename is the name of the file containing the three subroutines.

4. Link edit the main program with the user supplied subroutines as follows:

```
L80 B:filename,B:RACSUMT/N,B:RACSUMT/E
```

Note that the user defined filename precedes the main program RACSUMT.

5. Run the program by typing

B:READIN

READIN is the input program that allows one to interactively enter the data needed to solve the problem. After the data is entered, READIN saves the data on the disk before chaining to the main program RACSUMT. RACSUMT then reads the data back from the disk and proceeds to solve the problem.

To resolve the problem with different input values, simply repeat step 5.

4.3.6 USER-SUPPLIED SUBROUTINES

Each user-supplied subroutine must contain the COMMON card :

```
COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
```

The user may use blank COMMON to transfer data between his subroutines.

In the subroutines, the parameter I and J identify which constraint is needed. For example, in RESTNT when I=0, the value of the objective function is needed; when I=1, constraint $g_1(x)$ is needed; when I=2, $g_2(x)$ is needed, etc.

The following problem is used to show how to code the user supplied subroutines.

Minimize $f(x) = x_1^2 + x_2^3 - x_1x_2$

subject to

$$g_1(x) = 8x_1 + x_2^2 - 15 \geq 0$$

$$g_2(x) = 5x_1^4 + x_2^3 - 20 \geq 0$$

$$h_1(x) = x_1^2 + x_2^2 - 25 = 0$$

$$x_i \geq 0, \quad i=1,2$$

RESTNT (I,VAL)

This subroutine defines the objective function (to be minimized), the inequality constraints (≥ 0), and the equality constraints ($=0$). The variable VAL must be assigned the equation of the objective function or constraint depending on the value of I.

When I=0, this routine must set VAL = f(x).

When $I=1,\ldots,m$, this routine must set VAL = $g_I(x)$.

When $I=m+1,\ldots,m+\ell$, this routine must set VAL = $h_I(x)$. Note that the equality constraints follow all inequality constraints.

The non-negativity constraints do not have to be coded if option 5 on the CRT display is set to 1. The variable x is located in the labeled COMMON region named SHARE.

The RESTNT routine for the example problem is shown below.

```
      SUBROUTINE RESTNT (I,VAL)
C
C     THIS ROUTINE DEFINES THE OBJECTIVE FUNCTION (TO BE MINIMIZED) AND
C     THE CONSTRAINTS ( >=0 AND =0 )
C
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
C
      IF (I.GT.0) GO TO 50
C
C     * THE OBJECTIVE FUNCTION TO BE MINIMIZED
        VAL = X(1)**2 + X(2)**3 - X(1)*X(2)
        RETURN
C
C     *** THE INEQUALITY AND EQUALITY CONSTRAINTS ***
   50 GO TO (1,2,3),I
C
C     * THE 1ST INEQUALITY CONSTRAINT  G1(X) >= 0
    1   VAL = 8.*X(1) + X(2)**2 - 15.
        RETURN
C
C     * THE 2ND INEQUALITY CONSTRAINT  G2(X) >= 0
    2   VAL = 5.*X(1)**4 + X(2)**3 - 20.
        RETURN
C
C     * THE EQUALITY CONSTRAINT  H1(X) = 0
    3   VAL = X(1)**2 + X(2)**2 - 25.
        RETURN
      END
```

GRAD1(I)

This subroutine defines the gradient of the objective function and constraints. When I=0, the gradient of the objective function is needed and when I>0, the gradient of the Ith constraint is needed. The values of the gradient are placed in the array DEL(J) where DEL(J) is the Jth partial derivative of the Ith constraint.

For I=0, this routine must set DEL(J) = $\partial f(x)/\partial x_j$, j=1,...,n.
For I=1,...,m, this routine must set DEL(J) = $\partial g_I/\partial x_j$, j=1,...,n.
For I=m+1,...,m+ℓ, this routine must set DEL(J) = $\partial h_I(x)/\partial x_j$, j=1,...,n.

X and DEL are in the COMMON region SHARE. DEL is not initialized to zero before entering GRAD1 so all elements of DEL must be assigned a value, including the zero elements.

The GRAD1 routine for the example problem is shown below.

```
      SUBROUTINE GRAD1(I)
C
C     THIS ROUTINE DEFINES THE GRADIENT OF THE OBJECTIVE FUNCTION AND
C     CONSTRAINTS
C
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
C
      IF (I.GT.0) GO TO 50
C
C     * THE GRADIENT OF THE OBJECTIVE FUNCTION
        DEL(1) = 2.*X(1) - X(2)
        DEL(2) = 3.*X(2) - X(1)
        RETURN
C
C     * THE GRADIENT OF THE CONSTRAINTS
C
   50 GO TO (1,2,3),I
C
C     * THE GRADIENT OF G1(X) >= 0
    1   DEL(1) = 8.0
        DEL(2) = 2.*X(2)
        RETURN
C
C     * THE GRADIENT OF G2(X) >= 0
    2   DEL(1) = 20.*X(1)**3
        DEL(2) = 3.*X(2)**2
        RETURN
C
```

```
C        * THE GRADIENT OF H1(X) = 0
    3       DEL(1) = 2.*X(1)
            DEL(2) = 2.*X(2)
            RETURN
C
         END
```

MATRIX (J,L)

This subroutine supplies the upper triangle and diagonal elements of the MATRIX of second partial derivatives of f, g_J or h_J. The lower triangle elements of A, the array of second partial derivatives, must not be disturbed. The upper triangle and diagonal elements of A are all initialized to zero before being passed into MATRIX so only the nonzero elements of A need to be provided.

When J=0, this routine must set A(K,I) = $\partial^2 f(x)/\partial x_K \partial x_I$ for K=1,...,n; I=K,...,n.

When J=1,...,m, this routine must set A(K,I) = $\partial^2 g_J(x)/\partial x_K \partial x_I$ for K=1,...,n; I=K,...,n.

When J=m+1,...,m+ℓ, this routine must set A(K,I) = $\partial^2 h_J(x)/\partial x_K \partial x_I$ for K=1,...,n; I=K,...,n.

X and A are located in the COMMON region SHARE.

The MATRIX routine for the example problem is shown below.

```
         SUBROUTINE MATRIX (J,L)
C
C        THIS SUBROUTINE SUPPLIES THE UPPER TRIANGLE AND DIAGONAL ELEMENTS
C        OF THE MATRIX OF SECOND PARTIAL DERIVATIVES.
C        ONLY THE NONZERO ELEMENTS NEED TO BE PROVIDED.
C
         COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
C
         IF (J.GT.0) GO TO 50
C
C        ** THE SECOND PARTIALS OF THE OBJECTIVE FUNCTION
            A(1,1) = 2.
            A(1,2) = -1.
            A(2,2) = 3.
            RETURN
C
```

```
C      ** THE SECOND PARTIALS OF THE CONSTRAINTS **
   50   GO TO (1,2,3),J
C
C        * THE 2ND PARTIALS OF G1(X)
    1      A(2,2) = 2.
           RETURN
C
C        * THE 2ND PARTIALS OF G2(X)
    2      A(1,1) = 60.*X(1)**2
           A(2,2) = 6.*X(2)
           RETURN
C
C        * THE 2ND PARTIALS OF H1(X)
    3      A(1,1) = 2.
           A(2,2) = 2.
           RETURN
        END
```

4.4 INPUT TO THE COMPUTER PROGRAM

4.4.1 CRT DISPLAY OF QUESTIONS

```
                    RAC-SUMT ---  VERSION 4.1

PROBLEM NAME :

NUMBER OF VARIABLES :

NUMBER OF INEQUALITY CONSTRAINTS ( G(X) >= 0 ) :

NUMBER OF EQUALITY CONSTRAINTS ( H(X) = 0 ) :


                    ENTER THE INITIAL POINT :

X( 1) =

X( 2) =
  .
  .
  .
X( N) =


  THE DEFAULT VALUES FOR THE  PARAMETERS FOLLOW :

1)  R = 1.0
2)  C = 4.0
3)  EPSI = 0.1E-4
4)  THETA = 0.1E-2
5)  CONSTRAINT OPTION --- INCLUDE X(I) >= 0 CONSTRAINTS
6)  FINAL CONVERGENCE CRITERION :  RSIGMA < THETA
7)  SUBPROBLEM CONVERGENCE CRITERION #1: DELP < EPSI
8)  NO EXTRAPOLATION
9)  NO CHECKING FOR DERIVATIVES
10) UNCONSTRAINED MINIMIZATION TECHNIQUE : SECOND ORDER GRADIENT METHOD
    PRESS RETURN TO USE ALL DEFAULT VALUES
    ENTER OPTION NUMBER (1,2,...,10) TO CHANGE ONE OR MORE OPTIONS

ENTER OPTION NUMBER (RETURN IF FINISHED) : 1


1)  R -- PENALTY FACTOR
       ( RETURN  FOR  R = 1.0 )
           1    R COMPUTED BY FORMULA 1 (SEE USER'S GUIDE)
           2    R COMPUTED BY FORMULA 1 (SEE USER'S GUIDE)
           3    SPECIFY OWN VALUE OF R
     R OPTION CODE =

ENTER OPTION NUMBER (RETURN IF FINISHED) : 2
```

```
2)  C -- REDUCING FACTOR FOR R FROM STAGE TO STAGE
      ( RETURN FOR C = 4.0 )
    C =

ENTER OPTION NUMBER (RETURN IF FINISHED) : 3

3)  EPSI --- SUBPROBLEM STOPPING VALUE
      ( RETURN  FOR EPSI = 0.1E-4 )
    EPSI =

ENTER OPTION NUMBER (RETURN IF FINISHED) : 4

4)  THETA --- FINAL STOPPING VALUE
      ( RETURN  FOR THETA = 0.1E-2 )
    THETA =

ENTER OPTION NUMBER (RETURN IF FINISHED) : 5

5)  CONSTRAINT OPTION
        1   INCLUDE  X(I) >= 0  CONSTRAINTS
        2   DO NOT INCLUDE X(I) >= 0 CONSTRAINTS
    ENTER OPTION :

ENTER OPTION NUMBER (RETURN IF FINISHED) : 6

6)  FINAL CONVERGENCE CRITERION
        1   ABS[ F(X)/G ] - 1 < THETA
        2   RSIGMA < THETA
    FINAL CONVERGENCE CRITERION =

ENTER OPTION NUMBER (RETURN IF FINISHED) : 7

7)  SUBPROBLEM CONVERGENCE CRITERION
        1    SEE USER'S GUIDE
        2    SEE USER'S GUIDE
        3    GRADIENT OF P < EPSI
    SUBPROBLEM CONVERGENCE CRITERION =

ENTER OPTION NUMBER (RETURN IF FINISHED) : 8

8)   EXTRAPOLATION OPTION
        1   NO EXTRAPOLATION
        2   EXTRAPOLATE THROUGH LAST 2 MINIMA
        3   EXTRAPOLATE THROUGH LAST 3 MINIMA
     EXTRAPOLATION OPTION =

ENTER OPTION NUMBER (RETURN IF FINISHED) : 9

9)   KEY FOR CHECKING DERIVATIVES
        1   DO NOT CHECK DERIVATIVES
        2   SOLVE PROBLEM AFTER CHECKING FIRST DERIVATIVES
        3   CHECK FIRST DERIVATIVES BUT DO NOT SOLVE PROBLEM
        4   SOLVE PROBLEM AFTER CHECKING 1ST AND 2ND DERIVATIVES
        5   CHECK 1ST AND 2ND DERIVATIVES BUT DO NOT SOLVE PROBLEM
      KEY =
```

ENTER OPTION NUMBER (RETURN IF FINISHED) : 10

10) UNCONSTRAINED MINIMIZATION TECHNIQUE USED
    1 2ND ORDER GRADIENT METHOD
    2 SAME AS 1 WITH MODIFICATION
    3 STEEPEST DESCENT METHOD
    4 MODIFIED FLETCHER - POWELL METHOD
    METHOD =

4.4.2 USER'S GUIDE TO THE CRT DISPLAY

1) R -- PENALTY FACTOR
(RETURN FOR R = 1.0)

1. The value of r is made by finding an approximation solution $\min\{\nabla[P(x^0,r)[\nabla^2 P(x^0,r)]^{-1}\nabla P(x^0,r)]\}$ which is a good approximation only when x^0 is close to the boundary of a constraint or when $\nabla^2 f(x^0) = 0$ and when there are no equality constraints.

2. The value of r is made by finding the r that minimizes the magnitude of the gradient at x (ie. min $|\nabla P(x^0,r)|$). This can only be used if there are no equality constraints.

3. Specify own value of r. Several values of r may have to be tried to get the best solution to the problem. Possible values that may be tried are 10000, 1000, 100, 10, 1, 0.1, 0.01, 0.001.

2) C -- REDUCING FACTOR FOR R FROM STAGE TO STAGE
(RETURN FOR C = 4.0)
The parameter C (>0) is used to compute consecutive values of r; $r_{k+1} = r_k/C$. The value of C is usually chosen as 4.0 or 16.0.

3) EPSI --- SUBPROBLEM STOPPING VALUE
(RETURN FOR EPSI = 0.1E-4)
EPSI is the tolerance used to decide when the subproblem minimum has been reached. (see 7. SUBPROBLEM CONVERGENCE CRITERION).

4) THETA --- FINAL STOPPING VALUE
(RETURN FOR THETA = 0.1E-2)
THETA is the tolerance used to decide if the solution to the problem has been reached. Suggested values of THETA are 0.01, 0.001, 0.0001, 0.00001.

5) CONSTRAINT OPTION

1 INCLUDE X(I) >= 0 CONSTRAINTS

2 DO NOT INCLUDE X(I) >= 0 CONSTRAINTS

ENTER OPTION :

This option is set equal to 1 if the non-negativity constraints are to be included in the problem; otherwise, the option is set to 2.

6) FINAL CONVERGENCE CRITERION

1 Quit when $\left| \frac{G - F(x)}{G} \right| < \theta$

where G is the dual value. This criterion says quit when the relative difference between the dual value and function value is less than a specified tolerance (THETA).

2 Quit when $\left| r \sum_{j=1}^{m} \ln g_j(x) \right| < \theta$

This criterion says quit when the penalty term for inequality constraints is less than a tolerance .

The final convergence criterion is used to determine when the problem has been solved.

7) SUBPROBLEM CONVERGENCE CRITERION

1 Quit when $\left| \nabla_x P^t(x^i,r) \left[\frac{\partial^2 P(x,r)}{\partial x_i \ \partial x_j} \right]^{-1} \nabla_x P(x^i,r) \right| < \varepsilon$

2 Quit when $\left| \nabla_x P^t(x^i,r) \left[\frac{\partial^2 P(x,r)}{\partial x_i \ \partial x_j} \right]^{-1} \nabla_x P(x^i,r) \right| < \frac{P(x^{i-1}) - P(x^i)}{5}$

3 Quit when $\left| \nabla_x P(x^i,r) \right| < \varepsilon$

8) EXTRAPOLATION OPTION

1 NO EXTRAPOLATION

2 EXTRAPOLATE THROUGH THE LAST 2 SUBPROBLEM MINIMA

3 EXTRAPOLATE THROUGH THE LAST 3 SUBPROBLEM MINIMA

(Normally set to 1)

If option 2 or 3 are used, the program will use the previous two or three subproblem points to extrapolate to the final solution. The new point will then be used as a starting point for the next subproblem search. Options 2 or 3 are used to try to speed up convergence to the optimum point.

9) KEY FOR CHECKING DERIVATIVES

1 DO NOT CHECK DERIVATIVES.

2 SOLVE PROBLEM AFTER CHECKING FIRST DERIVATIVES.

3 CHECK FIRST DERIVATIVES BUT DO NOT SOLVE PROBLEM.

4 SOLVE PROBLEM AFTER CHECKING 1ST AND 2ND DERVIATIVES.

5 CHECK 1ST AND 2ND DERIVATIVES BUT DO NOT SOLVE PROBLEM.

Options 2 - 5 may be used if the problem has complex derivatives. The checking consists of printing out the values of the user-defined analytic derivatives and the numeric derivatives (computed by numeric differencing). If the two values are not similar in magnitude, then an error may be suspected in the user defined derivatives.

10) UNCONSTRAINED MINIMIZATION TECHNIQUE USED

1 A second order gradient method is used to minimize the unconstrained P-function. This method requires first and second derivatives of the objective function and constraints.

2 Same as 1, except that when an "orthogonal move" is made because of an indefinite Hessian matrix, $-\nabla P$ is added to the orthogonal move vector.

3 The steepest descent method, a first order gradient method, is used to minimized the P-function. Only first derivatives are required.

4 McCormick's modification of the Fletcher-Powell method is used to minimize the P-function. This method needs first derivatives.

4.5.1 TEST PROBLEMS

4.5.1 TEST PROBLEM 1 : NUMERIC EXAMPLE BY PAVIANI

4.5.1.1 SUMMARY

No. of variables : 3

No. of constraints : 1 nonlinear equality constraint

1 linear equality constraint

3 bounds on independent variables

Objective function :

$$\text{Minimize } f(x) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$$

Constraints :

$$h_1(x) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$

$$h_2(x) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

$$x_i \geq 0 \ , \quad i=1,2,3$$

Starting point : $x_i=2$, $i=1,2,3$

Parameters : $r = 1.0$, $C = 4.0$

$\text{EPSI} = 10^{-2}$, $\text{THETA} = 10^{-5}$

Unconstrained minimization technique used : modified Fletcher-Powell method

Results : $f(x) = 961.74$

$x_1 = 3.368$

$x_2 = 0.231$

$x_3 = 3.689$

$h_1(x) = 0.0006$

$h_2(x) = 0.0002$

No. of function evaluations : 38

Execution time : 1.2 min.

4.5.1.2 COMPUTER PRINTOUT OF RESULTS

```
                RAC-SUMT --- VERSION 4.1


          TEST PROBLEM 1

N =  3    M =  0    MZ =  2

OPTIONS SELECTED
1)  R =  .1000E+01     (USER SPECIFIED)
2)  C =  .4000E+01
3)  EPSI =  .1000E-01
4)  THETA =  .1000E-04
5)  CONSTRAINT OPTION --- INCLUDE X(I) >= 0 CONSTRAINTS
6)  FINAL CONVERGENCE CRITERION --- ABS[ F(X)/G ] - 1 < THETA
7)  SUBPROBLEM CONVERGENCE CRITERION #1
8)  EXTRAPOLATE THROUGH LAST 2 MINIMA
9)  SOLVE PROBLEM AFTER CHECKING 1ST AND 2ND DERIVATIIVES
10) UNCONSTRAINED MINIMIZATION TECHNIQUE -- MODIFIED FLETCHER - POWELL METHOD

F =  .9760000E+03    P =  .0000000E+01    G =  .0000000E+01

VALUES OF X VECTOR
 X( 1) =  .2000000E+01  X( 2) =  .2000000E+01  X( 3) =  .2000000E+01

VALUES OF THE CONSTRAINTS
  G( 1) = -.1300000E+02   G( 2) =  .2000000E+01   G(


VALUES OF OBJECTIVE FUNCTION PARTIALS

ANALYTICAL FIRST PARTIALS
 DEL( 1) = -.8000000E+01  DEL( 2) = -.1000000E+02  DEL( 3) = -.6000000E+01

NUMERICAL FIRST PARTIALS
 DEL( 1) = -.7934570E+01  DEL( 2) = -.9765625E+01  DEL( 3) = -.6103516E+01

VALUES OF CONSTRAINT NUMBER   1

ANALYTICAL FIRST PARTIALS
 DEL( 1) =  .4000000E+01  DEL( 2) =  .4000000E+01  DEL( 3) =  .4000000E+01

NUMERICAL FIRST PARTIALS
 DEL( 1) =  .3995895E+01  DEL( 2) =  .3995895E+01  DEL( 3) =  .3995895E+01

VALUES OF CONSTRAINT NUMBER   2

ANALYTICAL FIRST PARTIALS
 DEL( 1) =  .8000000E+01  DEL( 2) =  .1400000E+02  DEL( 3) =  .7000000E+01

NUMERICAL FIRST PARTIALS
 DEL( 1) =  .8010864E+01  DEL( 2) =  .1399994E+02  DEL( 3) =  .6980896E+01
```

```
VALUES OF OBJECTIVE FUNCTION PARTIALS

ANALYTICAL SECOND PARTIALS
 A( 1, 1) = -.200000E+01  A( 1, 2) = -.100000E+01  A( 1, 3) = -.100000E+01
 A( 2, 1) =  .000000E+01  A( 2, 2) = -.400000E+01  A( 2, 3) =  .000000E+01
 A( 3, 1) =  .000000E+01  A( 3, 2) =  .000000E+01  A( 3, 3) = -.200000E+01

NUMERICAL SECOND PARTIALS
 A( 1, 1) = -.200033E+01  A( 1, 2) = -.100136E+01  A( 1, 3) = -.100136E+01
 A( 2, 1) =  .000000E+01  A( 2, 2) = -.399590E+01  A( 2, 3) =  .000000E+01
 A( 3, 1) =  .000000E+01  A( 3, 2) =  .000000E+01  A( 3, 3) = -.199795E+01

VALUES OF CONSTRAINT NUMBER   1

ANALYTICAL SECOND PARTIALS
 A( 1, 1) =  .200000E+01  A( 1, 2) =  .000000E+01  A( 1, 3) =  .000000E+01
 A( 2, 1) =  .000000E+01  A( 2, 2) =  .200000E+01  A( 2, 3) =  .000000E+01
 A( 3, 1) =  .000000E+01  A( 3, 2) =  .000000E+01  A( 3, 3) =  .200000E+01

NUMERICAL SECOND PARTIALS
 A( 1, 1) =  .199914E+01  A( 1, 2) =  .000000E+01  A( 1, 3) =  .000000E+01
 A( 2, 1) =  .000000E+01  A( 2, 2) =  .199914E+01  A( 2, 3) =  .000000E+01
 A( 3, 1) =  .000000E+01  A( 3, 2) =  .000000E+01  A( 3, 3) =  .199914E+01

VALUES OF CONSTRAINT NUMBER   2

ANALYTICAL SECOND PARTIALS
 A( 1, 1) =  .000000E+01  A( 1, 2) =  .000000E+01  A( 1, 3) =  .000000E+01
 A( 2, 1) =  .000000E+01  A( 2, 2) =  .000000E+01  A( 2, 3) =  .000000E+01
 A( 3, 1) =  .000000E+01  A( 3, 2) =  .000000E+01  A( 3, 3) =  .000000E+01

NUMERICAL SECOND PARTIALS



       ***  POINT NUMBER    8    ***

RHO =   .1000000E+01    RSIGMA =  -.1010660E+01

F =  .9610892E+03    P =  .9603866E+03    G =  .9587054E+03

VALUES OF X VECTOR
 X( 1) =  .3395841E+01  X( 2) =  .2170724E+00  X( 3) =  .3727081E+01

VALUES OF THE CONSTRAINTS
  G( 1) =  .4699898E+00   G( 2) =  .2953072E+00   G(



       ***  POINT NUMBER   14    ***

RHO =   .2500000E+00    RSIGMA =  -.2567300E+00

F =  .9615558E+03    P =  .9613916E+03    G =  .9609908E+03
```

```
VALUES OF X VECTOR
 X( 1) =  .3374081E+01  X( 2) =  .2235025E+00  X( 3) =  .3702933E+01

VALUES OF THE CONSTRAINTS
  G( 1) =  .1460915E+00  G( 2) =  .4221725E-01   G(


      *** POINT NUMBER   16   ***

RHO =   .6250000E-01    RSIGMA =  -.6339629E-01

F =  .9615630E+03    P =  .9618439E+03    G =  .9620641E+03

VALUES OF X VECTOR
 X( 1) =  .3377104E+01  X( 2) =  .2206620E+00  X( 3) =  .3700392E+01

VALUES OF THE CONSTRAINTS
  G( 1) =  .1464233E+00  G( 2) =  .8842468E-02   G(


      *** POINT NUMBER   25   ***

RHO =   .1562500E-01    RSIGMA =  -.1645939E-01

F =  .9617327E+03    P =  .9617232E+03    G =  .9616998E+03

VALUES OF X VECTOR
 X( 1) =  .3367842E+01  X( 2) =  .2307426E+00  X( 3) =  .3689812E+01

VALUES OF THE CONSTRAINTS
  G( 1) =  .1031494E-01  G( 2) =  .1815796E-02   G(


      *** POINT NUMBER   27   ***

RHO =   .3906250E-02    RSIGMA =  -.4113468E-02

F =  .9617391E+03    P =  .9617462E+03    G =  .9617496E+03

VALUES OF X VECTOR
 X( 1) =  .3367891E+01  X( 2) =  .2306978E+00  X( 3) =  .3689177E+01

VALUES OF THE CONSTRAINTS
  G( 1) =  .5933762E-02  G( 2) = -.2868652E-02   G(


      *** POINT NUMBER   38   ***

RHO =   .9765625E-03    RSIGMA =  -.1030822E-02
```

```
F =  .9617449E+03    P =  .9617443E+03    G =  .9617427E+03

VALUES OF X VECTOR
 X( 1) =  .3367628E+01  X( 2) =  .2313299E+00  X( 3) =  .3688648E+01

VALUES OF THE CONSTRAINTS
  G( 1) =  .5550385E-03   G( 2) =  .1792908E-03   G(
```

*

```
FINAL VALUE OF F =    9.617449E+02


FINAL X VALUES
 X( 1) =  3.367628E+00  X( 2) =  2.313299E-01  X( 3) =  3.688648E+00
```

4.5.1.3 USER SUPPLIED SUBROUTINES

```
      SUBROUTINE RESTNT (I,VAL)
C
C    ** TEST PROBLEM 1 - PAVIANI **
C
      COMMON /SHARE/ X(20),DEL(20), A(20,20), N,M,MN,NP1,NM1
C
      IF (I.GT.0) GO TO 10
C
         VAL = 1000.0 - X(1)**2 - 2.0*X(2)**2 - X(3)**2 - X(1)*X(2)
     1             - X(1)*X(3)
         RETURN
C
   10 GO TO (1,2), I
C
    1    VAL = X(1)**2 + X(2)**2 + X(3)**2 - 25.0
         RETURN
C
    2    VAL = 8.0*X(1) + 14.0*X(2) + 7.0*X(3) - 56.0
         RETURN
C
      END
C
C
      SUBROUTINE GRAD1 (I)
C
      COMMON /SHARE/ X(20),DEL(20), A(20,20), N,M,MN,NP1,NM1
C
      IF (I.GT.0) GO TO 10
C
```

```
          DEL(1) = - 2.0*X(1) - X(2) - X(3)
          DEL(2) = - 4.0*X(2) - X(1)
          DEL(3) = - 2.0*X(3) - X(1)
          RETURN
C
   10   GO TO (1,2), I
C
    1     DEL(1) = 2.0 * X(1)
          DEL(2) = 2.0 * X(2)
          DEL(3) = 2.0 * X(3)
          RETURN
C
    2     DEL(1) =  8.0
          DEL(2) = 14.0
          DEL(3) =  7.0
          RETURN
C
        END
C
C
        SUBROUTINE MATRIX (J,L)
C
        COMMON /SHARE/ X(20),DEL(20), A(20,20), N,M,MN,NP1,NM1
C
        IF (J.GT.0) GO TO 10
C
          A(1,1) = -2.0
          A(1,2) = -1.0
          A(1,3) = -1.0
C
          A(2,2) = -4.0
          A(2,3) =  0.0
C
          A(3,3) = -2.0
          RETURN
C
   10   GO TO (1,2), J
C
    1     A(1,1) = 2.0
          A(2,2) = 2.0
          A(3,3) = 2.0
    2     RETURN
C
        END
```

4.5.2 TEST PROBLEM 2 : PROBLEM OF MAXIMIZING SYSTEM RELIABLITY

4.5.2.1 SUMMARY

No. of variables : 4

No. of constraints : 9

Objective function :

$$\text{Minimize} \quad f(x) = -1 + R_3[(1-R_1)(1-R_4)]^2 + (1-R_3)\{1 - R_2[1-(1-R_1)(1-R_4)]\}^2$$

Constraints :

$$g_1(x) = C - (2K_1R_1^{\alpha_1} + 2K_2R_2^{\alpha_2} + K_3R_3^{\alpha_3} + 2K_4R_4^{\alpha_4}) \geq 0$$

$$g_{i+1}(x) = 1 - R_i \geq 0, \quad i=1,2,3,4$$

$$g_{i+5}(x) = R_i - R_{i,min} \geq 0, \quad i=1,2,3,4$$

where $K_1=100$, $K_2=100$, $K_3=200$, $K_4=150$

$C=800$

$\alpha_i=0.6$, $R_{i,min}=0.5$, $i=1,2,3,4$

Starting point : $R_i=0.6$, $i=1,2,3,4$

Parameters : $r= .03578$, $C=4.0$

$EPSI=10^{-5}$, $THETA=10^{-5}$

Unconstrained minimization technique used : Steepest descent method

Results : $f(x) = 0.9999985$

$R_1 = 0.9970$

$R_2 = 0.9996$

$R_3 = 0.6622$

$R_4 = 0.6368$

No. of function evaluations : 38

Execution time : 3.0 min.

4.5.2.2 COMPUTER PRINTOUT OF RESULTS

```
                    RAC-SUMT --- VERSION 4.1

             TEST PROBLEM 2

N =  4     M =  9    MZ =  0

OPTIONS SELECTED
1)  R TO BE COMPUTED BY FORMULA 2
2)  C =  .4000E+01
3)  EPSI =  .1000E-04
4)  THETA =  .1000E-04
5)  CONSTRAINT OPTION --- DO NOT INCLUDE X(I) >= 0 CONSTRAINTS
6)  FINAL CONVERGENCE CRITERION --- RSIGMA < THETA
7)  SUBPROBLEM CONVERGENCE CRITERION #1
8)  NO EXTRAPOLATION
9)  NO CHECKING FOR DERIVATIVES
10) UNCONSTRAINED MINIMIZATION TECHNIQUE -- STEEPEST DESCENT METHOD

F = -.8862336E+00    P =  .0000000E+01    G =  .0000000E+01

VALUES OF X VECTOR
 X( 1) =  .6000000E+00  X( 2) =  .6000000E+00  X( 3) =  .6000000E+00
 X( 4) =  .6000000E+00  X(

VALUES OF THE CONSTRAINTS
  G( 1) =  .1375800E+03   G( 2) =  .4000000E+00   G( 3) =  .4000000E+00
  G( 4) =  .4000000E+00   G( 5) =  .4000000E+00   G( 6) =  .1000000E+00
  G( 7) =  .1000000E+00   G( 8) =  .1000000E+00   G( 9) =  .1000000E+00


      ***  POINT NUMBER    6    ***

RHO =   .3577597E-01    RSIGMA =   .2573350E+00

F = -.9748093E+00    P = -.7174743E+00    G = -.1296793E+01

VALUES OF X VECTOR
 X( 1) =  .7356728E+00  X( 2) =  .7904098E+00  X( 3) =  .7320088E+00
 X( 4) =  .6883459E+00  X(

VALUES OF THE CONSTRAINTS
  G( 1) =  .5433093E+02   G( 2) =  .2643272E+00   G( 3) =  .2095902E+00
  G( 4) =  .2679912E+00   G( 5) =  .3116541E+00   G( 6) =  .2356728E+00
  G( 7) =  .2904098E+00   G( 8) =  .2320088E+00   G( 9) =  .1883459E+00


      ***  POINT NUMBER   16    ***

RHO =   .8943993E-02    RSIGMA =   .7195718E-01
```

F = -.9896287E+00   P = -.9176715E+00   G = -.1070125E+01

VALUES OF X VECTOR
X(1) = .8135905E+00   X(2) = .8868126E+00   X(3) = .7150513E+00
X(4) = .6810546E+00   X(

VALUES OF THE CONSTRAINTS
G(1) = .3539966E+02   G(2) = .1864095E+00   G(3) = .1131874E+00
G(4) = .2849487E+00   G(5) = .3189454E+00   G(6) = .3135905E+00
G(7) = .3868126E+00   G(8) = .2150513E+00   G(9) = .1810546E+00

*** POINT NUMBER 22 ***

RHO = .2235998E-02   RSIGMA = .2150956E-01

F = -.9973105E+00   P = -.9758009E+00   G = -.1017434E+01

VALUES OF X VECTOR
X(1) = .9130118E+00   X(2) = .9494833E+00   X(3) = .6719643E+00
X(4) = .6503463E+00   X(

VALUES OF THE CONSTRAINTS
G(1) = .2745135E+02   G(2) = .8698821E-01   G(3) = .5051672E-01
G(4) = .3280357E+00   G(5) = .3496537E+00   G(6) = .4130118E+00
G(7) = .4494833E+00   G(8) = .1719643E+00   G(9) = .1503463E+00

*** POINT NUMBER 28 ***

RHO = .5589995E-03   RSIGMA = .6239673E-02

F = -.9993114E+00   P = -.9930718E+00   G = -.1004342E+01

VALUES OF X VECTOR
X(1) = .9586948E+00   X(2) = .9743827E+00   X(3) = .6640598E+00
X(4) = .6392964E+00   X(

VALUES OF THE CONSTRAINTS
G(1) = .2227216E+02   G(2) = .4130524E-01   G(3) = .2561730E-01
G(4) = .3359402E+00   G(5) = .3607036E+00   G(6) = .4586948E+00
G(7) = .4743827E+00   G(8) = .1640598E+00   G(9) = .1392964E+00

*** POINT NUMBER 32 ***

RHO = .1397499E-03   RSIGMA = .1788709E-02

F = -.9998465E+00   P = -.9980577E+00   G = -.1001104E+01

VALUES OF X VECTOR
X(1) = .9815737E+00 X(2) = .9874529E+00 X(3) = .6623129E+00
X(4) = .6368518E+00 X(

VALUES OF THE CONSTRAINTS
G(1) = .1868634E+02 G(2) = .1842630E-01 G(3) = .1254714E-01
G(4) = .3376871E+00 G(5) = .3631482E+00 G(6) = .4815737E+00
G(7) = .4874529E+00 G(8) = .1623129E+00 G(9) = .1368518E+00

*** POINT NUMBER 34 ***

RHO = .3493747E-04 RSIGMA = .4942107E-03

F = -.9999571E+00 P = -.9994630E+00 G = -.1000272E+01

VALUES OF X VECTOR
X(1) = .9896193E+00 X(2) = .9937997E+00 X(3) = .6622716E+00
X(4) = .6368153E+00 X(

VALUES OF THE CONSTRAINTS
G(1) = .1696405E+02 G(2) = .1038069E-01 G(3) = .6200314E-02
G(4) = .3377284E+00 G(5) = .3631847E+00 G(6) = .4896193E+00
G(7) = .4937997E+00 G(8) = .1622716E+00 G(9) = .1368153E+00

*** POINT NUMBER 36 ***

RHO = .8734368E-05 RSIGMA = .1390110E-03

F = -.9999923E+00 P = -.9998533E+00 G = -.1000071E+01

VALUES OF X VECTOR
X(1) = .9953239E+00 X(2) = .9975349E+00 X(3) = .6622406E+00
X(4) = .6368152E+00 X(

VALUES OF THE CONSTRAINTS
G(1) = .1583319E+02 G(2) = .4676104E-02 G(3) = .2465069E-02
G(4) = .3377594E+00 G(5) = .3631848E+00 G(6) = .4953239E+00
G(7) = .4975349E+00 G(8) = .1622406E+00 G(9) = .1368152E+00

*** POINT NUMBER 37 ***

RHO = .2183592E-05 RSIGMA = .3681667E-04

F = -.9999965E+00 P = -.9999597E+00 G = -.1000016E+01

VALUES OF X VECTOR
X(1) = .9962229E+00 X(2) = .9987994E+00 X(3) = .6622418E+00
X(4) = .6368178E+00 X(

```
VALUES OF THE CONSTRAINTS
  G( 1) =  .1557214E+02   G( 2) =  .3777087E-02   G( 3) =  .1200557E-02
  G( 4) =  .3377582E+00   G( 5) =  .3631822E+00   G( 6) =  .4962229E+00
  G( 7) =  .4987994E+00   G( 8) =  .1622418E+00   G( 9) =  .1368178E+00


      ***  POINT NUMBER   38   ***

RHO =    .5458980E-06    RSIGMA =    .9961238E-05

F = -.9999985E+00    P = -.9999885E+00    G = -.1000003E+01

VALUES OF X VECTOR
 X( 1) =  .9969606E+00  X( 2) =  .9996238E+00  X( 3) =  .6622428E+00
 X( 4) =  .6368231E+00  X(

VALUES OF THE CONSTRAINTS
  G( 1) =  .1538367E+02   G( 2) =  .3039360E-02   G( 3) =  .3761649E-03
  G( 4) =  .3377572E+00   G( 5) =  .3631769E+00   G( 6) =  .4969606E+00
  G( 7) =  .4996238E+00   G( 8) =  .1622428E+00   G( 9) =  .1368231E+00

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

FINAL VALUE OF F =   -9.999985E-01

FINAL X VALUES
 X( 1) =  9.969606E-01  X( 2) =  9.996238E-01  X( 3) =  6.622428E-01
 X( 4) =  6.368231E-01  X(
```

4.5.2.3 USER SUPPLIED SUBROUTINES

```
      SUBROUTINE RESTNT (I,VAL)
C
C     THE RELIABILITY PROBLEM
C
      REAL  R1, R2, R3, R4,  Q1, Q2, Q3, Q4, PART2
      REAL  C,  K1, K2, K3, K4,  A1, A2, A3, A4,  RMIN
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /CONST/ C, K1, K2, K3, K4, A1, A2, A3, A4, RMIN
      DATA  C, K1, K2, K3, K4 /800.0,  100.0, 100.0, 200.0, 150.0/
      DATA  A1, A2, A3, A4, RMIN / .60, .60, .60, .60,  .50/
C
      R1 = X(1)
      R2 = X(2)
      R3 = X(3)
      R4 = X(4)
      Q1 = 1.0 - R1
      Q2 = 1.0 - R2
      Q3 = 1.0 - R3
      Q4 = 1.0 - R4
      PART2 =  1.0 - R2*( 1.0 - Q1*Q4 )
C
      IF (I.GT.0) GO TO 100
C
C    * THE OBJECTIVE FUNCTION TO BE MINIMIZED
      VAL =  - 1.0 + R3*(Q1*Q4)**2 + Q3*PART2**2
      RETURN
C
C    * THE INEQUALITY CONSTRAINTS ( G(I) >= 0 )
  100 GO TO (1,2,3,4,5,6,7,8,9), I
C
    1 COST =  2*K1*R1**A1 + 2*K2*R2**A2 + K3*R3**A3 + 2*K4*R4**A4
      VAL = C - COST
      RETURN
C
    2 VAL = 1.0 - R1
      RETURN
    3 VAL = 1.0 - R2
      RETURN
    4 VAL = 1.0 - R3
      RETURN
    5 VAL = 1.0 - R4
      RETURN
    6 VAL = R1 - RMIN
      RETURN
    7 VAL = R2 - RMIN
      RETURN
    8 VAL = R3 - RMIN
      RETURN
    9 VAL = R4 - RMIN
      RETURN
C
      END
```

```
      SUBROUTINE GRAD1(I)
C
      REAL  R1, R2, R3, R4,  Q1, Q2, Q3, Q4, PART2
      REAL  C,  K1, K2, K3, K4,  A1, A2, A3, A4,  RMIN
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /CONST/ C, K1, K2, K3, K4, A1, A2, A3, A4, RMIN
C
      R1 = X(1)
      R2 = X(2)
      R3 = X(3)
      R4 = X(4)
      Q1 = 1.0 - R1
      Q2 = 1.0 - R2
      Q3 = 1.0 - R3
      Q4 = 1.0 - R4
      PART2 =  1.0 - R2*( 1.0 - Q1*Q4 )
C    * SET DEL TO ZERO BEFORE FILLING IN THE NONZERO ELEMENTS
      DO 50 INDEX = 1,4
         DEL(INDEX) = 0.0
   50 CONTINUE
C
      IF (I.GT.0) GO TO 100
C
C    * THE GRADIENT OF THE OBJECTIVE FUNCTION
      DEL(1) = 2.0 *R3*Q1*Q4*(-Q4)  +  2.0 *Q3*PART2*(-R2)*Q4
      DEL(2) =  -2.0 *Q3*PART2*( 1.0 - Q1*Q4 )
      DEL(3) =  (Q1*Q4)**2 - PART2**2
      DEL(4) = 2.0 *R3*Q1*Q4*(-Q1)  +  2.0 *Q3*PART2*(-R2)*Q1
      RETURN
C
C    * THE GRADIENT OF THE CONSTRAINTS
  100 GO TO (1,2,3,4,5,6,7,8,9), I
C
    1 DEL(1) =   - 2.0*K1*A1 * R1**(A1-1)
      DEL(2) =   - 2.0*K2*A2 * R2**(A2-1)
      DEL(3) =   - K3*A3 * R3**(A3-1)
      DEL(4) =   - 2.0*K4*A4 * R4**(A4-1)
      RETURN
    2 DEL(1) = -1.0
      RETURN
    3 DEL(2) = -1.0
      RETURN
    4 DEL(3) = -1.0
      RETURN
    5 DEL(4) = -1.0
      RETURN
    6 DEL(1) = 1.0
      RETURN
    7 DEL(2) = 1.0
      RETURN
    8 DEL(3) = 1.0
      RETURN
    9 DEL(4) = 1.0
      RETURN
      END
```

```
      SUBROUTINE MATRIX (J,L)
C
      REAL  R1, R2, R3, R4,  Q1, Q2, Q3, Q4, PART2
      REAL  C,  K1, K2, K3, K4,  A1, A2, A3, A4,  RMIN
      COMMON /SHARE/ X(20), DEL(20), A(20,20), N,M,MN,NP1,NM1
      COMMON /CONST/ C, K1, K2, K3, K4, A1, A2, A3, A4, RMIN
C
      R1 = X(1)
      R2 = X(2)
      R3 = X(3)
      R4 = X(4)
      Q1 = 1.0 - R1
      Q2 = 1.0 - R2
      Q3 = 1.0 - R3
      Q4 = 1.0 - R4
      PART2 =  1.0 - R2*( 1.0 - Q1*Q4 )
C
      IF (J.GT.0) GO TO 100
C
C   * THE SECOND PARTIALS OF THE OBJECTIVE FUNCTION
      A(1,1) =  2*R3*Q4*Q4 + 2*Q3*(R2**2)*(Q4**2)
      A(1,2) =  - 2*Q3*Q4*PART2 + 2*Q3*R2*Q4*(1.0 - Q1*Q4 )
      A(1,3) =  - 2*Q1*(Q4**2) + 2*R2*Q4*PART2
      A(1,4) =  2*R3*Q1*Q4 + 2*R3*Q1*Q4
     1         + 2*Q3*(R2**2) *Q4*Q1  +  2*Q3*R2*PART2
C
      A(2,2) =  2*Q3*(1.0 - Q1*Q4)**2
      A(2,3) =  2*PART2*(1.0 - Q1*Q4)
      A(2,4) =  2*Q3*(1.0 - Q1*Q4)*R2*Q1 + 2*Q3*PART2*Q1
      A(3,4) =  - 2*Q1*Q4*Q1 + 2*PART2*R2*Q1
      A(4,4) =  2*R3*(Q1**2) + 2*Q3*(R2**2)*(Q1**2)
      RETURN
C
C   * THE SECOND PARTIALS OF THE CONSTRAINTS
  100 GO TO (1,2,2,2,2,2,2,2,2), J
C
    1 A(1,1) =  - 2.0*K1*A1*(A1-1) * R1**(A1-2)
      A(2,2) =  - 2.0*K2*A2*(A2-1) * R2**(A2-2)
      A(3,3) =  - K3*A3*(A3-1) * R3**(A3-2)
      A(4,4) =  - 2.0*K4*A4*(A4-1) * R4**(A4-2)
    2 RETURN
C
      END
```

4.6 REFERENCES

1. Fiacco, A. V., and G. P. McCormick, "The Sequential Unconstrained Minimization Technique for Nonlinear Programming : A Primal-Dual Method", Management Sci., 10, 360-366, 1964.

2. Fiacco, A. V., and G. P. McCormick, "Computational Algorithm for the Sequential Unconstrained Minimizatin Technique for Nonlinear Programming", Management Sci., 10, 601-617, 1964.

3. Fiacco, A. V., and G. P. McCormick, "SUMT without parameters", Systems Research Memorandum No. 121, Technical Institute, Northwestern University, Evanston, Illinois, 1965.

4. Fiacco, A. V., and G. P. McCormick, "Extension of SUMT for Nonlinear Programming : Equality Constraints and Extrapolation", Management Sci., 12 (11) : 816-829, 1966.

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

6. Kuester, J. L. and J. H. Mize, Optimization Techniques with Fortran, McGraw-Hill Book Company, 1973.

5.1 CRITERIA USED IN COMPARING THE MICRO/PERSONAL COMPUTER VERSUS THE LARGE COMPUTER

Many of the criteria used in evaluating competing techniques [1] on the same computer can also be used in evaluating the micro/personal computer against the large computer. The criteria which are used in this study are:

1. Time required in a series of tests
   (Preparation time, queue time, and execution time)
2. Size of the problem
   (number of variables, number of inequality constraints, number of equality constraints)
3. Accuracy of the solution with respect to the optimal vector x^* and /or with respect to $f(x^*)$, $h(x^*)$, $g(x^*)$.
4. Simplicity of use

1. Time required in a series of tests

The total time required to solve a problem on the large computer includes preparation time, the queue time which is the time which has to be spent waiting in a queue for either a terminal or for other people's jobs to finish executing, and execution time. Of these times, the queue time can take up a significantly large proportion of the overall time needed to solve a problem. This is because each time the program has to be run, there is some queue time involved and because the program usually does not run the first time because of errors, there will be an accumulation of queue times. However, when using a micro/personal computer there is no queue time so often the same problem can be solved faster on a micro/personal computer than on the large computer.

2. Size of the problem

The size of the problem which can be solved on each of the programs is shown below :

The Hooke and Jeeves pattern search

Large : 50 variables

Micro : 50 variables

KSU-SUMT

Large : 20 variables

20 inequality constraints

20 equality constraints

Micro : 20 variables

20 inequality constraints

20 equality constraints

RAC-SUMT

Large : 20 variables

20 inequality constraints

20 equality constraints

Micro : 20 variables

20 inequality constraints

20 equality constraints

On each of the three programs, the dimensions of the micro computer was set equal to the dimensions of the programs written for the large computer. However, forthe RAC-SUMT program, although the main program fits into the 37K bytes of usable computer memory of the North Star computer, the user supplied subroutines may not fit into the memory. This is because the main program uses 28K bytes of memory which leaves only 9K bytes for the user supplied subroutines. In the RAC-SUMT program, three user supplied subroutines are required : RESTNT, GRAD, MATRIX. The RESTNT subroutine which supplies the objective function and the constraints may not be very large but the GRAD subroutine and the MATRIX subroutine which supply the first and second partial derivatives of the objective function and constraints can get quite large. Therefore the user supplied subroutines can easily exceed the 9K bytes.

3. Accuracy of the solution

The results of the test problems run on the large computer and the microcomputer are shown below :

The Hocke and Jeeves pattern search

Test problem 1 :

Large : $f(x^*) = 2960.74$

Micro : $f(x^*) = 2960.74$

Test problem 2 :

Large : $f(x^*) = 241,516$

Micro : $f(x^*) = 241,516$

KSU-SUMT

Test problem 1

Large : $f(x^*) = 962.50$

$g_1(x^*) = 2.73$

$g_2(x^*) = .352$

$g_3(x^*) = 4.17$

$h_1(x^*) = .01$

$h_2(x^*) = .005$

Micro : $f(x^*) = 962.34$

$g_1(x^*) = 2.79$

$g_2(x^*) = .335$

$g_3(x^*) = 4.14$

$h_1(x^*) = .06$

$h_2(x^*) = .01$

Test problem 2

Large : $f(x^*) = .9946$

$g_1(x^*) = .0454$

$g_2(x^*) = .1778$

$g_3(x^*) = .1203$

$g_4(x^*) = .1775$

$g_5(x^*) = .2170$

$g_6(x^*) = .3222$

$g_7(x^*) = .3797$

$g_8(x^*) = .3225$

$g_9(x^*) = .2830$

Micro : $f(x^*) = .9955$

$g_1(x^*) = .201$

$g_2(x^*) = .207$

$g_3(x^*) = .828$

$g_4(x^*) = .193$

$g_5(x^*) = .212$

$g_6(x^*) = .293$

$g_7(x^*) = .417$

$g_8(x^*) = .307$

$g_9(x^*) = .288$

RAC-SUMT

Test problem 2

Large : $f(x^*) = .999994$

$g_1(x^*) = .9067$

$g_2(x^*) = .0036$

$g_3(x^*) = .0042$

$g_4(x^*) = .1206$

$g_5(x^*) = .4267$

$g_6(x^*) = .4964$

$g_7(x^*) = .4958$

$g_8(x^*) = .3794$

$g_9(x^*) = .0733$

Micro : $f(x^*) = .999998$

$g_1(x^*) = 15.38$

$g_2(x^*) = .00304$

$g_3(x^*) = .00376$

$g_4(x^*) = .3378$

$g_5(x^*) = .3632$

$g_6(x^*) = .4970$

$g_7(x^*) = .4996$

$g_8(x^*) = .1622$

$g_9(x^*) = .1368$

The above results of the problem run on the micro/personal computer and the large computer are essentially the same. In the Hooke and Jeeves pattern search problems, the objective function values were identical when run on the micro/personal computer and the large computer. The objective function for the test problems run by the KSU-SUMT and RAC-SUMT were nearly identical for the micro/personal computer as compared to the large computer. The results for RAC-SUMT test problem 1 was not shown because the version of RAC-SUMT on the large computer could not handle equality constraints. Note that in nonlinear programming problems the objective function may not be unimodal, so that there may be several points which give the same value of the objective function. This is probably why there are differences in the values of the constraints for the KSU-SUMT and RAC-SUMT test problems although the objective functions are nearly identical.

An exact comparison of the results from the micro/personal computer and the large computer is also not valid because the programs stored on the micro/personal computer and the ones stored in the large computer are not identical. The programs stored in the large computer are an older version although for the Hooke and Jeeves pattern search and the KSU-SUMT program, they are essentially the same. Only in the RAC-SUMT program were any major changes made in the newer version but most of the changes were in terms of adding new features to the program while the basic method of the program remained unchanged. These results indicate that the micro/personal computer can produce solutions which are as good as those produced by the large computer.

4. Simplicity of use

For the large computer some job control language (JCL) statements are needed to run the programs whereas for the micro/personal computer a few operating systems commands are needed to invoke the Fortran compiler and the linkage editor in order to run the program. The commands needed to run the micro/personal computer are usually easier to learn and remember than the corresponding JCL neeeded to run the programs on the large computer. To illustrate the complexity of the JCL for the large computer, the JCL statements needed to run the RAC-SUMT program is shown below.

```
//  EXEC FORTGCLG
//FORT.SYSIN DD *

     the user supplied subroutines go here

//LKED.LIB DD DSN=DSBN7.HWANG.ORFILES,DISP=SHR
//LKED.SYSIN DD *
       INCLUDE LIB(RACSUMT)
       ENTRY MAIN
//GO.SYSIN DD *

     the user supplied data cards go here
/*
```

The more simple operating systems commands needed to run the RAC-SUMT program are as follow :

The following command is used to compile the user supplied subroutines.

```
F80 =B:filename
```

The following command is used to link edit the compiled user supplied subroutines with the compiled RAC-SUMT program and create a executable file.

```
L80 B:filename,B:RACSUMT/N,B:RACSUMT/E
```

The following command is used to begin execution of the RAC-SUMT program:

```
B:READIN
```

As shown above, it is much easier to remember the commands needed for the microcomputer than it is to remember or even understand the JCL statements needed for the large computer.

5.2 REASONS FOR USING THE MICRO/PERSONAL COMPUTER IN RESEARCH OR APPLICATIONS

One of the reasons for using a micro/personal computer is the easy accessibility to the micro/personal computer. There is no need to have a security number to use the micro/personal computer as there is for using the large computer. No computer funds are needed to run a program as for the large computer. There is also no restriction on the hours of use as for the large computer.

A second reason for using the micro/personal computer is the low operating cost of the micro/personal computer. The only cost for operating the micro/personal computer is the electricity cost for running the computer, the cost of paper for printing out results and the cost of mini disks for storing the programs. On the other hand, the operating cost for the large computer can be expensive as one or more operators are needed to keep the computer running, to mount tapes or disks when requested, and to dispatch computer printouts to users, among other tasks. In addition, an accountant is needed to keep track of the accounts of the various computer users. Systems programmers are also needed to maintain the system programs in good running order. All of these people are needed to keep the large computer working properly and to meet the needs of the various users of the large computer system. Their services can be quite expensive.

A third reason for using the micro/personal computer is the adequate capacity of the micro to handle the problems to be solved. Most often the complete capacity of a large computer is not needed when the problem to be solved is only moderately large. For many problems, the micro/personal computer has enough capacity to be able to handle them. For example, the Hooke and Jeeves pattern search program and the KSU-SUMT program require only 22K and 32K bytes of memory so they can easily fit into the available computer memory of a 64K microcomputer. The RAC-SUMT program requires more memory than what is available but with some modifications, it also can run on the micro/personal computer.

5.3 EXPERIENCE ON MICRO/PERSONAL COMPUTER

One of the attractive features of the micro/personal computer is the ability to make changes to the program easily and quickly. This is a feature of the word processing software that is available to create and edit programs. The word processing software locates particular statements quickly and allows additions, deletions, and replacements to be made very easily. For instance, to change a variable name throughout the program, only one command needs to be issued and all changes will be made. The word processing software used in creating the program was MicroPro's Wordstar. Having also used IBM's virtual machine system product editor (also known as XEDIT) on the large computer, my experience has been that the word processor on the microcomputer is just as sophisticated as that for the large computer.

One type of problem which was encountered when using the Fortran compiler was determining where an error occurred when an error message appeared. Although a line number indicating where the error occurred is supposed to be given, sometimes no line number was present. And when the line number is present, it often is off by one or two lines. Also, when an error occurs in a subroutine, the line number is given in reference to the start of the subroutine, whereas the word processing editor which was used numbered all lines with respect to the start of the program. There were therefore some adjustments needed to determine the location of the error in the subroutine. In addition to the line number where an error occurred, the last 20 characters scanned at the time the error was detected is given. These 20 characters are often misleading because the error is usually not in the 20 characters but a line or two before or after the statement which contained the 20 characters.

Another type of problem which was encountered when using the Fortran compiler was caused by the compiler not checking for all types of syntax errors. One of the syntax errors not checked for was incorrectly using single precision built-in functions like ABS, ALOG, and SQRT when the double precision functions DABS, DLOG, and DSQRT should have been used. Another type of error not checked for was the matching of parameters in the subroutine in number, type, and length with the parameters expected by the calling program. When these types of errors occurred, the results of calculations done by the program was often totally incorrect and many times error messages would appear during execution which were nonsensical like a message of 'Error --- Argument to COS too large' when the COS function was never used in the program.

These types of errors were some of the most difficult to debug and hopefully newer versions of the compiler will check for these additional types of errors. One of the reasons for the problems with the Fortran compiler is probably because the Fortran compiler is still in the developing stage and because it is a first version, we can expect errors to be present. Probably many of the errors will be taken care of in newer versions of the software.

One of the disadvantages of the microcomputer compared to the large computer is the limited memory capacity of the microcomputer. Although most microcomputers now on the market contain 64K bytes of memory, usually only 30-40K bytes are available for the program; the remainder of the memory is taken up by the operating system or reserved for special purposes. Thus, the size of the program which can fit into the microcomputer is limited to 30-40K bytes on many 64K byte microcomputers. For the North Star Horizon microcomputer used in this study which was running under the CP/M operating system, 37K bytes of the 64K bytes were avaialable for the program.

Both the Hooke and Jeeves pattern search program and the KSU-SUMT computer program were able to fit into the 37K bytes of available memory of the North Star Horizon microcomputer. However, the RAC-SUMT program was larger than the 37K bytes and thus would not fit into memory. To get around this problem, the original program was divided into two separate programs and only one of the programs was loaded at a time into memory. The RAC-SUMT program was able to run on the microcomputer in this way.

The size of the problem that can be solved by the RAC-SUMT program though is still limited. Whereas the RAC-SUMT program was dimensioned to solve a problem with 20 variables, 20 inequality constraints and 20 equality constraints, there is not enough memory to run a problem that large. This is because although the two separate parts of the RAC-SUMT program each fit into the computer memory, the user supplied routines must also fit into memory with the second part. The largest test problem used (4 variables, 9 inequality constraints) took up nearly all the available memory once it was loaded into the computer memory with the main program. Thus, a problem much larger than this will not fit into the North Star microcomputer.

Although the RAC-SUMT program is restricted by the 64K bytes of computer memory, the trend now is toward microcomputers with at least 128K bytes of main memory. With so much memory, the RAC-SUMT program along with the user-supplied subroutines will easily fit into the available memory. There will also be no need to divide the original program into two separate programs.

Another disadvantage of the micro/personal computer compared to the large computer is the slower execution speed of the micro/personal computer. The execution time of the test problems run on both the micro and the large computer showed that the micro was at least an order of magnitude slower than the large computer. In all test problems solved in this study, the micro/personal computer took less than four minutes to solve while the large computer solved all problems in less than five seconds. These problems were all solved using the single precision version of the programs. When the same problems were solved using double precision, the execution time on the micro/personal computer more than doubled. For example, test problem 2 solved by Hooke and Jeeves pattern search program took only 3 minutes using single precision but with double precision, it was still not finished after one hour of computation time.

The reason why the double precision version of the program took so much longer is that the calculation done in the program had to be carried out by software routines rather than hardware. At the time the Fortran software was purchased, there was hardware available to handle double precision, however, the Fortran software to take advantage of the special hardware was not yet available. As it becomes available, double precision will become less prohibitive to do on the micro/personal computer, but for now, if double precision results are needed, it will probably have to be done on the large computer.

However, for problems solved by single precision, the slower execution time as compared to the large computer was not significant in that execution time is only a small fraction of the overall time needed to solve a problem. Much more time is spent preparing data for the computer, entering the data into the computer, correcting mistakes in the data and waiting for results. For a micro/personal computer, the big savings in time is in not having to wait for a terminal or card punch to become available, waiting for turnaround time, and then waiting for the results to be printed. These savings in wait times are repeated every time the program has to be run because of errors in the data or changes made to the parameters in the program. So although the execution time of the micro/personal computer may be slower than for the large computer, the overall time needed to solve a problem will probably be less because of not having to wait for devices to become available.

Thus, from my experience on the micro/personal computer, I have found that on the plus side, the word processing capabilities on the micro/personal computer make program modification and correction a much easier task than before. Also on the plus side is the savings in time by not having to wait for a terminal to be free or waiting for the computer to process your job. On the negative side, the Fortran software for the micro/computer was not as developed as for the large computer, although this will probably be improved as newer versions come out. Another argument on the negative side is that the memory capacity of most micro/personal computers with 64K bytes of memory was not enough for the RAC-SUMT program, although this is also being corrected as newer micro/personal computers are coming out with more and more memory.

5.4 ADVANTAGES AND DISADVANTAGES OF USING THE MICRO/PERSONAL COMPUTER

The advantages of using a micro/personal computer include easy accessibility, low operating cost, adequate memory capacity to run the programs, no waiting for devices to become available, and results which are comparable to those for the large computer.

Disadvantages of using the micro/personal computer include the slower processing speed which makes programs using double precision arithmetic too slow to run on the micro. The slower processing speed though was not significant when running programs using single precision. Another disadvantage is the limited memory of the 64K microcomputer which restricts the size of problems that the RAC-SUMT program could solve. This limitation though is being overcome with the larger memory capacity of the newer micro/personal computers which allow memory expansion up to 512K bytes.

A third disadvantage is the problem encountered with a Fortran compiler which is still in the developing stage. The initial version of the Fortran compiler can be expected to still have errors in it and as was found out, it does not have all the features or error checking capabilities of the Fortran compiler for the large computer. We can expect that the Fortran software will improve as newer versions of it come out.

5.5 FUTURE STUDY

An interesting area of research would be to determine whether graphics could be used on the microcomputer to help in searching for a solution to the nonlinear programming problem.

5.6 REFERENCES

1. Crowder, H. P., R. S. Dembo, and J. M. Mulvey, "Reporting Computational Experiments in Mathematical Programming", Mathematical Programming, 15, pp. 316-329, 1978.

A COMPARATIVE STUDY OF NONLINEAR PROGRAMMING ROUTINES ON THE MICROCOMPUTER VERSUS THE LARGE COMPUTER

by

Frank P. Hwang

B.S., Kansas State University
Manhattan, Kansas 1981

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1984

ABSTRACT

With the microcomputer becoming ever more popular and affordable, a study was needed to determine the practicality and feasibility of putting nonlinear programming routines on the microcomputer.

The nonlinear programming programs under study were the Hooke and Jeeves Pattern Search, and two Sequential Unconstrained Minimization Techniques (SUMT), the KSU-SUMT program developed at KSU and the RAC-SUMT program developed at the Research Analysis Corporation, McClean, VA.

It was found from this study that the nonlinear programming programs would fit into the available memory of a 64K microcomputer. The size of problem that could be solved by the Hooke and Jeeves pattern search and the KSU-SUMT program was the same as for the large computer. However, for the RAC-SUMT program, a 64K microcomputer did not have enough memory to solve as large a problem.

In comparing the large computer versus the microcomputer for the nonlinear programming routines, it was found that the microcomputer compared favorably to the large computer in terms of ease of use, accuracy, and total time to run a problem. The operating system commands needed to run a Fortran program was somewhat easier to learn and remember for the microcomputer than for the large computer. The results of the test problems run on the microcomputer and large computer were nearly identical indicating that the accuracy of the results by the microcomputer were very good. In terms of total time needed to run a program which includes time needed to enter data into the terminal, wait for results and execution time, the microcomputer and large computer took about the same amount of time.