A COMPUTER ALGORITHM FOR CONSTRAINED MINIMIZATION

J.P. Indusi

July, 1970

USE-1-1
College of Engineering Report #175

- INTRODUCTION -

In this report we describe an algorithm which minimizes a real valued objective function f of n variables $x^1, x^2, \ldots, x^n$ subject to constraints of the form

(1 - 1) $$g(x^1, x^2, \ldots, x^n) = 0.$$

More generally there may be several constraints of the type (1 - 1) and indeed there may be some of the more general inequality form

(1 - 2) $$g(x^1, x^2, \ldots, x^n) \leq 0.$$

The objective function f and the constraints must be differentiable functions of the n variables $x^1, x^2, \ldots, x^n$. However, they need not be quadratic or convex but may be any general non-linear functions.

This report is divided into two parts, the first being a description of an algorithm. The second part will consist of explicit instructions on how to use a Fortran version of the algorithm. This deck will be available from the author or from the Computing Center, S.U.N.Y. at Stony Brook. The computer program is limited to problems of moderate size but otherwise seems capable of handling examples of considerable complexity. We feel that this is perhaps the most effective computer algorithm available today for the constrained minimization of non-linear functions.

We wish to extend sincere thanks to Ivan L. Johnson of the NASA Manned Spacecraft Center in Houston, Texas, for making available to us a computer program which already incorporates many of the features of the present deck.

This report would not have been possible without the continuous interest and support of Professor E.J. Beltrami of the State University of New York at Stony Brook. Much of the text of this report and the development of the Fortran program are the results of his efforts.

TABLE OF CONTENTS

- THE ALGORITHM -

Before beginning a discussion of the method of solving the general non-linear programming problem we establish some notation. The gradient of f will be denoted by ∇f. In general x will denote an n dimensional column vector and x_i will denote the ith such vector in a sequence of vectors. Similarly, the ith matrix in a sequence of positive-definite matrices will be denoted by H_i. The constraint equations will be indexed as g_j with g standing for the m element vector with components g_j. The n by m Jacobian of g is written as G where the columns of G are the gradients $\nabla g_1, \nabla g_2, \ldots, \nabla g_m$.

Without loss of generality we may assume that all the constraints are of the equality type [equation (1-1)] by means of the following device: Constraints of the inequality type [equation (1-2)] are equivalent to equality constraints given by

$$(2\text{-}1) \qquad g_j^2 (x^1, x^2, \ldots, x^n)\, u_j(x^1, x^2, \ldots, x^n) = 0$$

where

$$(2\text{-}2) \qquad u_j(x^1, x^2, \ldots, x^n) = \begin{cases} 0 \text{ if } g_j(x^1, x^2, \ldots, x^n) \leq 0 \\ 1 \text{ if } g_j(x^1, x^2, \ldots, x^n) > 0 \end{cases}$$

Under suitably mild hypothesis one can show that the minimum of f subject to the constraints $g_j = 0 \quad j = 1,2,\ldots,m$ can be found by solving the sequence of unconstrained problems in which the augmented functions f_k are minimized where

(2-3) $f_K = f + (1/2)g^T Kg$ for $K \to \infty$.

Here K is an m by m diagonal matrix with positive components k_j along the diagonal and $K \to \infty$ means that the k_j each increase without bound. g^T denotes the m constraints g_j arranged as a row vector since the T superscript indicates transpose. The function f_K is minimized for each K, each K matrix being larger than the previous one. It is intuitively reasonable that for K large the minimum of f_K requires that $(1/2)g^T Kg$ be at most moderate in size. But this means that in the limit we expect g to be zero, which is indeed what happens. This method is due to R. Courant and is called the penalty argument since the effect of letting $K \to \infty$ is that it penalizes the functions f_K for constraint violations.

From (2-3) one may derive the well known Lagrange and Kuhn-Tucker multiplier rules. In fact, from (2-3) one obtains

(2-4) $\nabla f_K = \nabla f + Kg \nabla g$

and at the minimum of the unconstrained f_K we have

(2-5) $\nabla f_K = 0 = \nabla f + Kg \nabla g.$

Passing to the limit in (2-5) we find that Kg tends to the multipliers λ and one then obtains the familiar rule

(2-6) $\nabla f = G\lambda.$

We have only sketched the penalty argument and multiplier rules here. The reader is referred to Chapter 2 of the book [1] where a thorough discussion

of the penalty argument is given along with concise statements and proofs of the above mentioned multiplier rules.

We turn now to a discussion of how one can minimize the functions f_K as given in(2-3). Computationally there are several problems we are confronted with. We cannot on a computer really allow K to increase without bound. Also in practice there may be some difficulty in choosing initial values for the penalty constants k_i. An up-to-date discussion of the various numerical difficulties encountered in the solution of non-linear problems is given in [2] and a discussion of how to choose the initial K matrix may be found in [1].

To minimize f_K, let us assume that we have made a judicious choice for the initial penalty matrix K and one has a starting guess $x_o = (x^1, x^2, \ldots, x^n)$ for the location of the minimum. We then form the augmented function f_K as given by (2-3) and carry out an unconstrained minimization according to one of several options, all of which are variants of the Davidon algorithm as modified by Fletcher and Powell [3]. This method has theoretically exact convergence for quadratic functions in a finite number of steps and in practice it achieves rapid convergence once a neighborhood of the minimum is attained. The iterative process proceeds according to the following equation:

$$(2\text{-}7) \qquad x_{i+1} = x_i - \alpha_i H_i \nabla f_K(x_i).$$

Here H_i is a positive definite n by n matrix generated by a suitable rank two correction to H_{i-1} and H_o is chosen to be the identity matrix I.

The positive scalars α_i are chosen so as to minimize f_K evaluated along the direction $-H_i \nabla f_K(x_i)$ starting at the point x_i. Fletcher and Powell in [3] establish a number of interesting properties of the algorithm. They prove that for a quadratic function the minimum is found in n steps. In addition, the H matrix tends to the inverse of the Hessian of f at the minimum. That is, if we define a matrix F by

$$F_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j} , \tag{2-8}$$

then as the procedure converges H tends to F^{-1} evaluated at the minimum. Later on in this report we will show how this converged H matrix will be used in a refinement phase based on Newton's method.

Let $S_i = - H_i \nabla f_{K,i}$ where for simplicity $\nabla f_{K,i}$ is the gradient of f_K evaluated at the point x_i. Also let $p_i = \lambda_i S_i$ where λ_i is chosen so that $f_K(x_i + \alpha_i S_i)$ is minimized and let $Y_i = \nabla f_{K,\, i+1} - \nabla f_{K,i}$. The equation for updating the H matrix is then

$$H_{i+1} = H_i + \frac{P_i P_i^T}{P_i^T Y_i} - \frac{H_i Y_i Y_i^T H_i}{Y_i^T H_i Y_i} \tag{2-9}$$

Equation (2-9) is the formula as given by Fletcher and Powell in [3]. The second and third terms each provide rank one corrective terms. Fletcher in [4] discusses the possibility of other updating formulas for the H matrix. The following formula, a result also found by Broyden [5], may be used to update the H matrix:

$$(2\text{-}10)\quad H_{i+1} = H_i - \frac{P_i Y_i^T H_i}{P_i^T Y_i} - \frac{H_i Y_i P_i^T}{P_i^T Y_i} + \left(1 + \frac{Y_i H_i Y_i}{P_i^T Y_i}\right) \frac{P_i P_i^T}{P_i^T Y_i}$$

Let the matrix H_{i+1} resulting from using equation (2-9) be denoted by H^0 and let the corresponding matrix from use of (2-10) be denoted by H^1. It is known that in certain ill-conditioned problems use of (2-9) alone may lead to singularity of H^0 and it may be that use of (2-10) alone may cause H^1 to become unbounded. Fletcher in [4] suggests the use of a convex combination of H^0 and H^1, that is,

$$(2\text{-}11)\quad H = (1 - \phi)\, H^0 + \phi H^1,$$

where $\phi \in [0, 1]$, to improve the algorithm. In practice one uses H^0 or equation (2-9) if $P_i^T Y_i < Y_i^T H_i Y_i$ and H^1 otherwise. Note that if f is quadratic then replacing P_i by $F^{-1}Y_i$ the above relation becomes $Y_i^T F^{-1} Y_i < Y_i^T H_i Y_i$ which if true indicates that H_i is "larger" than F^{-1} and hence updating using the "smaller" H^0 formula is indicated. However if $Y_i^T F^{-1} Y_i \geq Y_i^T H_i Y_i$ is true, then the indication is that H_i is "smaller" than F^{-1} so that H^1 should be used.

Other problems arising out of ill conditioning of the H matrix can occur. If H_i is nearly singular then the direction $-H_i \nabla f_{K,i}$ may be nearly orthogonal to $\nabla f_{K,i}$ and the algorithm will take extremely small steps and may possibly halt. One possible remedy is to discard H_i and let H_{i+1} be the identity matrix I. Bard in [7] suggests that whenever the cosine of the angle between $H_i \nabla f_{K,i}$ and $\nabla f_{K,i}$ turns out to be less than 10^{-5} (we use double precision calculations and prefer to use 10^{-8} for this tolerance), then

reinitialize H_i to a diagonal matrix whose jj element is minus the absolute value of the jth element of P_i to the jth element of $\nabla f_{K,i}$. In a personal communication from Prof. R.P. Tewarson of the Dept. of Applied Analysis of S.U.N.Y. at Stony Brook another remedy for the ill conditioning of the H matrix was brough to our attention. The denominator of the second term in equation (2-9) is the factor $P_i^T Y_i$ which may become very small and hence cause the elements of the matrix

$$(2\text{-}12) \qquad \frac{P_i P_i^T}{P_i^T Y_i}$$

to be extremely large. Adding these elements to the elements of the other correction terms in (2-9) may cause the contribution of the first and third terms to be negligible. Tewarson suggests that the first and third terms be multiplied by a factor β where

$$(2\text{-}13) \qquad \beta = \frac{P_i^T P_i}{P_i^T Y_i}$$

This then leads to the updating formula

$$(2\text{-}14) \qquad H_{i+1} = \beta \left(H_i - \frac{H_i Y_i Y_i^T H_i}{Y_i^T H_i Y_i} \right) + \frac{P_i P_i^T}{P_i^T Y_i}$$

which is used only when the cosine of the angle between $H_i \nabla f_{K,i}$ and $\nabla f_{K,i}$ is less than 10^{-8}. Favorable results on a badly scaled objective test function

f were noted using this rescaling factor. A similar factor for updating H_i using equation (2-10) was also developed. The only change in (2-10) is to multiply H_i by the factor $1 \Big/ P_i^T Y_i$.

The selection of the scalor α_i so that $f_K(x_i + \alpha_i S_i)$ is minimized does not appear to be a formidable task at first glance. However, in practice a great number of function evaluations, which in some problems are quite costly, may be needed. The importance of finding this one dimensional minimum is necessary to prove the finite step convergence property of the algorithm when the function to be minimized is quadratic. Consider $f_K(x_i + \alpha_i S_i)$ and, using the chain rule, set the derivative with respect to α_i equal to zero to obtain

$$\frac{df_{K,\, i+1}}{d\alpha_i} = \nabla f^T_{K,\, i+1} S_i = 0$$

which also implies

$$\nabla f^T_{K,\, i+1} P_i = 0 \qquad (2\text{-}15)$$

Equation (2-15) is needed to show quadratic convergence by Fletcher and Powell in [3]. A number of methods may be used to find the precise value of α_i at each step. Interval splitting techniques and comparison of function values at interior points may be used. One may also fit a polynomial through several points and then find the minimum of the polynomial. Experience has shown however that one of the most powerful techniques is one due to Johnson

and Meyers [6] which is a combination of Golden Section search and cubic fit. Beltrami in Chapter 3 of [1] gives a discussion of Golden Section search which is a simplification of the Fibonacci procedure.

As with all calculations done on a computer the exact value of α_i may not be found by the one dimensional search. Suppose that we start at x_i and proceed to x_{i+1} according to equation (2-7). If the α_i found in the one dimensional search was not at the exact minimum of $f(x_i + \alpha_i S_i)$ then x_{i+1} will not be the exact minimum along the direction $S_i = -H_i \nabla f_{K,i}$. Denote the exact minimum by $\hat{x}_{i+1}$. Then

$$\hat{x}_{i+1} = x_{i+1} + \beta S_i \tag{2-16}$$

where $S_i = x_{i+1} - x_i$, or

$$\hat{x}_{i+1} = x_i + (1 - \beta) S_i \tag{2-17}$$

where β is generally small. Also assume that the function f_K is roughly quadratic in the region where the one dimensional minimization is performed, so that $F = \dfrac{\partial^2 f}{\partial x_i \partial x_j}$ is approximately a constant matrix.

We know that $\nabla f_K(\hat{x}_{i+1})$ is orthogonal to the direction S_i but in general $\nabla f(x_{i+1})$ is not itself orthogonal. This means that the new direction $S_{i+1} = -H_{i+1} \nabla f_K(x_{i+1})$ will not be conjugate to the previous direction and this may lessen the effectiveness of the Davidon Fletcher Powell Algorithm.

The Davidon Fletcher Powell Algorithm is such that conjugate directions are generated, that is for a quadratic

$$(2\text{-}18) \quad P_i^T F P_k = 0 \quad \text{for } 0 \leq i < k$$

as shown in [3]. We wish to estimate the true but unknown value of the gradient of f_K at $\hat{x}_{i+1}$, $\nabla f_K(\hat{x}_{i+1})$, using $\nabla f_K(x_{i+1})$. By use of Taylor's theorem one can derive the following correction formula,

$$(2\text{-}19) \quad \nabla f_K(\hat{x}_{i+1}) = \left[I - \frac{\Delta \nabla f_K(x_{i+1}) S_i^T}{\Delta \nabla f_K(x_{i+1})^T S_i} \right] \nabla f_K(x_{i+1})$$

where

$$(2\text{-}20) \quad \Delta \nabla f_K(x_{i+1}) = \nabla f_K(x_{i+1}) - \nabla f_K(x_i)$$

is the difference in gradients. This corrected gradient will then be nearly orthogonal to the direction S_i and will help preserve the convergence properties of the algorithm should the one dimensional search produce a value of α_i not exactly at the minimum. This gradient correction scheme is due to Kelley and Myers [8].

Our discussion of the Davidon Fletcher Powell Algorithm and its variations and refinements is now nearly complete. We mention in passing that Fletcher in [4] shows that the one dimensional search to determine α_i may not have to be done at each iterate. This sacrifices the quadratic convergence of the algorithm but may cause a sizable reduction in total function evaluations. He shows that the α_i chosen without one dimensional search must

satisfy simultaneously

$$(2\text{-}21) \quad 1 - \mu \geq \left(\frac{\Delta f}{\nabla f_i^T, S_i} \right) \geq \mu$$

where $0 < \mu \ll 1$ and

$$(2\text{-}22) \quad S_i^T Y_i > 0.$$

Assuming that we have performed a minimization of f_K for fixed K via the Davidon Fletcher Powell Algorithm we wish now to update the penalty constant matrix K. The penalty constants are enlarged according to the relationship

$$(2\text{-}23) \quad (k_j)_{new} = (k_j)_{old} \frac{|g_j|}{\epsilon_j}$$

if $|g_j| > \epsilon_j$, where ϵ_j is a predetermined tolerance based on how much the user will accept constraint violations. This updating relationship is given in Johnson [9] and discussed in Chapter 3 of Beltrami [1].

After increasing the penalty constants we wish to again minimize f_K using the Davidon Fletcher Powell Algorithm. To do this we will have to update the H matrix for the change in the penalty constants. The updating of H may be done by

$$(2\text{-}24) \quad H^* = H - H \nabla g_j \left(\frac{\Delta k_j}{1 + \Delta k_j \nabla g_j^T H \nabla g_j} \right) \nabla g_j^T H$$

where H^* is the new H matrix and ∇g_j is the gradient of the jth constraint. Equation (2-24) is derived in Kelley et al [10]. The matrix form of (2-24) is

$$(2\text{-}25) \quad H^* = H - HG\,[G^T H G + (\Delta K)^{-1}\,]^{-1} G^T H$$

where it is assumed that each k_j is increased so that $(\Delta K)^{-1}$ exists.

After sufficient minimizations by the DFP Algorithm and updating of the penalty constants we may now use the converged H matrix in a Newton type acceleration phase. Following Kelley et al [10] or Johnson in [9], we proceed as follows. The first order necessary conditions for the constrained minimum problem is given by the system of equations

$$(2\text{-}26) \quad \begin{aligned} \nabla f + G\lambda &= 0 \\ g &= 0 \end{aligned}$$

where λ is an m vector of multipliers and g is the m vector of constraint equations g_j. Application of Newton's method to the system of equations (2-26) gives

$$(2\text{-}27) \quad \begin{bmatrix} (f + \lambda^T g)xx & \nabla g \\ \nabla g^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla f + G\lambda \\ g \end{bmatrix}$$

where $(f + \lambda^T g)xx$ is the matrix of second partial derivatives of $f + \lambda^T g$. This matrix may be approximated by

$$(2\text{-}28) \quad H^{-1} - \nabla g K \nabla g^T$$

where we have made use of the converged H of the DFP Algorithm. In terms of known quantities, the expression for Δx obtained from (2-27) is

$$(2\text{-}29)\quad \Delta x = -[H - H\nabla g(\nabla g^T H \nabla g)^{-1}\nabla g^T H]\nabla f - H\nabla g(\nabla g^T H \nabla g)^{-1} g$$

Equation (2-29) is used in an iterative manner starting with the last solution obtained from the DFP Algorithm along with the converged H matrix from that algorithm. The convergence rate of the iteration described by (2-29) should be nearly as rapid as Newton's method providing that the quantities $k_j g_j$ furnish good approximations to the multipliers λ_j.

Should all the constraints be satisfied after use of the DFP Algorithm, then the simple Newton's method

$$(2\text{-}30)\quad x_{i+1} = x_i - H\nabla f_K(x_i)$$

is used starting with the solution obtained from the DFP minimization phase.

Iteration in each phase is stopped by the standard techniques employed in numerical analysis. Whenever two subsequent function values differ by less than some predetermined tolerance than the iterations are terminated. In addition a test to determine if the gradient of f_K is sufficiently small is also used in the DFP phase. In all cases it is felt that at least n iterates, where n is the number of independent variables in the problem, is required for each minimization phase.

- NUMERICAL RESULTS -

As an example of the numerical results obtained by using the algorithm we consider a problem due to Ivan L. Johnson of NASA, Houston, Texas. The objective function f is given as

$$f = x_1 \; x_2 \; x_3 \; x_4 \tag{3 - 1}$$

subject to the constraints

$$\begin{aligned} g_1 &= x_1^3 + x_2 - 1 = 0 \\ g_2 &= x_1^2 \, x_4 - x_3 = 0 \\ g_3 &= x_4^2 - x_2 = 0 \end{aligned} \tag{3 - 2}$$

The constrained minimum of f lies at

$$\begin{aligned} x_1 &= 0.793700 \\ x_2 &= 0.707106 \\ x_3 &= 0.592731 \\ x_4 &= 0.840896 \end{aligned} \tag{3 - 3}$$

where f has an exact minimum of - 0.25. Using the algorithm on the problem as given in (3 - 1) and (3 - 2) with starting values of

$$\begin{aligned} x_1 &= 0.8 \\ x_2 &= 0.8 \\ x_3 &= 0.8 \\ x_4 &= 0.8 \end{aligned} \tag{3 - 4}$$

and initial penalty constants of 10^3 we found that the DFP phase after 12 iterates brought the value of f to - 0.250299 with all the constraints less than 10^{-3}. After five applications of the Newton Acceleration phase the value of f agreed with -0.25 to 16 places and the constraints all zero to 16 places. At this point the value of $\|\nabla f\|$ was less than 10^{-7}.

In order to evaluate the performance of the different variations and options in the algorithm a formidable test problem was needed. It was decided to rescale the NASA test problem described above in such a way that the resulting problem was sufficiently difficult and also improperly scaled. This rescaling or distorition was accomplished by rescaling x_2 to be 10 times larger and x_4 to be 10 times smaller than in the original problem. For the tests and comparisons however, x_2 was rescaled to be 100 times larger and x_4 was rescaled to be 100 times smaller than originally. Total iterates of the DFP Algorithm and total function evaluations were noted for each variation of the algorithm as soon as the $\|\nabla f\|$ was less than 10^{-5} and the constraint equations were all less than 10^{-3}. Each test was started at the same initial point as in equation (3 - 4) and initial penalty constants were 10^3. All calculations were done in double precision on an IBM 360/67 computer. The results are given in the following table.

Option Used	Distortion	Final value of function f	Total Iterates	Total function evaluations
H^o update	None	-.250299	12	82
H^o update	$10x_2, 1/10\ x_4$	-.250299	45	441
H^o update	$100x_2, 1/100x_4$	-.250299	123	1818
Bard's H reset	"	-.250299	96	1037
Tewarson's reset	"	-.250299	86	914
H^1 update	"	-.250299	83	719
H^o and H^1 combination	"	-.250299	79	714

In addition to the NASA test problem described above, a number of other test problems were tried. In each case the results obtained compared favorably to other methods of constrained minimization.

- A Fortran Program For Nonlinear Programming -

This part of the report will deal exclusively with the use of a Fortran program to implement the algorithm described in Part I. The algorithm and hence the program were designed primarily for nonlinear problems. Use of the program for linear problems is not encouraged since there are a number of more efficient algorithms available for such problems.

The program is comprised of a main program which reads initial data from punched cards, call the subroutines, updates the penalty constants and the H matrix, and prints some of the output. The following is a list of the subroutines and their primary function.

Subroutine Name	Function
NEWT	Newton's Refinement Phase
DAVIDN	DFP Algorithm
CUBIC	Cubic polynomial fit
IMT	Computes augmented function f_K
FFXGGX	Calls FUNT, computes constraints, ∇f, and gradients of constraints
MATIN	Matrix Inversion
FUNT	Computes value of function f

The user must supply in subroutine FUNT the algebraic expression for the objective function. In FUNT the variables x_i are called ALPHA (I) and the function is called FNT.

For example if

$$f = x_1^2 + x_2^2 + 3x_3$$

then the user must insert in subroutine FUNT

```
FNT = ALPHA (1) * * 2 + ALPHA (2) * * 2
    + 3.0D + 0 * ALPHA (3)
```

In subroutine FFXGGX the user must insert the following:

i) Between comment cards INSERT CONSTRAINT EQUATIONS; the expression for each constraint equation, equality constraints first, where g_i is coded as GBAR(I) and the variables x_i as ALPHA(I). For example if

$$g_i = x_1 + 2x_2 = 0 \text{ , } g_2 = -x_3 \leq 0$$

then the user must supply

```
GBAR(1) = -ALPHA(1) + 2.0D + 0 * ALPHA(2)
GBAR(2) = -ALPHA(3)
```

ii) Between comment cards INSERT GRADIENT OF F; the expression for each component of the gradient of f where $\frac{\partial f}{\partial x_i}$ is coded as FNTX(I). Following the previous example for

$$\frac{\partial f}{\partial x_1} = 2x_1, \quad \frac{\partial f}{\partial x_2} = 2x_2, \quad \frac{\partial f}{\partial x_3} = 3$$

the user must supply

```
FNTX(1) = 2.0D + 0* ALPHA(1)
FNTX(2) = 2.0D + 0* ALPHA(2)
FNTX(3) = 3.0D + 0
```

iii) Between comment cards INSERT GRADIENTS OF CONSTRAINTS; the expression for the gradient of each constraint as a row of the m by n matrix G. Row i of G is the gradient of constraint i and following the previous example we have

$$G = \begin{bmatrix} -1 & 2 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

so that one must insert

G(1,1) = 1.0D + 0

G(1,2) = 2.0D + 0

etc.

In addition the user must supply the following as SYSIN input data on punched cards with the formats as specified in parenthesis..

N number of variables, (I5)

MEQ number of equality constraints, (I5)

MINEQ number of inequality constraints, (I5)

XBARS(I) the initial guess or choice of the n independent variables x_i, (D23.16)

DELF a tolerance on the difference between two successful values of f_K, generally equal to $.1x10^{-8}$, (D23.16)

CAYY(I) the starting values for the penalty constants, there are MC = MEQ+MINEQ of these values to be supplied,(D23.16) See note 1.

GEPS(I) Tolerance on the value of the constraints generally all are taken to be 10^{-3}, (D23.16) See note 2.

DELFBR a tolerance similar to DELF except that DELFBR is used in the Newton acceleration phase to stop iteration, generally equal to $.1x10^{-12}$,(D23.16)

KUBIC — parameter which signifies which search method is to be used for the one-dimensional minimization,(I5) See note 3.

KEYH — parameter which signifies which updating formula for the H matrix is to be used in DAVIDN subroutine. KEYH = 0 signifies use of H^0 (Davidon-Fletcher-Powell), KEYH = 1 signifies use of H^1 (Broyden), and KEYH = 2 signifies use of H^0 or H^1 depending on the criteria given in Part 1 of this report, (I5).

In addition to the value of the objective function and the values of the independent (solution) variables x^i other useful information is clearly identified and printed out. Of particular interest are the values of the constraint equations and the value of the gradient of f at each iterate. For the equality constraints and the active inequality constraints the Lagrange and Kuhn-Tucker multipliers are printed out. At the end, the values of the active and inactive constraints in addition to the multipliers (which are associated with the equality and active inequality constraints) are printed out.

To illustrate the use of this deck on a sample problem consider the function

$$f(x_1,x_2) = (x_1-2)^2 + (x_2-1)^2$$

which is to be minimized subject to the constraints

$$g_1(x_1,x_2) = x_2 - x_1^2 \leq 0$$

$$g_2(x_1,x_2) = x_1 + x_2^2 \leq 0$$

The constrained minimum of f occurs at $x_1 = 1$, $x_2 = 1$ where the value of f is equal to 1. Both constraints are active at the minimum and the multipliers are each 2/3. Input information for this problem is N = 2, MEQ = 0, MINEQ = 2, XBAR(1) = 0.0D+0, XBAR(2) = 0.0D+0, DELF = 0.1D - 08, CAYY(1) = CAYY(2) = 16.0D+0, GEPS(1) = GEPS(2) = .001D+00, DELFBR = 0.1D - 12, KUBIC = 2, KEYH = 0. Partial output for this problem at the last iterate is

SOLUTION VECTOR IS

0.1D+01 0.1D+01

FUNCTION VALUE F = .999→990+00

GBAR(1) = GBAR(2) = 0.0D+00

The deck is supplied with this sample problem included. The user should run this problem to familiarize himself with the use of the deck. For further information concerning this program contact Mr. J.P. Indusi, Department of Applied Analysis, S.U.N.Y., Stony Brook, New York.

- Notes -

1. At present no precise manner of choosing the initial values of the penalty constants is known. Typical values range between 15 and 100. A reasonably safe try is 15 and if no convergence is observed then one should try larger values in subsequent trials.

2. The tolerances GEPS(I) are reduced or tightened automatically in the program.

3. There are 4 search options available. They are discussed briefly below and in the references given in Part 1.

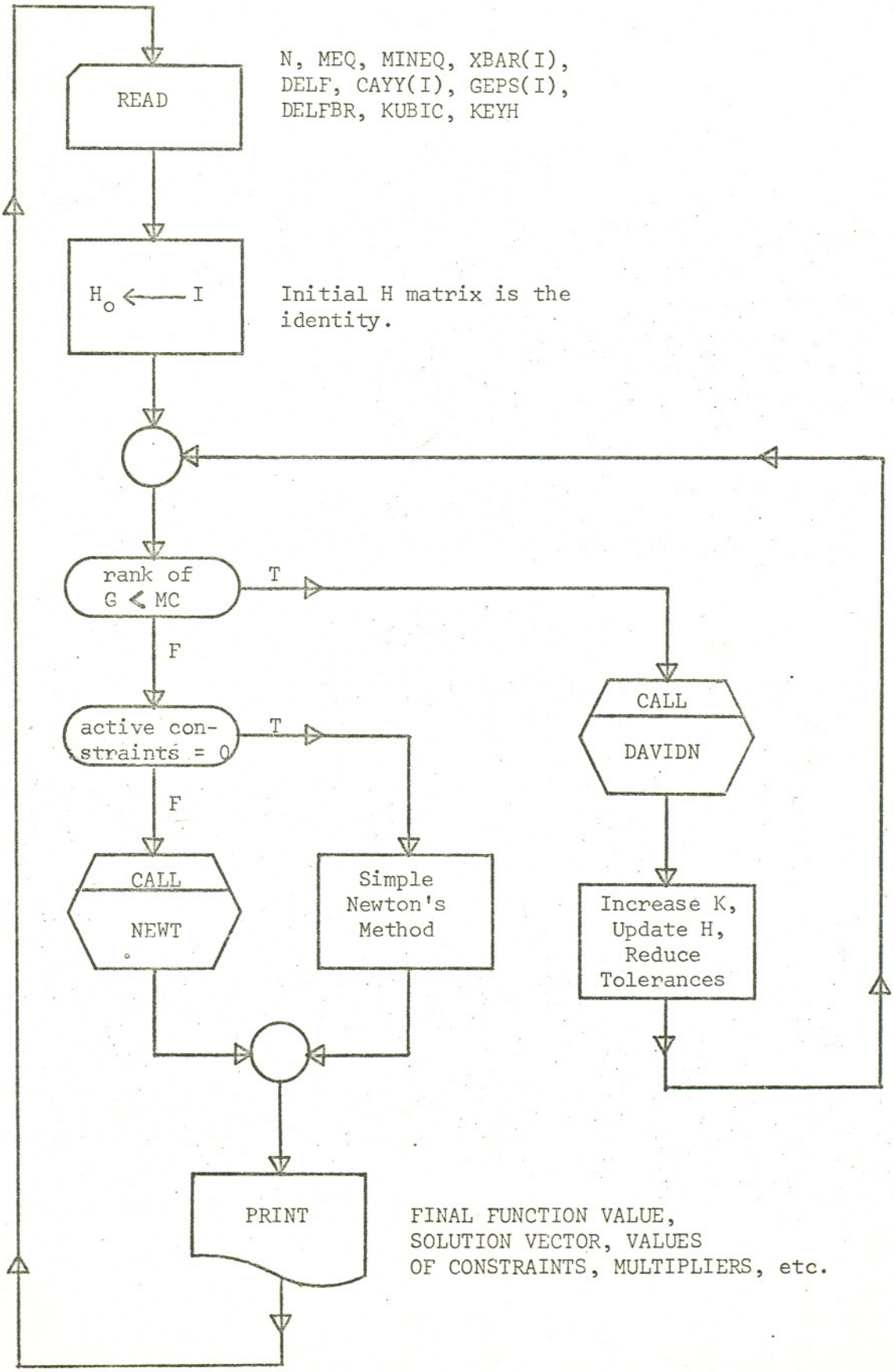
   i) KUBIC = 0 is search by Golden Section. A fairly sophisticated method of finding one dimensional minimum.

   ii) KUBIC = 1 is search by cubic interpolation method.

   iii) KUBIC = 2 is a combination of i) and ii) above. It is the most sophisticated of all the search options and requires the most number of function evaluations. For certain problems the program will switch from KUBIC = 1 to KUBIC = 2 automatically.

   iv) KUBIC = 3 is search by simple quadratic interpolation. Uses fewest function evaluations but should not be used when user suspects that the objective function is very non-linear.

MAIN PROGRAM FLOW CHART



- REFERENCES -

[1] Beltrami, E.J., "An Algorithmic Approach to Nonlinear Analysis and Optimization". Academic Press, New York, 1970.

[2] Beltrami, E.J., "A Comparison of Some Recent Iterative Methods for the Numerical Solution of Nonlinear Programs". Lecture Notes in Operations Research and Mathematical Economics, Springer-Verlag, New York, 1969. Originally given at Second International Conference on Computing Methods in Optimization Problems, San Remo, Italy, September 9-13, 1968.

[3] Fletcher, R. and Powell, M.J., "A Rapidly Convergent Descent Method for Minimization", Compt. J. 6,1963. pp. 163-168.

[4] Fletcher, R., "A New Approach to Variable Metric Algorithms", U.K.A.E.A. Research Group, Atomic Energy Research Establishment, Harwell, Great Britian, October, 1969.

[5] Broyden, C.G., "The Convergence of a Class of Double-rank Minimization Algorithms Parts I and II", Journal Institute of Mathematics and its Applications (to be published).

[6] Johnson, I.L. and Myers, G.E., "One-Dimensional Minimization Using Search by Crolden Section and Cubic Fit Methods", NASA MSC Internal Note No. 67-FM-172. November 13, 1967.

[7] Bard, Y., "Comparison of Gradient Methods for the Solution of Nonlinear Parameter Estimation Problems". Tech. Report No. 320-2955, IBM, New York Scientific Center, September, 1968, p. 14.

[8] Kelley, H.J. and Myers, G.E., "Conjugate Direction Methods for Parameter Optimization". Presented at 18th Congress of International Astronautical Federation, Belgrade, Yugoslavia, September 24-30, 1967.

[9] Johnson, Jr., I.L., "Impulsive Orbit Transfer Optimization by an Accelerated Gradient Method". NASA MSC Internal Note No. 68-FM-88, April 12, 1968.

[10] Kelley, H.J., Denham, W.F., Johnson,Jr.,I.L. and Wheatley, P.O., "An Accelerated Gradient Method for Parameter Optimization with Nonlinear Constraints". Journal of Astron. Sci., Vol. 13 (1966) pp. 166-169.