A COMPUTER IMPLEMENTATION AND TEST OF MIFFLIN'S ALGORITHM FOR NONLINEAR OPTIMIZATION

By

RODNEY WAYNE ROBISON

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1975

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
July, 1977

A COMPUTER IMPLEMENTATION AND TEST

OF MIFFLIN'S ALGORITHM FOR

NONLINEAR OPTIMIZATION

Thesis Approved:

Donald W. Grace
Thesis Adviser

James R. Van Doren

J. P. Chandler

[illegible]
Dean of Graduate College

989236

PREFACE

This thesis is a description, implementation, and test of a nonlinear optimization method as described by Robert Mifflin. The objective for the implementation was to compare the method with the method of Davidon, Fletcher and Powell.

The author wishes to express deep appreciation to his mother and father, John and Martha Robison, and his parents-in-law, Bill and Carol Woods, without whose guidance and support this education would not be possible.

Special thanks is due Dr. Donald Grace whose time and guidance in research and writing were invaluable.

I also wish to thank the entire faculty of the Computing and Information Sciences Department who took the time and patience to help me make this thesis and education possible. A thank you is also due Mrs. Pam Haught for typing the final copy of this thesis.

Finally, a special gratitude is due my lovely wife, Teresa, whose moral support and especially patience made all of this possible.

TABLE OF CONTENTS

TABLE

LIST OF FIGURES

CHAPTER I

OPTIMIZATION OF SYSTEMS

Introduction

The need to find the best combination and allocation of resources in order to maximize the yield of a system has always existed. The problem could be as simple as a farmer deciding how much and what to produce or as complex as scheduling manpower and finding the optimal configuration of machinery at a large refinery or manufacturing firm. Optimization techniques can also be applied to problems such as transportation schedules, diet schedules, or any problem where the input components or resources may be varied in order to optimize the output or objective of the system.

Many methods of attacking this optimizing problem have been developed. These algorithms range from crude brute force tactics to sophisticated and highly mathematical procedures. The method studied in this thesis employs both a brute force tactic and a mathematical procedure to find an optimal solution. The following definitions should aid in the discussion of the optimization of systems.

"A system is a collection of items from a circumscribed sector of reality that is the objective of study or interest. Therefore a system is a relative thing. In one situation a particular collection of objects may only be a small part of a larger system-a subsystem" (6, p. 3).

To consider the scope of a system, one must first observe the boundaries and the contents of the system. Inputs must be functionally described. The system processes must be well defined to show the effect of inputs on the system. Also, the result of those processes or objective of the system is the output value.

In order to study existing or proposed systems without building, disturbing, or destroying them, it is necessary to build a mathematical-logical economic model of the system and study the performance of that model rather than the actual system.

By using this model, we can change the values of certain system input variables and observe the effect on the system. This effect is measured by observing values taken on by certain system output variables or a combination of these variables called an objective function. Optimization is a technique or method of trying to find input variables of the model that maximize or minimize the objective value or show a stepwise improvement. The two most widely used techniques or methods of such problem-solving are simulation and mathematical programming.

In mathematical programming, we find an analytical representation of the system in terms of x_i's which represent the resources of the system. This representation consists of, first, an objective function that measures the effectiveness of a combination or allocation of system resources and second, if necessary, constraining functions that bound the amounts of resources available or constrain the values any x_i may take on. These functions form a solution space of feasible candidates for choices of x_i. If the choice of the x_i's is unrestricted, the problem is one of unconstrained minimization or maximization. Otherwise, when the x_i's are restricted in the values they are allowed to take on, then the problem is one of constrained minimization or maximization.

The mathematical program can also be further classified by determining if the objective function or constraining functions are linear or nonlinear. If the objective or any constraining function is nonlinear as shown in Figure 1, then the program is said to be nonlinear. Figure 2 demonstrates the case where the objective and all constraining functions are linear. This program is said to be a linear program.

In a linear program, if a local optimum is found, then it is guaranteed to be a global optimum. With nonlinear programs, this is not always the case. However, a class of nonlinear problems can be defined which are guaranteed to be free of multiple local optima. These are called convex programming problems.

A convex programming problem is one of minimizing a convex function or maximizing a concave function over a convex constraint set. Any local minimum of a convex programming problem is a global minimum. Convexity is a property of both a set and a function. A function is convex if a line segment drawn between any two points on the graph of the function never lies below the graph, and concave if it never lies above the graph. Algebraically a function f is convex if

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$$

for all x_1, x_2 in the domain of the definition of f and for $0 \leq \lambda \leq 1$. That is, a linear interpolation never underestimates the function. A set is said to be convex if for any two points in the space the line segment joining them is also in the space. Algebraically for a space S to be convex, $L \subset S$ where

Consider the problem

$$\text{minimize } z = (x_1 - 3)^2 + (x_2 - 4)^2$$

subject to the linear constraints

$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \\ 5 - x_1 - x_2 &\geq 0 \\ -2.5 + x_1 - x_2 &\leq 0 \end{aligned}$$

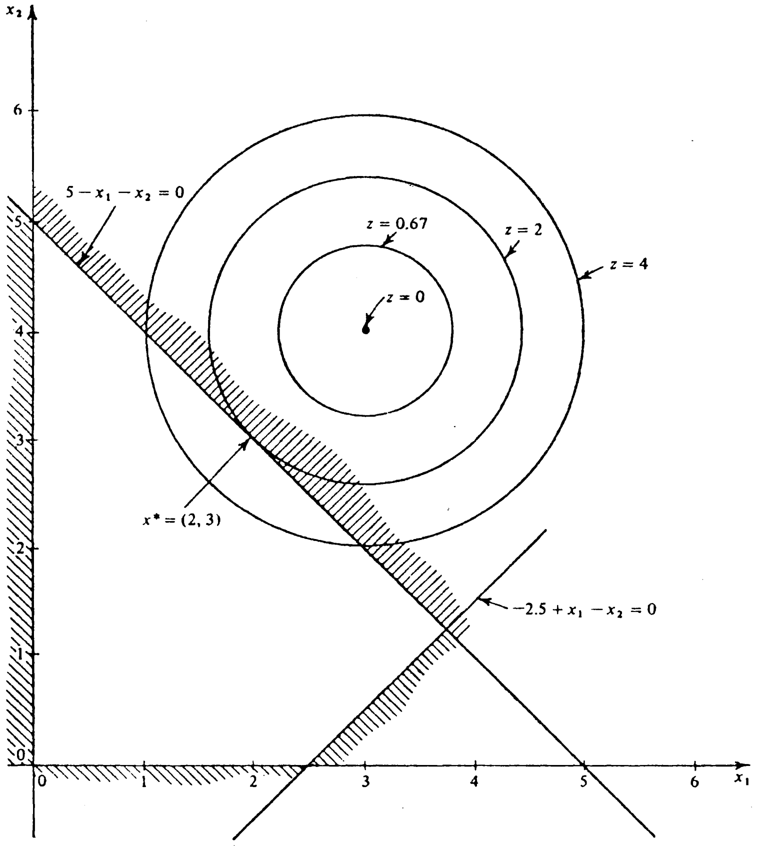

Figure 1. Example of a Nonlinear Program (4).

Geometry of Linear Programs. Consider the problem

$$\text{maximize } z = x_1 + 3x_2$$

subject to

$$-x_1 + x_2 \leq 1$$
$$x_1 + x_2 \leq 2$$
$$x_1 \geq 0, \qquad x_2 \geq 0$$

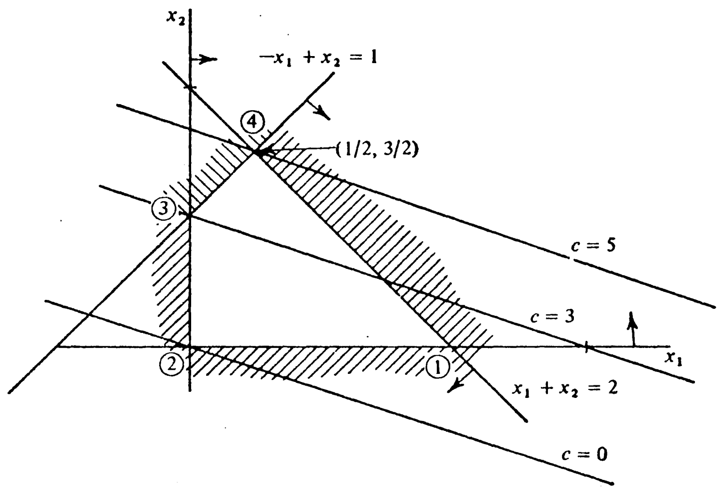

Figure 2. Example of a Linear Program (4).

$$L = \{x | x = \lambda x_1 + (1-\lambda)x_2, 0 \leq \lambda \leq 1\}$$

Although convexity is desirable, many real-world problems turn out to be nonconvex. In addition, there is no simple way to test a nonlinear problem for convexity because there is no simple way to test a nonlinear function for this property.

Many, if not most, existing methods of nonlinear programming fall roughly into two categories:

(1) methods of feasible directions, and

(2) penalty function techniques.

In methods of feasible directions first pick a starting point and find a direction such that a move in that direction violates no constraint and the objective function improves in that direction. One then moves a distance in this direction, obtaining a new and better point, and repeats the procedure until a point is obtained such that a direction can be found that violates no constraints and improves the objective value.

Penalty function techniques combine objective and constraining functions into a "penalty" function which is optimized with no constraints. In this way, a constrained problem is solved using unconstrained methods. Since unconstrained methods are easier and many powerful unconstrained algorithms exist, this is a very valuable tool. A not-so-practical example of this concept is in the problem requiring

$$\text{minimize } f(x)$$
$$\text{subject to } g(x) \geq 0.$$

Define

$$P(x) = f(x) + G(x)$$

where

$$G(x) = \begin{cases} \infty & , g(x) < 0 \\ 0 & , \text{elsewhere} \end{cases}$$

Chapter II will discuss a method of feasible directions proposed by Robert Mifflin of Yale University in 1974. This method is for unconstrained minimization of a real-valued function f defined on R^n and does not require the evaluation of partial derivatives of f. The algorithm is partly an approximate Newton method where both first and second order partial derivatives are approximated from function values and partly a method of location variations.

CHAPTER II

A SUPERLINEARLY CONVERGENT ALGORITHM FOR MINIMIZATION WITHOUT EVALUATING DERIVATIVES

This algorithm for unconstrained minimization of a real valued function of n variables, was presented by Robert Mifflin (7) of Yale University. "It is a second order extension of the method of local variations and it does not require any exact one variable minimizations. This method retains the local variations property of accumulation points being stationary for a continuously differentiable function. Furthermore, because this extension makes the algorithm an approximate Newton method, its convergence is superlinear for a twice continuously differentiable strongly convex function" (p. 100). That is,

$$\{||\underline{x}^{k+1} - \underline{x}^{*}||/||\underline{x}^{k} - \underline{x}^{*}\} \rightarrow 0 \quad \text{as } k \rightarrow \infty$$

where $\{\underline{x}^k\} \subset R^n$ is the algorithm sequence and $\underline{x}^* \varepsilon R^n$ minimizes f.

The Mifflin algorithm finds a candidate for the next base point or move point by combining both exploratory moves and searching a downhill or favorable direction. Of the points generated by these two methods, the one with the smallest functional value is kept as the candidate for the next base point. Then, if this point shows a better of smaller functional value is kept as the candidate for the next base point. Then, if this point shows a better of smaller functional value, it replaces the current base point and the process is repeated. If the candidate point is not an improvement, it is rejected as the new base point, the stepsize is reduced, and the process is repeated. The algorithm terminates when the stepsize and the functional improvement reach some user specified lower limits.

The algorithm parameters required are positive real numbers α, β, γ, δ, and ρ with $\rho<1$ and $\beta^2<(\rho|2n^2\gamma)$. The parameter δ is related to the word length of the computer being used and is chosen to avoid numberical problems such as overflow, resulting from division by small numbers. The parameter γ is an absolute bound over the elements of the matrix $\Delta^2 f$ and is used to keep the matrix bounded. The parameter α is an expansion factor used in a test of how the stepsize relates to the gradient norm. The parameter ρ and β are used in convergence testing.

Given the above parameters, the algorithm is as follows:

Step). Choose a starting solution point $\underline{x}\varepsilon R^n$ and a starting stepsize $s > 0$. Set the index $k = 1$ and the sequence values $\underline{x}^1 = \underline{x}$ and $s_1 = s$.

Step 1. Compute an n-vector of approximate first partial derivatives Δf by
$\Delta f_i = (1/2s)[f(\underline{x}+s\underline{e}_i) - f(\underline{x}-s\underline{e}_i)]$ for $i = 1,2,\ldots,n$
and an approximate gradient norm

$$||\Delta f|| = [\sum_{i=1}^{n}(\Delta f_i)^2]^{\frac{1}{2}}$$

Set the descent direction indicators

$$\sigma_i = \begin{cases} +1 & \text{if } \Delta f_i \le 0, \\ -1 & \text{if } \Delta f_i > 0, \end{cases} \quad \text{for } i = 1,2,\ldots,n$$

Define a best axis point x_a by

$$f(\underline{x_a}) = \min_{1\le i\le n} f(\underline{x}+s\sigma_i\underline{e}_i)$$

Step 2. Compute a n by n symmetric matrix of approximate second partial derivatives by

$$\Delta^2 f_{ii} = (1/s^2)[f(\underline{x}+s\underline{e}_i) + f(\underline{x}-s\underline{e}_i) - 2f(\underline{x})] \text{ for } i = 1,2,\ldots n,$$

$$\Delta^2 f_{ij} = (\sigma_i\sigma_j/s^2)[f(\underline{x}+s\sigma_i\underline{e}_i+s\sigma_j\underline{e}_j) + f(\underline{x}) - f(\underline{x}+s\sigma_i\underline{e}_i) - f(\underline{x}+s\sigma_j\underline{e}_j)] \text{ for } 1 \le j < i \le n$$

Define a best corner point $\underline{x}_c$ by

$$f(\underline{x}_c) = \min_{1\le j\le 1\le n} f(\underline{x}+s\sigma_i\underline{e}_i+s\sigma_j\underline{e}_j),$$

and a possible move point $\underline{x}_m$ by

$$f(\underline{x}_m) = \min\,[f(\underline{x}_a), f(\underline{x}_c)].$$

Step 3. For $1 \le j \le i \le n$, if $|\Delta^2 f_{ij}| > \gamma$, replace $\Delta^2 f_{ij}$ by γ sign $(\Delta^2 f_{ij})$.
Using the Modified Cholesky Factorization Procedure described later, with $H = \Delta^2 f$, compute matrices L, D and E such that $LDL^T = \Delta^2 f + E$.
Define an index q by

$$D_{qq} - E_{qq} = \min_{1\le i\le n}(D_{ii} - E_{ii})$$

Step 4. If $\alpha s > ||\Delta f||$ and $D_{qq}-E_{qq}>0$ go to step 7. If $\alpha s \le ||\Delta f||$, compute $\underline{y}^1$ satisfying

$LDL^T\underline{y}^1 = -\Delta f$

and set $p = 1$; and if $E \ne 0$, set

$\underline{y}^2 = -(||\underline{y}^1||/||\Delta\underline{f}||)\Delta f$

and $p = 2$, and if $DD_{qq} - E_{qq} < 0$,
compute $\underline{z}$ satisfying $L^T\,\underline{z} = e_q$ and set
$y^3 = 1\ \text{sign}(\underline{z}^T\Delta f)(||y^1||/||\underline{z}||)\underline{z}$
and set $p = 3$,
and define a search direction vector
$\underline{d}$ as the $\underline{y}^i$ which satisfies:
$\underline{d}^T\Delta f + \tfrac{1}{2}d^T\Delta^2 f\underline{d} = \min_{1\le i\le p}\,[(\underline{y}^i)^T\Delta f + (\underline{y}^i)^T(LDL^T - E)\underline{y}^i$.
Otherwise ($\alpha s > ||\Delta f||$ and $D_{qq}-E_{qq} < 0$) compute $\underline{z}$ as above and set
$d = -\text{ sign }(\underline{z}^T\ f)\underline{z}$.

Step 5. Compute, if possible, a search point $\underline{x} + t\underline{d}$, where t is a positive number satisfying

$f(\underline{x}+t\underline{d}) \le \rho t(\underline{d}^T\ \Delta f + \tfrac{1}{2}\ t d^T\Delta^2 f\underline{d})$.
The parameter ρ is chosen less than 1 because if f is nearly a strictly convex quadratic function in a neighborhood of a nonstationary point $\underline{x}$, $\Delta f \ne 0$ and $\Delta^2 f$, which is approximately the positive definite matrix $\nabla^2 f(x)$, is not modified at step 3 then

$d^T\Delta f + \tfrac{1}{2}d^T\Delta^2 fd < 0,$
$f(\underline{x}+d) - f(\underline{x}) < \rho(d^T\Delta f + \tfrac{1}{2}\underline{d}^T\Delta^2 f\underline{d})$.
and therefore, $t = 1$, satisfies the inequality of step 5. Thus, the approximate Newton point and, therefore, the search process should try $t = 1$ first whenever $\Delta^2 f$ is positive definite.

Step 6. If $f(\underline{x}_m)-f(\underline{x}) > -\alpha^2\beta^2s^2$, go to step 7. If $f(\underline{x}_m) - f(\underline{x}) \leq -\beta^2 \times ||\Delta f||^2$, choose some reduced stepsize $r\varepsilon(o,s]$ and go to step 8. Otherwise set $r = s$ and go to step 9.

Step 7. There was not a sufficient function value decrease and a move is not possible so set $r = \frac{1}{2}s$ and $\underline{x}_m = \underline{x}$.

Step 8. If $\underline{x} \neq \underline{x}^k$ replace k by k + 1. Set the sequence values $\underline{x}^k = \underline{x}$ and $\underline{s}_k = \underline{s}$.

Step 9. Replace $\underline{x}$ by $\underline{x}_m$ and s by r and to to step 1.

Termination criterion. In practice the algorithm could be stopped when s and $(f(\underline{x}) - f(\underline{x}_m))$ are both below some user specified limits or when an upper bound on the number of function evaluations is exceeded.

Modified Cholesky Factorization Procedure

"Positive definite symmetric matrices may be factored into triangular matrices that are transposes of each other. We have

$$A_s = L_s L_s^T$$

and the decomposition is often called the square-root factorization. It is extremely stable, never requires interchanging to avoid small pivots, and requires the least calculational labor of all decomposition, largely because of the symmetry. Positive definiteness, however, is essential lest complex elements appear in the factors. This restriction is not serious, for all symmetric matrices have real eigenvalues, and one may add a constant to all the eigenvalues simply by adding that same constant to the principal diagonal of the matrix. (Positive definiteness only requires all the eigenvalues to be positive.) Thus the Cholesky version of LR is the favorite algorithm of the family for symmetric matrices - adjusted if necessary to ensure positive eigenvalues" (1 p. 348). A modified version of the Cholesky algorithm follows.

Given a n by n symmetric matrix H and a positive number δ, this procedure determines a unit diagonal lower triangular matrix L, a positive diagonal matrix D and a nonnegative diagonal matrix E such that

$$LDL^T - E = H,\ D_{ii} \geq \delta > 0 \text{ for } i = 1,2,\ldots,n,$$

and

$$|(LDL^T)_{ij}| = |(H + E)_{ij}| < n\gamma \text{ for } 1 < j < i < n,$$

where

$$\gamma = \max[\delta, \max_{1 \leq j \leq i \leq n} |H_{ij}|].$$

This factorization is designed so that if H is positive definite and δ is sufficiently small, then E = 0 and, hence, $LDL^T = H$. The procedure is as follows:

```
Set $j = 1$.

Loop:  If $j = n + 1$, stop.  Otherwise, compute

          $L_{jr} = C_{jr}/D_{rr}$ for $r = 1,2,\ldots,j-1$

       $C_{ij} = H_{ij} - \sum_{r=1}^{j-1} C_{ir} L_{jr}$ for $i = j, j+1, \ldots, n$,

       $D_{jj} = \max[\delta, |C_{jj}|, (1/\gamma) \max_{j+1 \leq i \leq n} /C_{ij}/^2]$,

       $E_{jj} = D_{jj} - C_{jj}$

    Replace $j$ by $j + 1$ and go to Loop.
```

In steps 1 and 2 the first and second order derivatives are approximated. These approximations will be exact if f is a quadratic. A total of $\frac{1}{2}(n+n^2)$ function evaluations are required for this approximation. A total of $\frac{1}{2}(n+n^2)$ exploratory moves are considered as the trail move point. These exploratory points do not require extra function evaluations other than those used in approximating derivatives.

Step 3 first ensures that the approximate Hessian matrix $\Delta^2 f$ is bounded. The parameter γ should be sufficiently large and δ sufficiently small that $\Delta^2 f$ is not modified whenever $\Delta^2 f$ is positive definite.

Therefore γ should be chosen to be an upper bound over the elements of the matrix of second partials over the optimization region. The matrix of second partials is then factorized by the Modified Cholesky Factorization such that

$$LDL^T - E = \Delta^2 f$$

These results will be used in determining the best search direction in step 4.

In step 4, if $D_{qq} - E_{qq} < 0$ then there is an indication of negative curvature along the direction vector $\underline{z} = (L^T)^{-1}\underline{e}_q$. The search direction vector $\underline{d}$ is then chosen from up to three possible candidates $\underline{y}^i$ providing the stepsize is small relative to the approximate gradient norm or there is an indication of negative curvature. The $\underline{y}^1$ direction is an approximate Newton direction. The $\underline{y}^2$ direction is the negative gradient direction and $\underline{y}^3$ is the $\underline{z}$ vector above. This has been found to be a good search direction if there is an indication of negative curvature.

The best choice of the $\underline{y}^i$ is then determined by choosing the $\underline{y}^i$ which satisfies:

$$\underline{d}^T\Delta f + \tfrac{1}{2}\Delta^2 f\underline{d} = \min_{1 \leq i \leq p}[(\underline{y}^i)^T\Delta f + \tfrac{1}{2}(\underline{y}^i)^T(LDL^T - E)\underline{y}^i]$$

"Preliminary computational experience indicate the $\underline{y}^i$ that minimizes the two term Taylor series to be the best choice" (Mifflin, p. 105).

In step 5 the value of t is to be sought by a one-variable minimization search process. The move point from step 2 is replaced by $\underline{x} + t\underline{d}$ if $\underline{x} + t\underline{d}$ has a smaller function value than the better of $\underline{x}_a$ and $\underline{x}_c$.

In steps 6 and 7, if there is not a sufficient function value decrease relative to s^2, then a move is not desirable. The stepsize is halved at step 7 and there is a return to step 1 by way of steps 8 and 9 with $\underline{x}$ unchanged. Otherwise a second function value decrease test is made, this time relative to $||\Delta^2 f||$. Sufficient decrease here allows us to reduce the stepsize to any positive value not exceeding the current stepsize and to define $\underline{x}$ as a sequence point at step 8. Insufficient decrease leaves the stepsize unchanged and bypasses step 8.

"In step 8 the sequence values are defined with the properties $f(\underline{x}^k) > f(\underline{x}^{k+1})$ and $s_k \geq s_{k+1}$. If f is strongly convex then all of the points become sequence points" (Mifflin, p. 107).

The Mifflin algorithm will be compared to the algorithm of Davidon, Fletcher and Powell in Chapter 3. The algorithm of Davidon, Fletcher and Powell is described by R. Fletcher and M. J. D. Powell (Vol. 6, Iss. 2, 1963, pp. 163-168). "A Rapid Descent Method for Minimization", Computer Journal. The program for the Davidon, Fletcher and Powell method was obtained through IBM's Scientific Subroutine Package library.

CHAPTER III

COMPARISON OF THE MIFFLIN ALGORITHM TO THAT OF DAVIDON, FLETCHER AND POWELL

To minimize $f(\underline{x})$, we can start with the Taylor's expansion of $f(\underline{x})$ about $\underline{x}_0$.

$$f(\underline{x}) = f(\underline{x}_0) + \nabla f(\underline{x}_0)(\underline{x}-\underline{x}_0) + \tfrac{1}{2}(\underline{x}-\underline{x}_0)^T \nabla^2 f(\underline{x}_0)(\underline{x}-\underline{x}_0) + \ldots$$

The first three terms closely resemble the general quadratic function.

$$F(\underline{x}) = C + \underline{b}\ \underline{x} + \underline{x}\ A\underline{x}$$

If we want to minimize $f(\underline{x})$, we can do so by truncating the Taylor's expansion, differentiating, setting this result to zero, and solving for $\underline{x}$.

$$\frac{\partial f(\underline{x})}{\partial\ \underline{x}} \cong \nabla f(\underline{x}_0) + \nabla^2 f(\underline{x}_0)(\underline{x}-\underline{x}_0)$$

$$0 = \nabla f(\underline{x}_0) + \nabla^2 f(\underline{x}_0)(\underline{x}-\underline{x}_0)$$

$$\underline{x}-\underline{x}_0 = -[\nabla^2 f(\underline{x}_0)]^{-1}\nabla f(\underline{x}_0)$$

$$\underline{x} = \underline{x}_0 - [\nabla^2 f(\underline{x}_0)]^{-1}\nabla f(\underline{x}_0)$$

This gives a new approximation for $\underline{x}$ based on an initial given, $\underline{x}_0$. In general, this iterative algorithm is:

$$\underline{x}_{i+1} = \underline{x}_i - \underline{x}_{i-1} - \ldots - \underline{x}_0$$

Since the first three terms of the Taylor's expansion are used this approximation is exact for a quadratic. Notice also that both the direction and the stepsize are determined.

General minimization procedures can be designed which will minimize a quadratic function of n variables in n steps. Many are based on the ideas of conjugate directions (4).

The general quadratic function can be written as above and letting $\underline{x}^*$ minimize $F(\underline{x}) = 0$.

$$\nabla F(\underline{x}^*) = b + A\underline{x}^* = 0 \tag{3.1}$$

Given a point $\underline{x}_0$ and a set of linearly independent directions $\{\underline{s}_0, \underline{s}_1, \ldots, \underline{s}_{n-1}\}$, constants β_i can be found such that

$$\underline{x}^* = \underline{x}_0 + \sum_{i=0}^{n-1} \beta_i \underline{s}_i \tag{3.2}$$

If the directions $\underline{s}_i$ are A-conjugate, i.e., satisfy

$$\underline{s}_i^T A \underline{s}_j = 0, \; i \neq j, \; i,j = 0, 1, \ldots, n-1 \tag{3.3}$$

and none are zero, then the $\underline{s}_i$ are easily shown to be linearly independent and the β_i can be determined as follows:

$$\underline{s}_j^T A \underline{x}^* = \underline{s}_j^T A \underline{x}_0 + \sum_{i=0}^{n-1} \beta_i \, \underline{s}^T A \underline{s}_i$$

$$\underline{s}_j^T A \underline{x}^* = \underline{s}_j^T A \underline{x}_0 + \beta_j \underline{s}_j^T A \underline{s}_j$$

$$\beta_j = -(\underline{b} + A\underline{x}_0)^T \frac{\underline{s}_i}{\underline{s}_j^T A \underline{s}_j} \tag{3.4}$$

Now consider an iterative procedure, starting at $\underline{x}_0$ and successively minimizing $F(\underline{x})$ down the directions $\underline{s}_0, \underline{s}_1, \ldots, \underline{s}_{n-1}$, where these directions satisfy (3.3). Successive points are then determined by the relations

$$\underline{x}_{i+1} = \underline{x}_i + \alpha_i \underline{s}_i, \; i = 0, 1, \ldots, n-1 \tag{3.5}$$

where α_i is determined by minimizing $f(\underline{x}_i + \alpha \underline{s}_i)$, as in the optimum gradient method, so that

AND (3,5)
$$\underline{s}_i^T \nabla F(\underline{x}_{i+1}) = 0 \qquad (3.c)$$

using (3.1) in (3.6) gives

$$\underline{s}_i \; (b + A(\underline{x}_i + \alpha_i \underline{s}_i)) = 0$$

or

$$\alpha_i = -\; (b + Ax_i)^T \frac{\underline{s}_i}{\underline{s}_i^T A \underline{s}_i}$$

From (3.5),

$$\underline{x}_j = \underline{x}_0 + \sum_{j=0}^{i-1} \alpha_j \underline{s}_j$$

so that

$$\underline{x}_i^T A \underline{s}_i = \underline{x}_0^1 A \underline{s}_i + \sum_{j=0}^{i-1} \alpha_j \underline{s}_j^T A \underline{s}_i = \underline{x}_0^T A \underline{s}_i$$

and

$$\alpha_i = -\; (\underline{b} + a\underline{x}_0)^T \frac{s_i}{s_i^T A s_i}$$

which is identical to (3.4) Hence, this sequential process leads, in n steps, to $\underline{x}^*$ where the minimum is attained.

"A method presented by Fletcher and Powell is probably the most powerful general procedure now known for finding a local minimum of a general function $f(\underline{x})$. It is designed so that, when applied to a quadratic, it minimizes in n iterations. It does this by generating conjugate directions" (4 p. 7). This method, invented by Davidon, shall further be referred to as DFP. An iteration of this method as described by Lasdon (4) follows.

$$H_0 = \text{any positive definite matrix}$$

$$\underline{s}_i = -H_i \nabla f(\underline{x}_i)$$

Choose $\alpha = \alpha_i$ by minimizing $f(\underline{x}_i + \alpha \underline{s}_i)$,

$$\underline{\sigma} = \alpha_i \underline{s}_i$$

$$\underline{x}_{i+1} = \underline{x}_j + \underline{\sigma}_i$$

$$H_{i+1} = H_i + A_i + B_i$$

where the matrices A_i and B_i are defined by

$$A_i = \frac{\underline{\sigma}_i \underline{\sigma}_i^T}{\underline{\sigma}_i^T \underline{y}_i} , \quad \underline{y}_i = \nabla f(\underline{x}_{i+1}) - \nabla f(\underline{x}_i)$$

$$B_i = \frac{-H_i \underline{y}_i \underline{y}_i^T H_i}{\underline{y}_i^T H_i \underline{y}_i}$$

Notice that the numerators of A_i and B_i are both matrices, while the denominators are scalars. Thus, starting with H_o, these matrix adjustments are added to H_i to form H_{i+1}, while maintaining positive definiteness. Davidon, Fletcher and Powell (4) prove the following:

1. The matrix H_i is positive definite for all i. As a consequence of this, the method will usually converge, since

$$\frac{\partial}{\partial \alpha} f(\underline{x}_i + \alpha \underline{s}_i) \Big|_{\alpha=0} = -\nabla f^T(\underline{x}_i) \, H_i \nabla f(\underline{x}_i) < 0$$

   That is, the function f is initially decreasing along the direction $\underline{s}_i$, so that the function can be decreased at each iteration by minimizing down $\underline{s}_i$.

2. When the method is applied to the quadratic, then

   (a) the direction $\underline{s}_i$ (or equivalently $\underline{\sigma}_i$ are A-conjugate, thus leading to a minimum in n steps.

   (b) the matrix H_i converges to the inverse of the matrix of second partials of the quadratic.

Both Mifflin's algorithm and the DFP algorithm are similar since they both employ a search in a downhill direction for a new base point. Both methods also use some form of derivatives to determine the downhill direction.

They differ in the method used to find the derivatives. Davidon, Fletcher and Powell require the user to supply an analytical representation of the first derivative that is evaluated with each function evaluation of an exploratory point. This is, of course, dependent upon the implementation used. Derivatives could just as well be approximated by differences. The important thing to note is DFP requires only first derivative calculation. This calculation is then used to determine the first partials and matrix of conjugate directions. The Mifflin algorithm determines first and second derivatives by differences and given the functional value of the exploratory point require 2n function evaluations for the first derivative and $\frac{1}{2}(n^2 - n)$ function evaluations for the second derivative. This derivative calculation implies more input and work for the user of DFP in supplying the first derivative analytically and faster convergence because of this added accuracy over the difference method of calculating derivatives.

The algorithm of Mifflin also differs from that of Davidon, Fletcher and Powell by having more than one method of selecting a new base point. Along with a search in a downhill direction, the Mifflin algorithm also tries $\frac{1}{2}(n^2 + n)$ exploratory moves in a fixed set of directions. In each iteration, the best move of these two methods--the one with the smallest functional value--is taken to be the next base point. This procedure requires no extra function evaluations over those required in calculating derivatives.

In order to further compare and test the performance of the two algorithms, define the following various functions and their numbers for table reference.

Function 1. $f(x,y) = (x - 5)^2 + (y - 5)^2$
This is a quadratic function with a minimum of 0 at (5,5). Figure 3 illustrates the contours of this function.

Function 2. $f(x,y) = x^4 + y^2 - 10x$
This is a quartic function with a minimum of approximately -10.179 at approximately (-13.572, 0). Figure 4 illustrates the contours of this function.

Function 3. $f(x,y) = 100(y - x^2)^2 + (1 - x)^2$
The Rosenbrock, or "parabolic valley", function with a minimum of 0 at (1,1). Figure 5 illustrates the contours of this function.

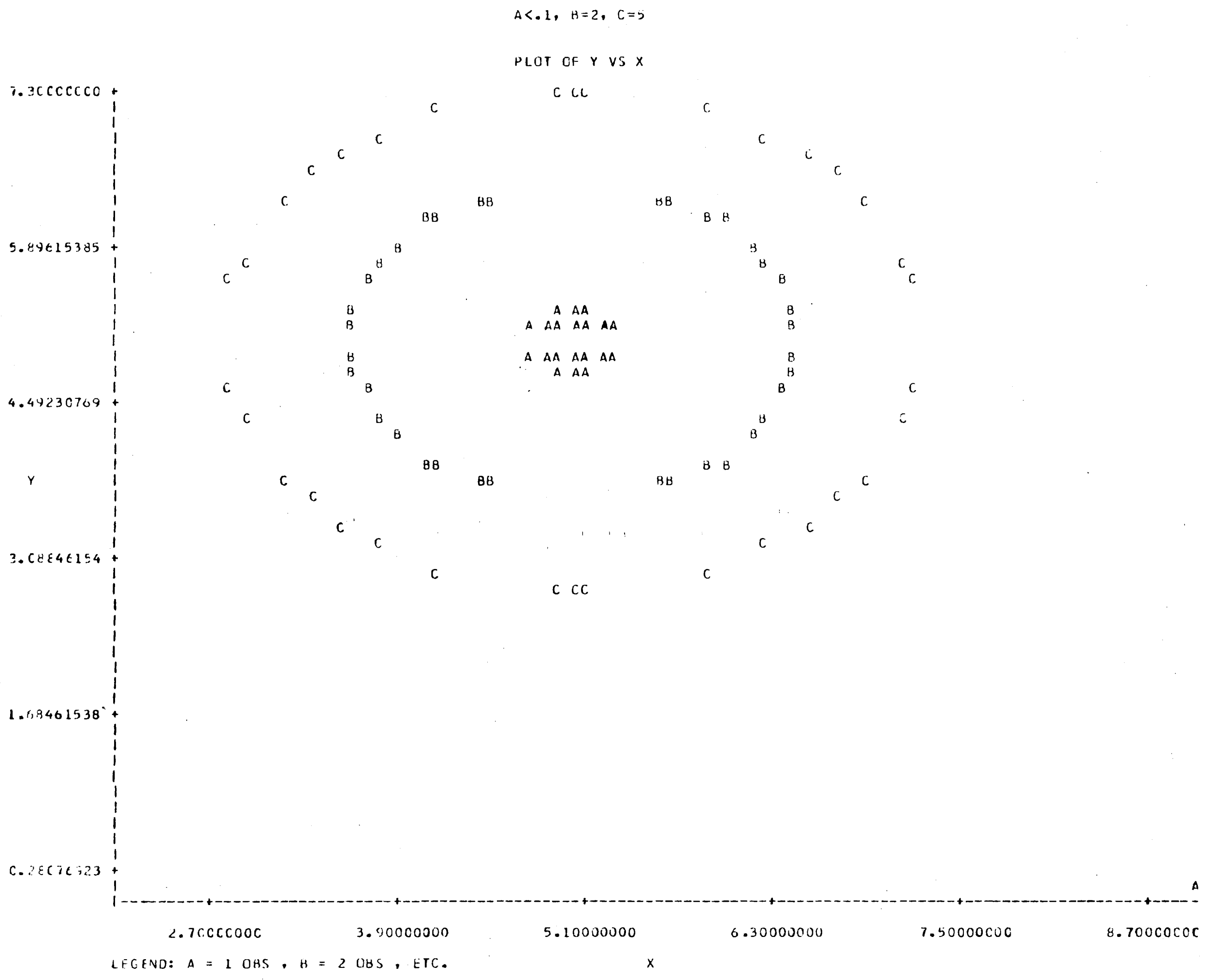

Figure 3. Contour Lines of Function 1.

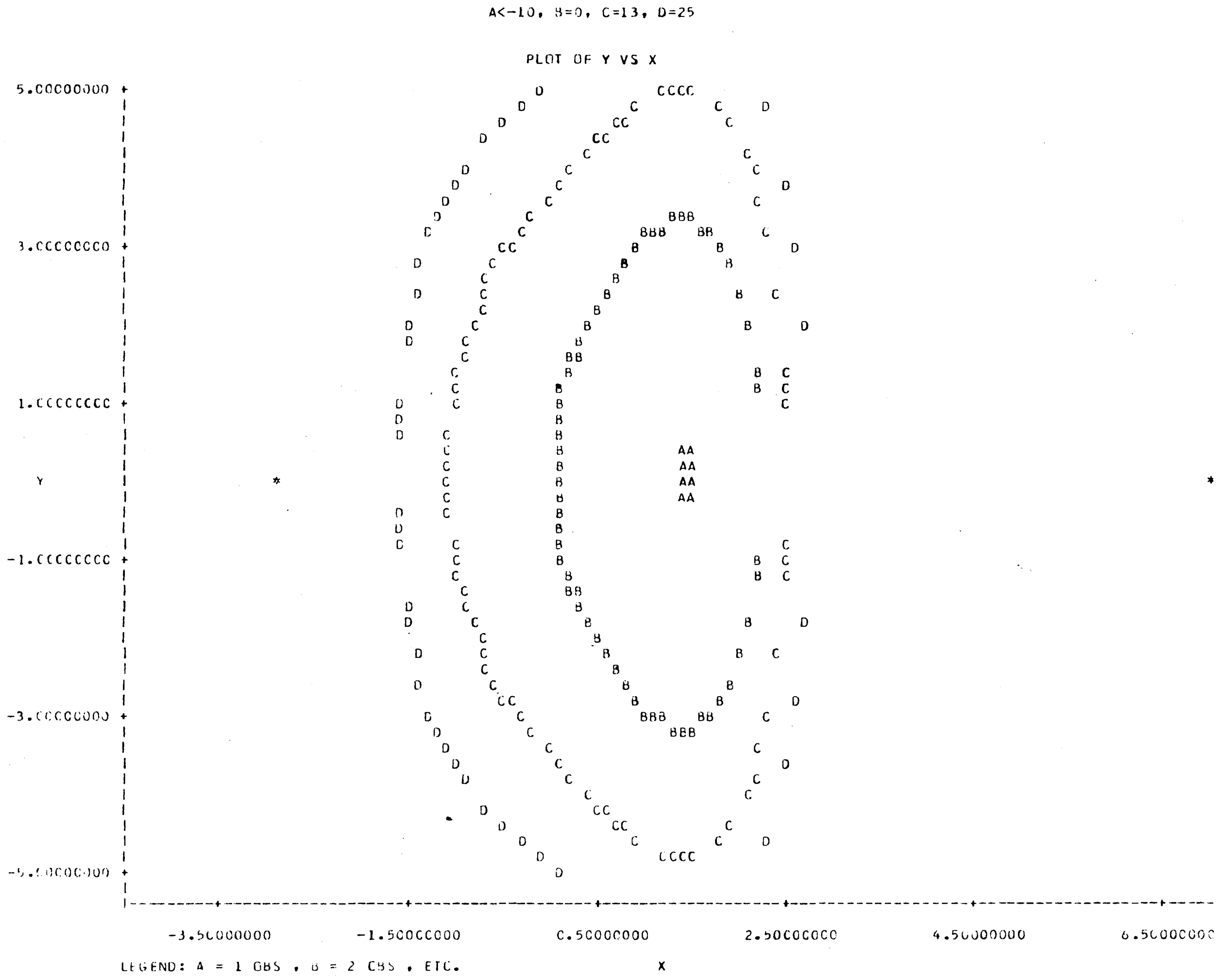

Figure 4. Contour Lines of Function 2.

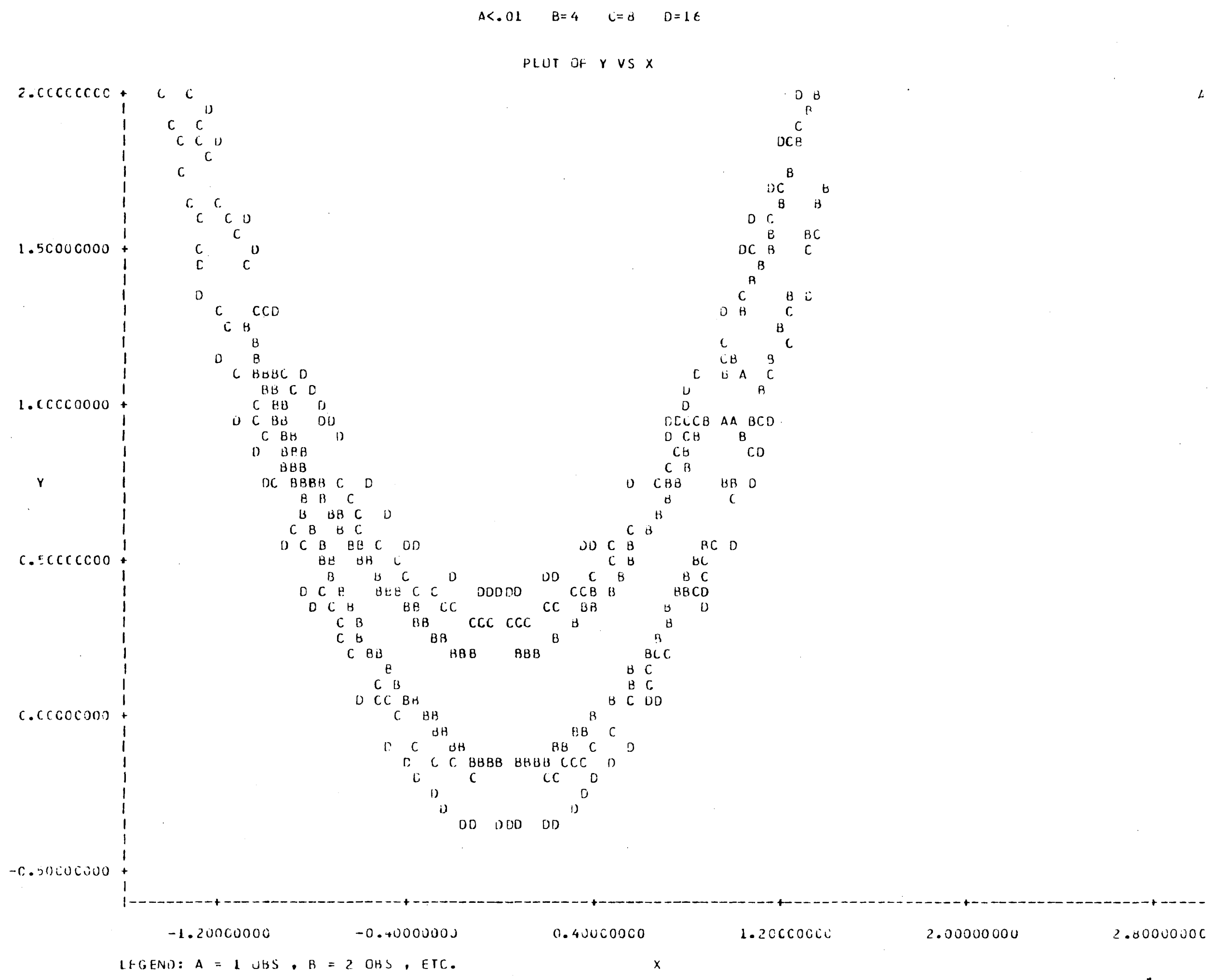

Figure 5. Contour Lines of Function 3.

TABLE I

COMPARISON OF THE ACTUAL PERFORMANCE OF THE MIFFLIN ALGORITHM TO THE DAVIDON FLETCHER POWELL ALGORITHM

Function 1	Function Evaluations	Function Value	Iterations
Mifflin	9	0	1
DFP	3	0	2
Function 2			
Mifflin	46	-.10079 D 02	6
DFP	20	-.10079 D 02	7
Function 3			
Mifflin	182	.93617 D-13	25
DFP	60	.2 D-26	18

Table I illustrates the performance of the two algorithms on each of the functions described. Notice that, as expected, the number of function evaluations required by the Mifflin algorithm is higher than the number required by Davidon, Fletcher and Powell. This is, as expected, because of the derivative calculation made by Mifflin not required by Davidon, Fletcher and Powell.

Function 1 was easily minimized by both algorithms with a starting point of (0,0) and an initial stepsize of .1 . As expected Mifflin solved the quadratic in one iteration using 9 function evaluations. DFP solved the problem in 2 iterations requiring only 3 function and first derivative evaluations.

Function 2 was solved by both algorithms with a starting point of (-3, -3) and an initial stepsize of 1. Mifflin's algorithm solved the problem with slightly fewer iterations than DFP. The exploratory move of Mifflin proved to be an advantage on this problem and often provided a better move point than the line search.

Function 3 was solved by both algorithms with a starting point of (-1.2, 1.) and an initial stepsize of .1 . DFP solved the Rosenbrock function with 60 function evaluations in 18 algorithm iterations. Mifflin's algorithm, however, converged slowly and require 180 function evaluations in 25 algorithm iterations.

It should be noted that Mifflin's algorithm requires on the order of n^2 function evaluations per iteration as compared to on the order of n function evaluations per iteration by DFP. This is due to the fact that Mifflin's algorithm approximates first and second partial derivatives and the DFP algorithm makes a first partial derivative evaluation with each function evaluation. This approximation by Mifflin could also lead to numerical and accuracy problems often incurred in calculating and using second derivatives.

The Mifflin's algorithm also has no lower bound on the stepsize, which may lead to round-off errors particularly in calculating derivatives. Scaling errors may occur, particularly in the Cholesky factorization calculations of L and D if the choice of δ is too small.

It would seem that the method presented by Mifflin would be a good choice for minimization if the user is willing to use on the order of n^2 function evaluations per iteration as compared to on the order of n function evaluations per iteration used by DFP. Mifflin's method would although, have some power where the matrix of second partials is not positive definite because of the exploratory move as a "back-up" possibility of a new base point.

A modification to Mifflin's algorithm that might improve the performance would be to either calculate first and second order partials analytically or to calculate first partials analytically and second partials by differences of first partials. If possible, this could cut down the number of function evaluations and replace the approximation of derivatives by exact derivatives.

Other modifications of updating only parts of the matrix of second partials and faster Cholesky factorizations when the Cholesky factors are known could also be designed (7).

In conclusion, it is suggested that the Mifflin algorithm as presented here be avoided. "There are a number of minimization techniques which do not require derivatives. Of these, tests performed thus far indicate that Powell's method is the most efficient" (Lasdon, P.11). If derivatives are known analytically or maybe approximated, then DFP certainly would be a better choice.

One last caution to the user of any mathematical program is that the most that can be guaranteed of Mifflin's or any other minimization technique without limiting the objective functions, is that it will find a local minimum. In general, this is the point nearest the starting point.

SELECTED BIBLIOGRAPHY

(1) Acton, F. S. Numerical Methods that Work. Harper and Row, New York, Evanston, and London.

(2) Colella, A. M., and O'Sullivan, M. J., and Carlino, D. J. Systems Simulation; Methods and Application. D. C. Heath Co., Lexington, Mass.

(3) Gale, David. The Theory of Linear Economic Models. McGraw-Hill Book Co., New York, Toronto, and London.

(4) Lasdon, Leon S. Optimization Theory for Large Systems. The MacMillan Co., New York.

(5) Meier, Newell, Pazer. Simulation in Business and Economics. Prentice-Hall, Englewood Cliffs, N. J.

(6) Pritsker, A. B. The GASP IV Simulation Language. John Wiley & Sons, Inc., New York.

(7) Mifflin, Robert. 1975. A Superlinearly Convergent Algorithm for Minimization Without Evaluating Derivatives. Mathematical Programming. Vol. 9, No. 1:100-117.

APPENDIX A

FORTRAN LISTING OF THE MIFFLIN ALGORITHM

```
$JOB PAGES=200,TIME=15
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER ERR
      EXTERNAL QUAD
      EXTERNAL QUAR
      EXTERNAL F
      DIMENSION X(5)
      X(1)=-1.2E0
      X(2)=1.E0
      S=.1E0
      EPS=1.E-8
      ITER=30
      N=2
      CALL MFFLN (X,N,S,F,EPS,ITER,FX,ERR)
      S=1.
      X(1)=-3.
      X(2)=-3.
      CALL MFFLN (X,N,S,QUAR,EPS,ITER,FX,ERR)
      S=.1
      X(1)=0.
      X(2)=0.
      CALL MFFLN (X,N,S,QUAD,EPS,ITER,FX,ERR)
      STOP
      END
      DOUBLE PRECISION FUNCTION F(X)
      IMPLICIT REAL*8 (A-H,L,O-Z)
      DIMENSION X(5)
      COMMON IVAL
      IVAL=IVAL+1
      F = 100.*(X(2)-X(1)**2)**2+(1.-X(1))**2
      RETURN
      END

      DOUBLE PRECISION FUNCTION QUAR (X)
      IMPLICIT REAL*8 (A-H,L,O-Z)
      DIMENSION X(5)
      COMMON IVAL
      QUAR=X(1)**4+X(2)**2-10.*X(1)
        IVAL=IVAL+1
      RETURN
      END

      DOUBLE PRECISION FUNCTION QUAD (X)
      IMPLICIT REAL*8 (A-H,L,O-Z)
      DIMENSION X(5)
      COMMON IVAL
      QUAD=(X(1)-5.)**2+(X(2)-5.)**2
      IVAL=IVAL+1
       RETURN
      END
```

```
48        SUBROUTINE MFFLN (X,N,S,F,EPS,ITER,FX,ERR)
      C
      C        PURPOSE:  TO IMPLEMENT MIFFLIN'S NON-LINEAR OPTIMIZATION
      C        METHOD
      C
      C
      C        AUTHOR:  ROD ROBISON
      C
      C**********************************************************************
      C
      C        THIS IS AN ALGORITHM FOR UNCONSTRAINED MINIMIZATION OF A
      C   REAL-VALUED FUNCTION F DEFINED ON R**N THAT DOES NO REQUIRE THE
      C   EVALUATION OF PARTIAL DERIVATIVES OF F.  THE ALGORITHM IS PARTLY
      C   AN APPROXIMATE NEWTON METHOD WHERE BOTH FIRST AND SECOND ORDER
      C   PARTIAL DERIVATIVES ARE APPROXIMATED FROM FUNCTION VALUES AND
      C   PARTLY A METHOD OF LOCATION VARIATIONS WHICH USES A SUBSET OF THESE
      C   SAME FUNCTION VALUES.  FOR ALL OF OUR CONVERGENCE RESULTS WE ASSUME
      C   F IS BOUNDED FROM BELOW AND CONTINUOUSLY DIFFERENTIABLE ON R**N.
      C
      C**********************************************************************
      C
      C
      C
      C
      C
      C   INPUT VARIABLES
      C
      C   EPS - CONVERGENCE EPSILON
      C
      C   ITER - MAXIMUM NUMBER OF ITERATIONS TO BE PERFORMED
      C
      C   ERR - RETURNED ERROR FLAG
      C          1 - MAXIMUM NUMBER OF ITERATIONS PERFORMED
      C          0 - NORMAL TERMINATION
      C
      C   S - SCALAR STEPSIZE
      C
      C   N - DIMENSION OF THE FUNCTION F TO BE MINIMIZED
      C
      C   X - THE BASE POINT OR STARTING POINT OF EACH ITERATION
      C
      C
      C   F - THE FUNCTION TO BE MINIMIZED - NOTE THIS FUNCITON MUST BE
      C       DECLARED EXTERNAL IN THE MAIN PROCEDURE
      C
      C   FX - THE RETURNED MINIMUM FUNCTION VALUE
      C
      C
      C
      C
      C   LIST OF OTHER IMPORTANT PROGRAM VARIABLES
      C
      C   L - A LOWER TRIANGULAR MATRIX USED IN THE CHOLESKY FACTORIZATION
      C
      C   E - A NON-NEGATIVE DIAGONAL MATRIX USED IN THE CHOLESKY
      C   FACTORIZATION
      C
      C   D - A POSITIVE DIAGONAL MATRIX USED IN THE CHOLESKY FACTORIZATION
      C
```

```
C     XCOR - THE CORNER POINT BEING CONSIDERED IN STEP 2
C
C     XMOV - THE MOVE POINT BEING CONSIDERED IN STEP 2
C
C     SIGMA - AN ARRAY OF DESCENT DIRECTION INDICATORS USED IN STEP 1 -
C     THE VALUES OF THE ARRAY ARE EITHER -1 OR 1
C
C     AXIS - THE AXIS POINT USED AS A CANDIDATE FOR A MOVE POINT IN STEP 1
C
C     H - THE MATRIX OF APPROXIMATE SECOND PARTIAL DERIVATIVES OF F
C
C     Y - THE MATRIX OF SEARCH DIRECTIONS DEFINED BY STEP 4
C
C     Z - A VECTOR USED AND COMPUTED IN FINDING THE BEST SEARCH DIRECTION
C     IN STEP 4
C
C     XONE - THE STARTING POINT PREVICUS TO ANY STEP
C
C     T - A TEMPORARY MATRIX USED IN CALCULATING Y IN STEP 4
C
C     RDCE - A REDUCTION FACTOR FOR A SUCCESSFUL STEP IN STEP 7.
C     EXPERIMENTATION SHOWS A REASONABLE CHOICE FOR RDCE TO BE
C     APPROXIMATELY 1.
C
C     IVAL - THE TOTAL NUMBER OF FUNCTION EVALUATIONS.  THIS SHOULD BE
C     A COMMON VARIABLE INCREMENTED BY SUBROUTINE F.
C
C     DELTA - A POSITIVE SCALAR LOWER LIMIT ON THE CHOLESKY FACTORIZED
C     MATRIX D RELATED TO THE WORD LENGTH AND CHOSEN TO AVOID NUMERICAL
C     PROBLEMS RESULTING FROM DIVISION BY ZERO.
C
C     GAMMA - AN UPPER LIMIT ON THE ELEMENTS OF THE MATRIX OF SECOND
C     PARTIALS OVER THE OPTIMIZATION REGION.
C
C     BETTA - A COMPARISON FACOTR CHOSEN SUCH THAT
C        BETTA**2 < 1./(2.*N**2*GAMMA)
C
C*******************************************************************************
C
C
C
           IMPLICIT REAL*8 (A-H,L,O-Z)
           INTEGER ERR,FLAG,P
           DIMENSION L(5,5),X(5),X1(5),DF(5),X2(5),YINV(5,5),E(5),D(5),
          1 XCOR(5),XMOV(5),SIGMA(5),AXIS(5),H(5,5),X3(5),Y(3,5),Z(5),
          2 XONE(5),T(5,5)
            DIMENSION A(5),B(5)
           COMMON IVAL
           DATA KW,KR/6,5/
           IVAL=0
           BETA=1.E-6
           ALPHA=10.
           GAMMA=1.E15
           DELTA=1.E-5
           ERR=0
           RDCE=.75
C
C   READ THE VALUES FOR N,S, AND STARTING X
C
           DO 1 K = 1,N
```

```
      1 XONE(K) = X(K)
        FX=F(X)
        DO 9999 KK1=1,ITER
C
C       STEP 1
C
        WRITE(KW,200) KK1,(X(K),K=1,N),FX
    200 FORMAT(17H1ITERATION NUMBER,I3/3H0X=,2E25.12/6H0F(X)=,E25.12)
C
C  APPROXIMATE THE FIRST PARTIALS
C
C
         DO 14 K=1,N
           X2(K)=X(K)
     14    X1(K)=X(K)
C
        SUM=0.
        DO  12 I=1,N
         X1(I)=X(I)+S
         X2(I)=X(I)-S
        A(I)=F(X1)
        B(I)=F(X2)
        DF(I)=(A(I)-B(I))/(2.*S)
         X1(I)=X(I)
        X2(I)=X(I)
C
C  CALCULATE THE GRADIENT NORM
C  AND SET BEST DESCENT VECTORS SIGMA
C
          IF(DF(I))15,15,16
     15   SIGMA(I)=1.
          GO TO 18
     16   SIGMA(I)=-1.
     18   SUM=SUM+DF(I)**2
     12 CONTINUE
        XNORM=DSQRT(SUM)
C
C  NOW FIND THE BEST AXIS POINT AXIS
C
        DO 22 I=1,N
         IF (SIGMA(I)) 20,20,21
     20  X2(I)=B(I)
         GO TO 22
     21  X2(I)=A(I)
     22  CONTINUE
        M=IMIN(X2,N)
        TEMP3=X2(M)
          DO 24 K=1,N
     24     AXIS(K)=X(K)
          AXIS(M)=X(M)+S*SIGMA(M)
        WRITE(KW,700)(AXIS(K),K=1,N),TEMP3
    700 FORMAT(12H0AXIS POINT=,2E25.12,10X,2HF=,E25.12)
C
C       STEP 2
C
C  NOW APPROXIMATE THE HESSIAN MATRIX H
C
        TEMP=GAMMA
        DO 29 J=1,N
          X1(J)=X(J)
```

```
   29   CONTINUE
      DO 25 I=1,N
        DO 26 K=1,I
          IF(I-K)28,27,28
   27 H(I,I)=(A(I)+B(I)-2.*FX)/(S*S)
          GO TO 26
   28    X1(I)=X(I)+S*SIGMA(I)
         X1(K)=X(K)+S*SIGMA(K)
        C=F(X1)
        SUM=C+FX
C
C  DEFINE THE BEST CORNER POINT
C
        IF (TEMP-C)32,32,30
   30   TEMP=C
        DO 31 JJ=1,N
         XCOR(JJ)=X1(JJ)
   31     CONTINUE
        TEMP2=C
   32   CONTINUE
      IF(SIGMA(I)) 33,33,34
   33 SUM=SUM-B(I)
      GO TO 35
   34 SUM=SUM-A(I)
   35 IF (SIGMA(K)) 36,36,37
   36 SUM=SUM-B(K)
      GO TO 38
   37 SUM=SUM-A(K)
   38 X1(I)=X(I)
      X1(K)=X(K)
         H(I,K)=SIGMA(K)*SIGMA(I)*SUM/(S*S)
         H(K,I)=H(I,K)
   26   CONTINUE
   25 CONTINUE
      WRITE(KW,701)(XCOR(J),J=1,N),TEMP2
  701 FORMAT(14HCCORNER POINT=,2E25.12,10X,2HF=,E25.12)
C
C  DEFINE THE POSSIBLE MOVE POINT
C
      IF(TEMP2-TEMP3)312,311,311
  312 DO 313 K=1,N
  313   XMOV(K)=XCOR(K)
      FMOV=TEMP2
      GO TO 40
  311 DO 314 K=1,N
  314   XMOV(K)=AXIS(K)
      FMOV=TEMP3
   40 CONTINUE
       WRITE(KW,710)(DF(J),J=1,N),XNORM
  710 FORMAT(14HOTHE GRADIENT ,2E25.12,/19HOTHE GRADIENT NORM ,E25.12)
      WRITE(KW,707)((H(J,K),K=1,N),J=1,N)
  707 FORMAT(19HOTHE HESSIAN MATRIX,2(/10X,2E25.12))
C     STEP 3
C
C****CHECK TO SEE IF H IS BOUNDED
C
      DO 315 I=1,N
        DO 316 J=1,I
        C1=DABS(H(I,J))
        IF(C1-GAMMA)316,316,317
   317 C=1.
       IF (H(I,J)) 320,321,321
   320 C=-1.
   321 H(I,J)=GAMMA*C
   316      CONTINUE
   315   CONTINUE
       CALL CHLSK (H,L,E,N,DELTA,D)
       WRITE(KW,703)((L(I,J),J=1,N),I=1,N),(D(J),J=1,N)
   703 FORMAT(8HOLMATRIX,2(/1H/,2E25.12),/10HOD MATRIX ,2(/1HO,2E25.12))
       WRITE(KW,740)(E(J),J=1,N)
   740 FORMAT(15HOTHE E MATRIX &,2E25.12)
       DO 39 I=1,N
    39   X1(I)=D(I)-E(I)
       IQ=IMIN(X1,N)
C
C   STEP 4
C
       IF(ALPHA*S-XNORM) 42,42,41
    41 IF(D(IQ)-E(IQ))60,70,70
    42 CONTINUE
C
C  CALCULATE Y1
C
       CALL TEST (L,D,E,T,N)
       DO 43 J=1,N
         T(J,J)=T(J,J)+E(J)
    43   CONTINUE
       CALL XINV(T,N,YINV)
       DO 44 J=1,N
         SUM=0.
         DO 45 K=1,N
           SUM=SUM-YINV(J,K)*DF(K)
    45     CONTINUE
         Y(1,J)=SUM
    44   CONTINUE
       P=1.
C
C  CHECK FOR E=0
C
       SUM=0.
       DO 48 K=1,N
         SUM=SUM+E(K)
    48   CONTINUE
       IF(SUM) 51,500,51
C
C CALCULATE THE NORM OF Y1
C
    51 SUM=0.
       DO 52 K=1,N
    52   SUM=SUM+Y(1,K)**2
       YNRM1=DSQRT(SUM)
C
CALCULATE A Y2 VECTOR
C
       TEMP=-YNRM1/XNORM
       DO 53 K=1,N
    53   Y(2,K)=TEMP*DF(K)
       P=2
    50 IF(D(IQ)-E(IQ))54,500,500
C
```

```
C   COMPUTE Z VECTOR
C
   54 SUM=0.
        CALL XINV (L,N,YINV)
      DO 55 K=1,N
        Z(K)=YINV(IQ,K)
   55   SUM=SUM + Z(K)**2
      C=0.
      DO 56 K=1,N
   56   C=C+Z(K)*DF(K)
      C1=1.
      IF (C) 520,500,522
  522 C1=-1.
  520 C=C1*YNRM1/DSQRT(SUM)
      DO 57 K=1,N
   57   Y(3,K)=C*Z(K)
      P=3
C
C   DEFINE THE SEARCH DIRECTION VECTOR D
C
  500 CALL TEST (L,D,E,T,N)
      DO 63 I=1,P
        C1=0.
        C2=0.
        DO 62 J=1,N
          C2=C2+Y(I,J)*DF(J)
          DO 61 K=1,N
   61       C1=C1+Y(I,J)*Y(I,K)*T(K,J)
   62   CONTINUE
      X1(I)=C1+C2/2.
   63 CONTINUE
      M=IMIN(X1,P)
      DMIN=X1(M)
      DO 64 K=1,N
   64   D(K)=Y(M,K)
      GO TO 501
C
C   COMPUTE Z
C
   60 CALL XINV (L,N,YINV)
      DO 65 K=1,N
        Z(K)=YINV(IQ,K)
   65   CONTINUE
C
C   CALCULATE Z TRANSPOSE * DF
C
      SUM=0.
      DO 66 K=1,N
   66   SUM=SUM+Z(K)*DF(K)
      C1=1.
      IF(SUM) 69,69,68
   68 C1=-1.
   69 CONTINUE
      DO 67 K=1,N
   67   D(K)=C1*Z(K)
  501 CONTINUE
C
C    STEP 5
C
      WRITE(KW,706)(D(J),J=1,N)
  706 FORMAT(27HOTHE BEST SEARCH DIRECTION ,2E25.12)
      CALL SRCH (F,X,D,FX,TT,N)
      IF(TT) 510,510,502
  502 DO 503 K=1,N
  503 X1(K)=X(K)+D(K)*TT
      FX1=F(X1)
      WRITE(KW,713)(X1(K),K=1,N),FX1
  713 FORMAT(18HOTHE SEARCH POINT ,2E25.12,10X,5HF(X)=,E25.12)
      IF(FX1-FMOV) 504,510,510
  504 DO 505 K=1,N
  505  XMOV(K)=X1(K)
      FMOV=FX1
  510 CONTINUE
C
C   STEP 6
C
      TEMP=FX
      C1=FMOV-FX
      C2=(-ALPHA*BETA*S)**2
      IF(C1-C2) 71,71,70
   71 C2=(-BETA*XNORM)**2
      IF(C1-C2)72,72,73
   72 R=S*RDCE
      GO TO 80
   73 R=S
      GO TO 90
C
C   STEP 7
C
   70 R=S/2.
C
      DO 74 K=1,N
   74   XMOV(K)=X(K)
      FMOV=FX
C
C   STEP 8
C
   80 DO 82 K=1,N
        IF(XONE(K)-X(K))82,90,82
   82   CONTINUE
      DO 84 K=1,N
   84   XONE(K)=X(K)
C
C   STEP 9
C
   90 DO 91 K=1,N
   91   X(K)=XMOV(K)
      S=R
      FX=FMOV
      WRITE(KW,210)IVAL
  210 FORMAT(21HOFUNCTION EVALUATIONS,I6)
      WRITE(KW,702)(XMOV(J),J=1,N),FMOV
  702 FORMAT(14HOMOVE POINT = ,2E25.12,6H   F = ,E25.12)
      WRITE(KW,705)S
  705 FORMAT(21HOTHE NEW STEPSIZE IS ,E25.12)
C
C   TEST FOR CONVERGENCE
C
      IF (EPS-TEMP+FMOV) 9999,9999,92
   92 IF (EPS-S) 9999,9999,93
   93 RETURN
 9999 CONTINUE
      ERR=1
      RETURN
      END
      SUBROUTINE TEST (L,D,E,H,N)
C
C---> SUBROUTINE TEST CALCULATES THE MATRIX H=LDL(T)-E FOR STEP 4
C
      IMPLICIT REAL*8 (A-H,L,O-Z)
      INTEGER R,C
      DIMENSION L(5,5)
      DIMENSION T(5,5)
      DIMENSION D(5),E(5),H(5,5)
C
      DO 10 R=1,N
        DO 5 C=1,N
          T(R,C)=0.
    5     CONTINUE
   10   CONTINUE
C
C
      DO 25 R=1,N
        DO 24 C=1,R
          T(R,C)=L(R,C)*D(C)
   24     CONTINUE
   25   CONTINUE
C
C
      DO 30 R=1,N
        DO 28 C=1,N
          SUM=0.
          DO 26 I=1,N
            SUM=SUM+T(R,I)*L(C,I)
   26       CONTINUE
          H(R,C)=SUM
   28     CONTINUE
   30   CONTINUE
      DO 40 K=1,N
   40 H(K,K)=H(K,K)-E(K)
      RETURN
      END
      SUBROUTINE XINV (LX,N,LINV)
C--->  SUBROUTINE XINV FINDS THE INVERSE OF A MATRIX L AND STORES IT IN
C        THE MATRIX LINV
      IMPLICIT REAL*8 (A-H,L,O-Z)
      DIMENSION LX(5,5)
      DIMENSION L(5,5),LINV(5,5)
C   INITIAL THE MATRIX LINV
C
C   INITIALIZE THE L MATRIX
C
      DO 31 J=1,N
        DO 30 K=1,N
          L(J,K)=LX(J,K)
          LINV(K,J)=0.
   30     CONTINUE
        LINV(J,J)=1.
   31   CONTINUE
C
C
C  CHECK FOR A ZERO DIAGONAL ELEMENT
C
      DO 40 J=1,N
        IF(L(J,J))40,41,40
   41   RETURN
   40 CONTINUE
C
C
C
C  FIND THE INVERSE BY ROW REDUCTION METHOD
      DO 20 K=1,N
        C=L(K,K)
        DO 5 J=1,N
          LINV(K,J)=LINV(K,J)/C
          L(K,J)=L(K,J)/C
    5     CONTINUE
        DO 8 J=1,N
          IF(J-K) 9,8,9
    9     C=L(J,K)
          DO 10 I=1,N
            L(J,I)=L(J,I)-L(K,I)*C
            LINV(J,I)=LINV(J,I)-LINV(K,I)*C
   10       CONTINUE
    8     CONTINUE
   20   CONTINUE
      RETURN
       END
```

```
         SUBROUTINE CHLSK(H,L,E,N,DELTA,D)
C---> SUBROUTINE CHLSK DOES A MODIFIED CHOLESKY FACTORIZATION FINDING A
C           MATRIX L,D,E SUCH THAT LDL(T)-E=H
         IMPLICIT REAL*8 (A-H,L,O-Z)
         INTEGER R
         DIMENSION L(5,5)
         DIMENSION D(5),E(5),C(5,5),H(5,5)
         GAMMA=DELTA
         DO 2 J=1,N
           DO 1 K=1,N
             IF (GAMMA-H(J,K)) 3,1,1
    3        GAMMA=H(J,K)
    1        CONTINUE
    2      CONTINUE
C
C   INITIALIZE MATRIX L
           DO 5 M=1,N
            DO 6 I=M,N
    6         L(M,I)=0.
    5     L(M,M)=1.
C
         DO 100 J=1,N
C   CONPUTE THE VALUES FOR MATRIX L
           K=J-1
            IF(K)10,20,10
   10     DO 12 R=1,K
   12         L(J,R)=C(J,R)/D(R)
C   COMPUTE VALUES FOR MATRIX C
   20     DO 22 I=J,N
              SUM=0.
              IF(K)26,22,26
   26         DO 28 R=1,K
   28           SUM=SUM+C(I,R)*L(J,R)
   22         C(I,J)=H(I,J)-SUM
C   COMPUTE THE DIAGONAL ELEMENT OF D
            AMAX=DELTA
            AC=DABS(C(J,J))
            IF(DELTA-AC)30,32,32
   30       AMAX=AC
   32       K=J+1
            IF(K-N)34,34,40
   34  DO 36 I=K,N
              AC=1./GAMMA*DABS(C(I,J))**2
              IF(AMAX-AC)38,36,36
   38         AMAX=AC
   36         CONTINUE
   40     D(J)=AMAX
            E(J)=D(J)-C(J,J)
  100     CONTINUE
         RETURN
         END
```

```
      FUNCTION IMIN(X,N)
C---> FUNCTION IMIN FINDS THE SUBSCRIPT OF THE MIN VALUE IN THE ARRAY X
      IMPLICIT REAL*8 (A-H,L,O-Z)
      DIMENSION X(5)
      LOW=1
      DO 10 K=1,N
      IF(X(LOW)-X(K))10,10,9
    9 LOW=K
   10 CONTINUE
      IMIN=LOW
      RETURN
      END
      SUBROUTINE SRCH (F,XX,D,FX,T,N)
C---> SUBROUTINE SRCH DOES A ONE VARIBLE MINIMIZATION ON T IN F(X+TD)
C         BY FITING A PARABOLA TO THE CURVE AND THEN MINIMIZING THE PARABOLA
      IMPLICIT REAL*8 (A-H,L,O-Z)
      DIMENSION XX(5),X(5),Y(5),X1(5),D(5)
      X(1)=0.
      Y(1)=FX
      T=.5
      DO 12 I=2,3
        X(I)=T
        DO 10 J=1,N
          X1(J)=XX(J)+T*D(J)
   10     CONTINUE
        Y(I)=F(X1)
        T=T+.5
   12   CONTINUE
C
      CALL FIT (X,Y,A,B,C)
      IF(A)99,99,52
   52 T=-B/(2.*A)
      RETURN
   99 T=0.
      RETURN
      END
      SUBROUTINE FIT (X,Y,A,B,C)
      IMPLICIT REAL*8 (A-H,L,O-Z)
      DIMENSION X(5),Y(5)
C--> THIS SUBROUTINE FITS A PARABOLA TO THREE SETS OF POINTS (X,Y) AND
C  RETURNS THE VALUES OF A,B,C FOR A PARABOLA OF THE FORM P(X)=
C                                                    A*X**2 + B*X + C
C
      A1=Y(1)
      A2=(Y(2)-A1)/(X(2)-X(1))
      A3=(Y(3)-A1-(X(3)-X(1))*A2)/((X(3)-X(1))*(X(3)-X(2)))
      B=A2-A3*X(1)-A3*X(2)
      C=A1-A2*X(1)+A3*X(1)*X(2)
      A=A3
      RETURN
      END

$ENTRY
```

APPENDIX B

FORTRAN LISTING OF THE DFP ALGORITHM

```
$JOB TIME=60,PAGES=50
      DOUBLE PRECISION X,G,F,H
      EXTERNAL F1
      EXTERNAL F2
      EXTERNAL ROSBK
      DIMENSION X(2),G(2),H(9)
      COMMON KOUNT
      DATA KW/6/
      KOUNT=0
      N=2
      EST=0.
      EPS=10.D-10
      LIMIT=20
      X(1)=-1.2
      X(2)=-1.
      CALL DFMFP (ROSBK,N,X,F,G,EST,EPS,LIMIT,IER,H)
      WRITE(KW,10)F,KOUNT,X
   10 FORMAT('0A MINIMUM OF ',E25.12,/' WAS FOUND AFTER ',I10,/
     1 ' FUNCTION EVALUATIONS WITH X=',2E25.12)
C
C
      KOUNT=0
      X(1)=-3.
      X(2)=-3.
      CALL DFMFP (F1,N,X,F,G,EST,EPS,LIMIT,IER,H)
      WRITE(KW,10)F,KOUNT,X
C
C
      X(1)=0.
      X(2)=0.
      KOUNT=0
      CALL DFMFP (F2,N,X,F,G,EST,EPS,LIMIT,IER,H)
      WRITE(KW,10)F,KOUNT,X
      STOP
      END

      SUBROUTINE ROSBK (N,ARG,VAL,GRAD)
      DOUBLE PRECISION X,Y
      DOUBLE PRECISION ARG,VAL,GRAD
      DIMENSION ARG(N),GRAD(N)
      COMMON KOUNT
      KOUNT=KOUNT+1
      X=ARG(1)
      Y=ARG(2)
      VAL=100.*(Y-X**2)**2 + (1.-X)**2
      GRAD(1)=-400.*X*(Y-X**2)-2.*(1.-X)
      GRAD(2)=200.*(Y-X**2)
      WRITE(6,100)KOUNT,VAL,X,Y
  100 FORMAT('0KOUNT=',I5,10X,'F(X)=',E25.12,10X,'X=',2E25.12)
      RETURN
      END

      SUBROUTINE F1(N,ARG,VAL,GRAD)
      DOUBLE PRECISION X,Y,ARG,VAL,GRAD
      DIMENSION ARG(N),GRAD(N)
      COMMON KOUNT
      KOUNT=KOUNT+1
      X=ARG(1)
      Y=ARG(2)
      VAL = X**4 + Y**2 + 10.*X
      GRAD(1)= 4.*X**3 + 10.
      GRAD(2)=  2.*Y
      WRITE(6,100)KOUNT,VAL,X,Y
  100 FORMAT('OKOUNT=',I5,10X,'F(X)=',E25.12,10X,'X=',2E25.12)
      RETURN
      END

      SUBROUTINE F2(N,ARG,VAL,GRAD)
      DOUBLE PRECISION X,Y,ARG,VAL,GRAD
      DIMENSION ARG(N),GRAD(N)
      COMMON KOUNT
      KOUNT=KOUNT+1
      X=ARG(1)
      Y=ARG(2)
      VAL=(X-5.)**2 + (Y-5.)**2
      GRAD(1)=2.*(X-5.)
      GRAD(2)=2.*(Y-5.)
      WRITE(6,100)KOUNT,VAL,X,Y
  100 FORMAT('OKOUNT=',I5,10X,'F(X)=',E25.12,10X,'X=',2E25.12)
      RETURN
      END
C                                                                       DFMF  10
C     ..................................................................DFMF  20
C                                                                       DFMF  30
C        SUBROUTINE DFMFP                                               DFMF  40
C                                                                       DFMF  50
C        PURPOSE                                                        DFMF  60
C           TO FIND A LOCAL MINIMUM OF A FUNCTION OF SEVERAL VARIABLES  DFMF  70
C           BY THE METHOD OF FLETCHER AND POWELL                        DFMF  80
C                                                                       DFMF  90
C        USAGE                                                          DFMF 100
C           CALL DFMFP(FUNCT,N,X,F,G,EST,EPS,LIMIT,IER,H)               DFMF 110
C                                                                       DFMF 120
C        DESCRIPTION OF PARAMETERS                                      DFMF 130
C           FUNCT  - USER-WRITTEN SUBROUTINE CONCERNING THE FUNCTION TO DFMF 140
C                    BE MINIMIZED. IT MUST BE OF THE FORM               DFMF 150
C                    SUBROUTINE FUNCT(N,ARG,VAL,GRAD)                   DFMF 160
C                    AND MUST SERVE THE FOLLOWING PURPOSE               DFMF 170
C                    FOR EACH N-DIMENSIONAL ARGUMENT VECTOR  ARG,       DFMF 180
C                    FUNCTION VALUE AND GRADIENT VECTOR MUST BE COMPUTEDDFMF 190
C                    AND, ON RETURN, STORED IN VAL AND GRAD RESPECTIVELYDFMF 200
C                    ARG,VAL AND GRAD MUST BE OF DOUBLE PRECISION.      DFMF 210
C           N      - NUMBER OF VARIABLES                                DFMF 220
C           X      - VECTOR OF DIMENSION N CONTAINING THE INITIAL       DFMF 230
C                    ARGUMENT WHERE THE ITERATION STARTS. ON RETURN,    DFMF 240
C                    X HOLDS THE ARGUMENT CORRESPONDING TO THE          DFMF 250
C                    COMPUTED MINIMUM FUNCTION VALUE                    DFMF 260
C                    DOUBLE PRECISION VECTOR.                           DFMF 270
C           F      - SINGLE VARIABLE CONTAINING THE MINIMUM FUNCTION    DFMF 280
C                    VALUE ON RETURN, I.E. F=F(X).                      DFMF 290
C                    DOUBLE PRECISION VARIABLE.                         DFMF 300
C           G      - VECTOR OF DIMENSION N CONTAINING THE GRADIENT      DFMF 310
C                    VECTOR CORRESPONDING TO THE MINIMUM ON RETURN,     DFMF 320
C                    I.E. G=G(X).                                       DFMF 330
C                    DOUBLE PRECISION VECTOR.                           DFMF 340
C           EST    - IS AN ESTIMATE OF THE MINIMUM FUNCTION VALUE.      DFMF 350
C                    SINGLE PRECISION VARIABLE.                         DFMF 360
C           EPS    - TESTVALUE REPRESENTING THE EXPECTED ABSOLUTE ERROR.DFMF 370
C                    A REASONABLE CHOICE IS 10**(-16), I.E.             DFMF 380
C                    SOMEWHAT GREATER THAN 10**(-D), WHERE D IS THE     DFMF 390
```

```
C                    NUMBER OF SIGNIFICANT DIGITS IN FLOATING POINT      DFMF 400
C                    REPRESENTATION.                                     DFMF 410
C                    SINGLE PRECISION VARIABLE.                          DFMF 420
C           LIMIT  - MAXIMUM NUMBER OF ITERATIONS.                       DFMF 430
C           IER    - ERROR PARAMETER                                     DFMF 440
C                    IER = 0 MEANS CONVERGENCE WAS OBTAINED              DFMF 450
C                    IER = 1 MEANS NO CONVERGENCE IN LIMIT ITERATIONS    DFMF 460
C                    IER =-1 MEANS ERRORS IN GRADIENT CALCULATION        DFMF 470
C                    IER = 2 MEANS LINEAR SEARCH TECHNIQUE INDICATES     DFMF 480
C                    IT IS LIKELY THAT THERE EXISTS NO MINIMUM.          DFMF 490
C           H      - WORKING STORAGE OF DIMENSION N*(N+7)/2.             DFMF 500
C                    DOUBLE PRECISION ARRAY.                             DFMF 510
C                                                                        DFMF 520
C        REMARKS                                                         DFMF 530
C            I) THE SUBROUTINE NAME REPLACING THE DUMMY ARGUMENT  FUNCT DFMF 540
C               MUST BE DECLARED AS EXTERNAL IN THE CALLING PROGRAM.    DFMF 550
C           II) IER IS SET TO 2 IF , STEPPING IN ONE OF THE COMPUTED     DFMF 560
C               DIRECTIONS, THE FUNCTION WILL NEVER INCREASE WITHIN      DFMF 570
C               A TOLERABLE RANGE OF ARGUMENT.                           DFMF 580
C               IER = 2 MAY OCCUR ALSO IF THE INTERVAL WHERE F           DFMF 590
C               INCREASES IS SMALL AND THE INITIAL ARGUMENT WAS          DFMF 600
C               RELATIVELY FAR AWAY FROM THE MINIMUM SUCH THAT THE       DFMF 610
C               MINIMUM WAS OVERLEAPED. THIS IS DUE TO THE SEARCH        DFMF 620
C               TECHNIQUE WHICH DOUBLES THE STEPSIZE UNTIL A POINT       DFMF 630
C               IS FOUND WHERE THE FUNCTION INCREASES.                   DFMF 640
C                                                                        DFMF 650
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED                   DFMF 660
C           FUNCT                                                        DFMF 670
C                                                                        DFMF 680
C        METHOD                                                          DFMF 690
C           THE METHOD IS DESCRIBED IN THE FOLLOWING ARTICLE             DFMF 700
C           R. FLETCHER AND M.J.D. POWELL, A RAPID DESCENT METHOD FOR    DFMF 710
C           MINIMIZATION,                                                DFMF 720
C           COMPUTER JOURNAL VOL.6, ISS. 2, 1963, PP.163-168.            DFMF 730
C                                                                        DFMF 740
C     ..................................................................DFMF 750
C                                                                        DFMF 760

      SUBROUTINE DFMFP(FUNCT,N,X,F,G,EST,EPS,LIMIT,IER,H)                DFMF 770
C                                                                        DFMF 780
C        DIMENSIONED DUMMY VARIABLES                                     DFMF 790
      DIMENSION H(9),X(N),G(N)                                           DFMF 800
      DOUBLE PRECISION X,F,FX,FY,OLDF,HNRM,GNRM,H,G,DX,DY,ALFA,DALFA,    DFMF 810
     1AMBDA,T,Z,W,DSQRT,DABS,DMAX1                                       DFMF 820
C                                                                        DFMF 830
C        COMPUTE FUNCTION VALUE AND GRADIENT VECTOR FOR INITIAL ARGUMENTDFMF 840
      CALL FUNCT(N,X,F,G)                                                DFMF 850
C                                                                        DFMF 860
C        RESET ITERATION COUNTER AND GENERATE IDENTITY MATRIX            DFMF 870
      IER=0                                                              DFMF 880
      KOUNT=0                                                            DFMF 890
      N2=N+N                                                             DFMF 900
      N3=N2+N                                                            DFMF 910
      N31=N3+1                                                           DFMF 920
    1 K=N31                                                              DFMF 930
      DO 4 J=1,N                                                         DFMF 940
      H(K)=1.D0                                                          DFMF 950
      NJ=N-J                                                             DFMF 960
      IF(NJ)5,5,2                                                        DFMF 970
    2 DO 3 L=1,NJ                                                        DFMF 980
```

```
      KL=K+L                                                            DFMF 990
    3 H(KL)=0.D0                                                        DFMF1000
    4 K=KL+1                                                            DFMF1010
C                                                                       DFMF1020
C        START ITERATION LOOP                                           DFMF1030
    5 KOUNT=KOUNT +1                                                    DFMF1040
      WRITE(6,1000)
 1000 FORMAT(1H0)
C                                                                       DFMF1050
C        SAVE FUNCTION VALUE, ARGUMENT VECTOR AND GRADIENT VECTOR       DFMF1060
      OLDF=F                                                            DFMF1070
      DO 9 J=1,N                                                        DFMF1080
      K=N+J                                                             DFMF1090
      H(K)=G(J)                                                         DFMF1100
      K=K+N                                                             DFMF1110
      H(K)=X(J)                                                         DFMF1120
C                                                                       DFMF1130
C        DETERMINE DIRECTION VECTOR H                                   DFMF1140
      K=J+N3                                                            DFMF1150
      T=0.D0                                                            DFMF1160
      DO 8 L=1,N                                                        DFMF1170
      T=T-G(L)*H(K)                                                     DFMF1180
      IF(L-J)6,7,7                                                      DFMF1190
    6 K=K+N-L                                                           DFMF1200
      GO TO 8                                                           DFMF1210
    7 K=K+1                                                             DFMF1220
    8 CONTINUE                                                          DFMF1230
    9 H(J)=T                                                            DFMF1240
C                                                                       DFMF1250
C        CHECK WHETHER FUNCTION WILL DECREASE STEPPING ALONG H.         DFMF1260
      DY=0.D0                                                           DFMF1270
      HNRM=0.D0                                                         DFMF1280
      GNRM=0.D0                                                         DFMF1290
C                                                                       DFMF1300
C        CALCULATE DIRECTIONAL DERIVATIVE AND TESTVALUES FOR DIRECTION  DFMF1310
C        VECTOR H AND GRADIENT VECTOR G.                                DFMF1320
      DO 10 J=1,N                                                       DFMF1330
      HNRM=HNRM+DABS(H(J))                                              DFMF1340
      GNRM=GNRM+DABS(G(J))                                              DFMF1350
   10 DY=DY+H(J)*G(J)                                                   DFMF1360
C                                                                       DFMF1370
C        REPEAT SEARCH IN DIRECTION OF STEEPEST DESCENT IF DIRECTIONAL  DFMF1380
C        DERIVATIVE APPEARS TO BE POSITIVE OR ZERO.                     DFMF1390
      IF(DY)11,51,51                                                    DFMF1400
C                                                                       DFMF1410
C        REPEAT SEARCH IN DIRECTION OF STEEPEST DESCENT IF DIRECTION    DFMF1420
C        VECTOR H IS SMALL COMPARED TO GRADIENT VECTOR G.               DFMF1430
   11 IF(HNRM/GNRM-EPS)51,51,12                                         DFMF1440
C                                                                       DFMF1450
C        SEARCH MINIMUM ALONG DIRECTION H                               DFMF1460
C                                                                       DFMF1470
C        SEARCH ALONG H FOR POSITIVE DIRECTIONAL DERIVATIVE             DFMF1480
   12 FY=F                                                              DFMF1490
      ALFA=2.D0*(EST-F)/DY                                              DFMF1500
      AMBDA=1.D0                                                        DFMF1510
C                                                                       DFMF1520
C        USE ESTIMATE FOR STEPSIZE ONLY IF IT IS POSITIVE AND LESS THAN DFMF1530
C        1. OTHERWISE TAKE 1. AS STEPSIZE                               DFMF1540
      IF(ALFA)15,15,13                                                  DFMF1550
   13 IF(ALFA-AMBDA)14,15,15                                            DFMF1560
   14 AMBDA=ALFA                                                        DFMF1570
   15 ALFA=0.D0                                                         DFMF1580
C                                                                       DFMF1590
C        SAVE FUNCTION AND DERIVATIVE VALUES FOR OLD ARGUMENT           DFMF1600
   16 FX=FY                                                             DFMF1610
      DX=DY                                                             DFMF1620
C                                                                       DFMF1630
C        STEP ARGUMENT ALONG H                                          DFMF1640
      DO 17 I=1,N                                                       DFMF1650
   17 X(I)=X(I)+AMBDA*H(I)                                              DFMF1660
C                                                                       DFMF1670
C        COMPUTE FUNCTION VALUE AND GRADIENT FOR NEW ARGUMENT           DFMF1680
      CALL FUNCT(N,X,F,G)                                               DFMF1690
      FY=F                                                              DFMF1700
C                                                                       DFMF1710
C        COMPUTE DIRECTIONAL DERIVATIVE DY FOR NEW ARGUMENT.  TERMINATE DFMF1720
C        SEARCH, IF DY IS POSITIVE. IF DY IS ZERO THE MINIMUM IS FOUND  DFMF1730
      DY=0.D0                                                           DFMF1740
      DO 18 I=1,N                                                       DFMF1750
   18 DY=DY+G(I)*H(I)                                                   DFMF1760
      IF(DY)19,36,22                                                    DFMF1770
C                                                                       DFMF1780
C        TERMINATE SEARCH ALSO IF THE FUNCTION VALUE INDICATES THAT     DFMF1790
C        A MINIMUM HAS BEEN PASSED                                      DFMF1800
   19 IF(FY-FX)20,22,22                                                 DFMF1810
C                                                                       DFMF1820
C        REPEAT SEARCH AND DOUBLE STEPSIZE FOR FURTHER SEARCHES         DFMF1830
   20 AMBDA=AMBDA+ALFA                                                  DFMF1840
      ALFA=AMBDA                                                        DFMF1850
C        END OF SEARCH LOOP                                             DFMF1860
C                                                                       DFMF1870
C        TERMINATE IF THE CHANGE IN ARGUMENT GETS VERY LARGE            DFMF1880
      IF(HNRM*AMBDA-1.D10)16,16,21                                      DFMF1890
C                                                                       DFMF1900
C        LINEAR SEARCH TECHNIQUE INDICATES THAT NO MINIMUM EXISTS       DFMF1910
   21 IER=2                                                             DFMF1920
      RETURN                                                            DFMF1930
C                                                                       DFMF1940
C        INTERPOLATE CUBICALLY IN THE INTERVAL DEFINED BY THE SEARCH    DFMF1950
C        ABOVE AND COMPUTE THE ARGUMENT X FOR WHICH THE INTERPOLATION   DFMF1960
C        POLYNOMIAL IS MINIMIZED                                        DFMF1970
   22 T=0.D0                                                            DFMF1980
   23 IF(AMBDA)24,36,24                                                 DFMF1990
   24 Z=3.D0*(FX-FY)/AMBDA+DX+DY                                        DFMF2000
      ALFA=DMAX1(DABS(Z),DABS(DX),DABS(DY))                             DFMF2010
      DALFA=Z/ALFA                                                      DFMF2020
      DALFA=DALFA*DALFA-DX/ALFA*DY/ALFA                                 DFMF2030
      IF(DALFA)51,25,25                                                 DFMF2040
   25 W=ALFA*DSQRT(DALFA)                                               DFMF2050
      ALFA=DY-DX+W+W                                                    DFMF2060
      IF(ALFA) 250,251,250                                              DFMF2061
  250 ALFA=(DY-Z+W)/ALFA                                                DFMF2062
      GO TO 252                                                         DFMF2063
  251 ALFA=(Z+DY-W)/(Z+DX+Z+DY)                                         DFMF2064
  252 ALFA=ALFA*AMBDA                                                   DFMF2065
      DO 26 I=1,N                                                       DFMF2070
   26 X(I)=X(I)+(T-ALFA)*H(I)                                           DFMF2080
C                                                                       DFMF2090
C        TERMINATE, IF THE VALUE OF THE ACTUAL FUNCTION AT X IS LESS    DFMF2100
C        THAN THE FUNCTION VALUES AT THE INTERVAL ENDS. OTHERWISE REDUCEDFMF2110
```

```
C        THE INTERVAL BY CHOOSING ONE END-POINT EQUAL TO X AND REPEAT   DFMF2120
C        THE INTERPOLATION.  WHICH END-POINT IS CHOOSEN DEPENDS ON THE  DFMF2130
C        VALUE OF THE FUNCTION AND ITS GRADIENT AT X                    DFMF2140
C                                                                       DFMF2150
      CALL FUNCT(N,X,F,G)                                               DFMF2160
      IF(F-FX)27,27,28                                                  DFMF2170
   27 IF(F-FY)36,36,28                                                  DFMF2180
   28 DALFA=0.D0                                                        DFMF2190
      DO 29 I=1,N                                                       DFMF2200
   29 DALFA=DALFA+G(I)*H(I)                                             DFMF2210
      IF(DALFA)30,33,33                                                 DFMF2220
   30 IF(F-FX)32,31,33                                                  DFMF2230
   31 IF(DX-DALFA)32,36,32                                              DFMF2240
   32 FX=F                                                              DFMF2250
      DX=DALFA                                                          DFMF2260
      T=ALFA                                                            DFMF2270
      AMBDA=ALFA                                                        DFMF2280
      GO TO 23                                                          DFMF2290
   33 IF(FY-F)35,34,35                                                  DFMF2300
   34 IF(DY-DALFA)35,36,35                                              DFMF2310
   35 FY=F                                                              DFMF2320
      DY=DALFA                                                          DFMF2330
      AMBDA=AMBDA-ALFA                                                  DFMF2340
      GO TO 22                                                          DFMF2350
C                                                                       DFMF2360
C        TERMINATE, IF FUNCTION HAS NOT DECREASED DURING LAST ITERATION DFMF2370
   36 IF(OLDF-F+EPS)51,38,38                                            DFMF2380
C                                                                       DFMF2390
C        COMPUTE DIFFERENCE VECTORS OF ARGUMENT AND GRADIENT FROM       DFMF2400
C        TWO CONSECUTIVE ITERATIONS                                     DFMF2410
   38 DO 37 J=1,N                                                       DFMF2420
      K=N+J                                                             DFMF2430
      H(K)=G(J)-H(K)                                                    DFMF2440
      K=N+K                                                             DFMF2450
   37 H(K)=X(J)-H(K)                                                    DFMF2460
C                                                                       DFMF2470
C        TEST LENGTH OF ARGUMENT DIFFERENCE VECTOR AND DIRECTION VECTOR DFMF2480
C        IF AT LEAST N ITERATIONS HAVE BEEN EXECUTED. TERMINATE, IF     DFMF2490
C        BOTH ARE LESS THAN  EPS                                        DFMF2500
      IER=0                                                             DFMF2510
      IF(KOUNT-N)42,39,39                                               DFMF2520
   39 T=0.D0                                                            DFMF2530
      Z=0.D0                                                            DFMF2540
      DO 40 J=1,N                                                       DFMF2550
      K=N+J                                                             DFMF2560
      W=H(K)                                                            DFMF2570
      K=K+N                                                             DFMF2580
      T=T+DABS(H(K))                                                    DFMF2590
   40 Z=Z+W*H(K)                                                        DFMF2600
      IF(HNRM-EPS)41,41,42                                              DFMF2610
   41 IF(T-EPS)56,56,42                                                 DFMF2620
C                                                                       DFMF2630
C        TERMINATE, IF NUMBER OF ITERATIONS WOULD EXCEED  LIMIT         DFMF2640
   42 IF(KOUNT-LIMIT)43,50,50                                           DFMF2650
C                                                                       DFMF2660
C        PREPARE UPDATING OF MATRIX H                                   DFMF2670
   43 ALFA=0.D0                                                         DFMF2680
      DO 47 J=1,N                                                       DFMF2690
      K=J+N3                                                            DFMF2700
      W=0.D0                                                            DFMF2710
      DO 46 L=1,N                                                       DFMF2720
      KL=N+L                                                            DFMF2730
      W=W+H(KL)*H(K)                                                    DFMF2740
      IF(L-J)44,45,45                                                   DFMF2750
   44 K=K+N-L                                                           DFMF2760
      GO TO 46                                                          DFMF2770
   45 K=K+1                                                             DFMF2780
   46 CONTINUE                                                          DFMF2790
      K=N+J                                                             DFMF2800
      ALFA=ALFA+W*H(K)                                                  DFMF2810
   47 H(J)=W                                                            DFMF2820
C                                                                       DFMF2830
C        REPEAT SEARCH IN DIRECTION OF STEEPEST DESCENT IF RESULTS      DFMF2840
C        ARE NOT SATISFACTORY                                           DFMF2850
      IF(Z*ALFA)48,1,48                                                 DFMF2860
C                                                                       DFMF2870
C        UPDATE MATRIX H                                                DFMF2880
   48 K=N31                                                             DFMF2890
      DO 49 L=1,N                                                       DFMF2900
      KL=N2+L                                                           DFMF2910
      DO 49 J=L,N                                                       DFMF2920
      NJ=N2+J                                                           DFMF2930
      H(K)=H(K)+H(KL)*H(NJ)/Z-H(L)*H(J)/ALFA                            DFMF2940
   49 K=K+1                                                             DFMF2950
      GO TO 5                                                           DFMF2960
C        END OF ITERATION LOOP                                          DFMF2970
C                                                                       DFMF2980
C        NO CONVERGENCE AFTER  LIMIT  ITERATIONS                        DFMF2990
   50 IER=1                                                             DFMF3000
      RETURN                                                            DFMF3010
C                                                                       DFMF3020
C        RESTORE OLD VALUES OF FUNCTION AND ARGUMENTS                   DFMF3030
   51 DO 52 J=1,N                                                       DFMF3040
      K=N2+J                                                            DFMF3050
   52 X(J)=H(K)                                                         DFMF3060
      CALL FUNCT(N,X,F,G)                                               DFMF3070
C                                                                       DFMF3080
C        REPEAT SEARCH IN DIRECTION OF STEEPEST DESCENT IF DERIVATIVE   DFMF3090
C        FAILS TO BE SUFFICIENTLY SMALL                                 DFMF3100
      IF(GNRM-EPS)55,55,53                                              DFMF3110
C                                                                       DFMF3120
C        TEST FOR REPEATED FAILURE OF ITERATION                         DFMF3130
   53 IF(IER)56,54,54                                                   DFMF3140
   54 IER=-1                                                            DFMF3150
      GOTO 1                                                            DFMF3160
   55 IER=0                                                             DFMF3170
   56 RETURN                                                            DFMF3180
      END                                                               DFMF3190
```

VITA

Rodney Wayne Robison

Candidate for the Degree of

Master of Science

Thesis: A COMPUTER IMPLEMENTATION AND TEST OF MIFFLIN'S ALGORITHM FOR NONLINEAR OPTIMIZATION

Major Field: Computing and Information Sciences

Biographical:

Personal Data: Born in Ottawa Kansas, February 4, 1952, the son of John and Martha Robison.

Education: Graduate from Chickasha High School, Chickasha, Oklahoma in May, 1970; received Bachelor of Science degree in Mathematics from Oklahoma State University, Stillwater, Oklahoma, in May 1975; completed requirements for the Master of Science degree at Oklahoma State University, Stillwater, Oklahoma, in July, 1977.

Professional Experience: Scientific Programmer for the Agronomy Department, Soil Morphology Laboratory, Oklahoma State University, Stillwater, Oklahoma, January, 1974 - August, 1976; graduate teaching assistant, Oklahoma State University, Computing and Information Sciences Department, August, 1976-May, 1977.