AERE - R 7125

United Kingdom Atomic Energy Authority

RESEARCH GROUP

Report

FORTRAN SUBROUTINES FOR MINIMIZATION BY QUASI-NEWTON METHODS

R. FLETCHER

Theoretical Physics Division,

Atomic Energy Research Establishment,

Harwell, Berkshire.

1972

Price £1.00 net from H. M. Stationery Office.

C. 13

AERE - R 7125

FORTRAN SUBROUTINES FOR MINIMIZATION BY QUASI-NEWTON METHODS

by

R. Fletcher

Theoretical Physics Division,
U.K.A.E.A. Research Group,
Atomic Energy Research Establishment,
HARWELL

June, 1972

HL72/2747(C13)
/KE

CONTENTS

APPENDICES

TABLES

ILLUSTRATION

SBN 7058 0112 8

1. Introduction

This report presents two quasi-Newton subroutines for solving the problem of minimizing a function $F(\underset{\sim}{x})$ of n variables $\underset{\sim}{x}$. It is assumed that the gradient vector $\underset{\sim}{\nabla}F(\underset{\sim}{x}) = \underset{\sim}{g}(\underset{\sim}{x})$ and the hessian matrix $G = [\partial^2 F/(\partial x_i \partial x_j)]$ exist. Quasi-Newton methods have achieved a high degree of popularity since they were introduced some 14 years ago. However many such methods have been published since that time and in this paper the recent ideas of Gill, P.E. and Murray, W. ('Quasi-Newton Methods for Unconstrained Optimization', Journal of the Institute of Mathematics and its Applications, Vol. 9, p. 91-108, 1972) will be followed. It is recommended that this report be read in conjunction with Gill & Murray's paper.

The main feature of classical quasi-Newton methods is that an approximation H_k to G^{-1} is stored and used by analogy with Newton's method. That is to say, given an approximation to the solution $\underset{\sim}{x}_k$ on the k^{th} iteration, then $\underset{\sim}{x}_{k+1}$ is chosen as a point

$$\underset{\sim}{x}_{k+1} = \underset{\sim}{x}_k + \alpha \underset{\sim}{p}_k \qquad (1)$$

where

$$\underset{\sim}{p}_k = - H_k \underset{\sim}{g}_k \qquad (2)$$

H_k is updated after each iteration by making a change of low rank. The scalar α in (1) is chosen ideally to optimize $F(\underset{\sim}{x}_k + \alpha \underset{\sim}{p}_k)$ with respect to α, or in practice so that $\underset{\sim}{x}_{k+1}$ satisfies some criterion of approximate optimality. The total amount of work per iteration is $O(n^2)$ computer operations.

In this paper two problems are referred to, namely that in which both $F(\underset{\sim}{x})$ and $\underset{\sim}{g}(\underset{\sim}{x})$ can be calculated explicitly, and that in which only $F(\underset{\sim}{x})$ can be calculated. In the latter case a quasi-Newton method can only be implemented by approximating the derivatives $\underset{\sim}{g}(\underset{\sim}{x})$ by differences. These two aspects of the problem are considered in sections 2 and 3 of this paper respectively. The main aim of the work has been to develop FORTRAN subroutines for each of the problems, and these are given in Appendices 1 and 2 respectively together with the specification sheets for the subroutines.

2. The Method with Derivatives

There are three main problems in implementing a quasi-Newton method when both $F(\underset{\sim}{x})$ and $\underset{\sim}{g}(\underset{\sim}{x})$ can be calculated explicitly. The first of these concerns the representation of the approximation which is maintained to the hessian matrix. In early implementations it was convenient to keep an approximation H_k to G^{-1}. Gill and Murray's contribution (loc.cit.)

was to represent G by the product

$$G \approx L\,D\,L^T = B \tag{3}$$

where L is a lower unit triangular matrix and D a diagonal matrix. An important feature of this method is that the approximation to G is always kept positive definite and it is easy to guarantee this by ensuring $d_{ii} > 0$ for all i. The product $-B^{-1}g_k$ is achieved by making two back substitutions. When a rank one correction is made to B,

$$B^* = B + \underset{\sim}{u}\,\underset{\sim}{u}^T \tag{4}$$

say, then it is possible to update L and D to L*,D* in $3n^2/2 + O(n)$ multiplications. Gill and Murray give two methods for updating D; the first requires more work (including n square roots) and n extra storage locations, but Gill and Murray prefer it because they believe it is easier to ensure positive definiteness in the formulae for obtaining d^*_{ii}. However the second method which they give has been used here because of its efficiency. The method uses the recurrence

$$d^*_{ii} = d_{ii} + v_i^2\left(\sigma - \sum_{j=1}^{i-1} \beta_j^2\, d^*_{jj}\right) \tag{5}$$

where $\underset{\sim}{v}$ and $\underset{\sim}{\beta}$ are certain vectors, and it is possible that d^*_{ii} could become negative due to accumulation of round off errors, even when it is theoretically predicted to be positive. It is easy to correct for this however, and if a negative d^*_{ii} is found then it is replaced by the smallest positive d_{ii} in any previous matrix D. This is equivalent to making an increase to d_{ii} before applying the recurrence. No problem with round off errors has occurred when using this device. It is worth pointing out that the use of LDL^T representations with rank 2 correction formulae requires $4n^2 + O(n)$ multiplications per iteration whereas the classical method of recurring an approximation to G^{-1} requires only $3n^2 + O(n)$. However this is a minor disadvantage when compared against the ease with which positive definiteness of the representation of G can be assured.

The second problem which must be solved when implementing a Quasi-Newton method is in the choice of the correction formula to be used. There is a clear advantage to be gained by maintaining positive definiteness, and this leaves three competing formulae, the DFP formula, the complementary DFP formula, and the switching strategy of Fletcher ('A new approach to variable metric algorithms', Computer Journal, Vol. 13, pp. 317-322, 1970) which chooses either the DFP formula or its complement depending upon the test

$$\underset{\sim}{\delta}^T \underset{\sim}{\gamma} > \underset{\sim}{\gamma}^T H \underset{\sim}{\gamma} \qquad (6)$$

where $\underset{\sim}{\delta} = \underset{\sim}{x}_{k+1} - \underset{\sim}{x}_k$, $\underset{\sim}{\gamma} = \underset{\sim}{g}_{k+1} - \underset{\sim}{g}_k$ and $H \approx G^{-1}$. Unfortunately it is not convenient to use this test when (3) is used to represent G because it involves calculating $H\underset{\sim}{\gamma}$, a quantity not necessary in the Gill and Murray formulation. However one way of looking at (6) is to consider it to be comparing H and the true G^{-1} (because $\underset{\sim}{\delta} = G^{-1}\underset{\sim}{\gamma}$ for a quadratic function) in the direction $\underset{\sim}{\gamma}$. A similar test therefore which it is convenient to compute with the Gill and Murray formulation, is to use the DFP formula if

$$\underset{\sim}{\delta}^T LDL^T \underset{\sim}{\delta} < \underset{\sim}{\delta}^T \underset{\sim}{\gamma} \qquad (7)$$

which can be thought of as comparing G and LDL^T in the direction $\underset{\sim}{\delta}$. Tests have been carried out using all three formulae on a variety of test problems (Table 1), and they show that use of the DFP formula alone is much inferior. The performance of the complementary DFP formula alone and the modified switching strategy is virtually identical, but the latter has been preferred in the FORTRAN subroutine because it is felt to be more flexible.

TABLE 1

Comparison of various strategies for updating the approximation to the hessian matrix

Problem n	DFP formula		Complementary DFP formula		Switching formula	
Rosenbrock's function						
2	45	48[a]	33	42	34	44
Chebyquad						
2	3	6	3	6	3	6
4	9	13	9	13	9	13
6	22	26	15	18	16	20
8	31	38	21	26	20	25
Trigonometric functions						
2	9	12	9	12	9	12
4	30	36	16	22	16	22
6	14	19	13	18	13	18
8	15	22	16	23	16	23
10	18	24	18	24	18	24
20	71	86	41	56	41	56
30	197	215[b]	67	85	70	88
40	161	176[b]	81	98	81	97

(a) Entries are No. of iterations, No. of function & gradient evaluations.
(b) Low accuracy.

A third problem is that of choosing the parameter α in (1), the so-called linear search subproblem. For rapid ultimate convergence it is expected that a value $\alpha = 1$ will be used, but in the early stages when LDL^T is a poor approximation to G, $\alpha = 1$ may be too large. In these circumstances a good approximation to make is that the reduction in F is likely to be similar to that obtained on the previous iteration, in which case a quadratic interpolation would give $\alpha = 2(F_k - F_{k-1})/\underset{\sim}{g}_k^T \underset{\sim}{p}_k$. Thus a likely choice for α is given by

$$\alpha = \min \left\{ 1,\ 2(F_k - F_{k-1})/\underset{\sim}{g}_k^T \underset{\sim}{p}_k \right\} \qquad (8)$$

This strategy requires the user to estimate the likely reduction in F on the first iteration. The linear search can be terminated when a value of α is obtained for which $F_{k+1} < F_k$ and

$$\left| g(\underset{\sim}{x}_k + \alpha \underset{\sim}{p}_k)^T \underset{\sim}{p}_k \right| < \rho(-\underset{\sim}{g}_k^T \underset{\sim}{p}_k) \qquad (9)$$

where ρ is a fixed parameter in the range $0 < \rho < 1$. Because it has been shown (Fletcher, loc.cit.) that it is most efficient overall when using Quasi-Newton methods to look for low accuracy in the linear search, a value of $\rho = .9$ has been used. If $F(\underset{\sim}{x}_k + \alpha \underset{\sim}{p}_k) \geqslant F_k$ or if (9) is not satisfied and $\underset{\sim}{g}(\underset{\sim}{x}_k + \alpha \underset{\sim}{p}_k)^T \underset{\sim}{p}_k > 0$ then a new value of α is determined using the well known cubic interpolation formula. If (9) is not satisfied but $\underset{\sim}{g}(\underset{\sim}{x}_k + \alpha \underset{\sim}{p}_k)^T \underset{\sim}{p}_k < 0$ then a new value of α is determined by extrapolation assuming a linear behaviour of $\underset{\sim}{g}(\underset{\sim}{x}_k + \alpha \underset{\sim}{p}_k)$ with respect to α. To ensure stability α is not allowed to increase by a factor of greater than 10 on extrapolation. Two points will be noticed about this search method. One is that it guarantees $\underset{\sim}{\delta}^T \underset{\sim}{\gamma} > 0$ which is a necessary and sufficient condition for the approximation to G to remain positive definite, and another is that it requires on average about 1.25 function evaluations per iteration as will be seen from Table 1.

Finally the problem of how to terminate the algorithm will be considered. Because the algorithm generally converges rapidly near the minimum, it has been decided to terminate when $|(\underset{\sim}{x}_k - \underset{\sim}{x}_{k+1})_{(i)}| < \varepsilon_i$ for all i, where $\underset{\sim}{\varepsilon}$ is a tolerance vector supplied by the user. It is however important not to set $\underset{\sim}{\varepsilon}$ too large when using this criterion.

The new subroutine (identifier VA09A) has been tested against two other subroutines in the Harwell Subroutine Library, namely VA01A, a convention Quasi-Newton algorithm using

the DFP formula and an accurate linear search, and VA06A, a Quasi-Newton algorithm of an unconventional type due to Powell ('A FORTRAN subroutine for unconstrained minimization requiring first derivatives of the objective function' report AERE - R 6469, 1970) which can be guaranteed to converge in exact arithmetic. The results in Table 2 show that VA09A is superior to both VA01A and VA06A in both efficiency and reliability. In particular VA06A is more affected by the presence of round-off errors, and it was also found difficult to use because of the need to supply a convergence tolerance on the gradient vector. It will be noted that the results for VA09A do not quite correspond to those for the switching formula in Table 1. This is because the results of Table 1 were obtained with a maximum extrapolation factor of 4 rather than 10.

TABLE 2

Comparison of VA09A against other existing library subroutines

Problem n	VA01A		VA06A		VA09A	
Rosenbrock's function						
2	30	62[a]	42	43	34	44
Chebyquad						
2	2	7	13	14	3	6
4	8	22	18	19	9	13
6	11	26	27	28	16	20
8	failed		66	67	20	25
Trigonometric functions						
2	7	18[b]	13	14	8	11
4	14	31	19	20	16	22
6	11	26	24	25	12	17
8	12	29	22	23	15	22
10	17	38	37	38	18	25[b]
20	-	98	84	85	42	55
30	-	165	failed		74	92
40	failed		failed		82	96

(a) Entries as for Table 1

(b) Finds a different minimum.

3. The Method without Derivatives

When Quasi-Newton methods are modified to use difference approximations to derivatives then a number of additional problems raise their heads. However the means of representing

and updating the approximation to G and of terminating the iteration are no different to the case when derivatives are available, and the recommendations of section 2 have been followed. The new problems lie in the choice of the differencing interval, the choice of finite difference formula, and in the way in which the linear search subproblem is solved.

In the first paper on this subject, Stewart ('A Modification of Davidon's Minimization Method to Accept Difference Approximations of Derivatives', Journal of the ACM, Vol. 14, pp. 72-83, 1967) recommended that the diagonal elements of H^{-1} be recurred as an approximation to the diagonal elements of G. This could then be used to estimate the truncation error in using the forward difference formula

$$g_i(\underset{\sim}{x}) = (F(\underset{\sim}{x} + h\underset{\sim}{e_i}) - F(\underset{\sim}{x}))/h + O(h) \qquad (10)$$

where $\underset{\sim}{e_i}$ is the i-th coordinate vector. The differencing interval h could then be chosen in such a way as to balance the effects of trunation and round off error. However Gill and Murray give good reasons why this strategy should not be used and prefer to set $h = 2^{-t/2}$, assuming the variables have been scaled suitably, where t is the number of significant binary digits in F. This choice of differencing interval has been used in the implementation given here.

Another problem is the choice of whether to use the forward difference formula (10) or the central difference formula

$$g_i(\underset{\sim}{x}) = \frac{F(\underset{\sim}{x} + h\underset{\sim}{e_i}) - F(\underset{\sim}{x} - h\underset{\sim}{e_i})}{2h} + O(h^2) . \qquad (11)$$

Various strategies were considered for this decision, the most simple being to switch from forward differences to central differences if the step $\alpha\underset{\sim}{p}$ is less than the tolerance $\underset{\sim}{\varepsilon}$ required on the solution by the user. This is very similar to the method which Gill and Murray suggest. Gill and Murray also suggest that a return be made to using forward differences in certain circumstances. Typically one might do this if $||\alpha\underset{\sim}{p}|| > 100\ ||\underset{\sim}{\varepsilon}||$ say. Unfortunately it was found that on switching to central differences near the minimum the value of $||\alpha\underset{\sim}{p}||$ could increase greatly, and it was not easy to pick out a suitable tolerance on a test for returning to forward differences. Thus in the implementation of the strategy given here, the algorithm stays with central differences until convergence, once the switch has been made. An alternative strategy is to estimate $\underset{\sim}{g_k}^T\underset{\sim}{p_k}$ to second order accuracy during the linear search process and to switch to central

differences when this estimate of $g_k^T p_k$ and that obtained directly from the gradient vector fail to agree to a certain accuracy. The switch back to forward differences could be made if $g_k^T p_k$ evaluated using the g from both (10) and (11) agreed to a certain accuracy. Some of these quantities were monitored on some test runs on various problems and it was found that the agreement between the two estimates of $g_k^T p_k$ was not always a good indicator of when the switch to central differences ought to be made. It was therefore not decided to implement this more sophisticated strategy, but to use the simple one.

Finally the problem of how to implement the linear search subproblem must be solved. The method used is described by the flow diagram of figure 1, and some features of this will be discussed in more detail. Firstly note that as in section 2, equation (8) is used to determine α initially. It is important however to ensure that a small $F_k - F_{k-1}$ does not cause a small α_k and hence a small $F_{k+1} - F_k$ and so on. To prevent this, provision is made for an extrapolation phase with at least two function evaluations per iteration. A second point is that for efficiency the extrapolation loop is only continued while it is predicted that good progress will be made. This is quantified as follows: if F^-, F and F^+ are three values of F(x) at equally spaced points x^-, x, x^+ on the extrapolation phase, where $x^+ = x + \alpha p = x^- + 2\alpha p$, then it may be possible to predict a point, x_{min} say, which is the minimum of a quadratic function which interpolates the F's. If the point $x^+ + 2\alpha p$ lies on the opposite side of x_{min} to x^+ then it is decided not to examine $F(x^+ + 2\alpha p)$ and the extrapolation phase is stopped; this is done by setting INT=2 as indicated in figure 1. To prevent this decision causing premature termination, the termination test is not carried out if an exit is made from the linear search with INT=2.

In fact alternatives to this linear search have been considered and a version was programmed in which an extra interpolation was carried out in the lower FINISH block. However numerical tests suggested that this change made the method less efficient. Hence the version described in figure 1 was finally implemented.

The method was implemented in a FORTRAN subroutine VA10A and was compared against the Harwell subroutine Library routine VA04A which is based on Powell's conjugate direction technique. Both methods were applied to the solution of various problems with a tolerance of .00005, and the results are shown in Table 3. In fact this tolerance was only achieved on the smaller problems due to the effects of round off error. However the smallest function value obtained by each method is shown in the table.

TABLE 3

Comparison of routines VA04A and VA10A

Problem n	VA04A			VA10A		
Rosenbrock's function						
2	32	178	$.9_{10}-10$ [a]	35	172	$.7_{10}-10$
Chebyquad						
2	8	35	$.7_{10}-11$	12	55	$.1_{10}-10$
4	32	104	$.6_{10}-12$	12	84	$.5_{10}-9$
6	114	369	$.4_{10}-10$	20	191	$.2_{10}-8$
8	400	868	$.2_{10}-6$	27	402	$.1_{10}-8$
Trigonometric functions						
2	10	55	$.3_{10}-7$	12	91	$.4_{10}-6$
4	40	137	$.5_{10}-6$	14	121	$.1_{10}-4$
6	102	349	$.9_{10}-6$	12	145	$.3_{10}-5$
8	152	500	$.3_{10}-5$	13	238	$.1_{10}-5$
10	200	701	$.8_{10}-4$	21	365	$.3_{10}-5$
20	500	1872	$.2_{10}1$	38	1235	$.1_{10}-4$
30	1360	4038	$.5_{10}0$	59	2809	$.7_{10}-4$
40	920	3340	$.2_{10}2$	86	4949	$.9_{10}-3$

(a) Entries are 'No. of linear searches', 'No. of function evaluations', and 'difference between the final $F(\underset{\sim}{x})$ and its minimum value'.

Although VA04A performs better on some of the two variable problems, VA10A becomes progressively better as n increases. Only VA10A manages to get reasonable accuracy on the large problems of 20 variables and more, and VA04A essentially fails on these problems. The results of the comparison are such that it has been decided to recommend the use of VA10A in the first instance when solving general minimization problems without derivatives.

APPENDIX 1

A FORTRAN subroutine VA09A for minimization when the function and first derivative can be calculated explicitly

1. Purpose

To find the minimum of a function $F(\underline{x})$ of several variables, given that the gradient vector $(\partial F/\partial x_1, \partial F/\partial x_2, \ldots, \partial F/\partial x_n)$ can be calculated. The subroutine replaces VA01A to which it is superior in various ways (see section 5), and should be used whenever derivatives can be evaluated readily. It should however not be used either if storage space is at a premium (use VA08A) or if the function is a sum of squares (use VA07A). The subroutine complements VA06A, the latter requires four times the storage, and some comparisons (R. Fletcher, A.E.R.E. Report, in preparation) indicate that VA06A is marginally slower and more affected by round off error. As VA06A is more difficult to use, it is suggested that VA09A should be used in the first instance on any problem. If VA09A fails then VA06A should be tried as it is guaranteed to converge if the effect of rounding errors can be neglected.

2. Argument List

CALL VA09A(FUNCT,N,X,F,G,H,W,DFN,EPS,MODE,MAXFN,IPRINT,IEXIT)

FUNCT — An IDENTIFIER of the users subroutine - see section 3.

N — An INTEGER to be set to the number of variables ($N \geqslant 2$).

X — A REAL ARRAY of N elements in which the current estimate of the solution is stored. An initial approximation must be set in X on entry to VA09A and the best estimate obtained will be returned on exit.

F — A REAL number in which the best value of $F(\underline{x})$ corresponding to X above will be returned.

G — A REAL ARRAY of N elements in which the gradient vector corresponding to X above will be returned. Not to be set on entry.

H — A REAL ARRAY of N*(N+1)/2 elements in which an estimate of the hessian matrix $\partial^2 F/(\partial x_i \partial x_j)$ is stored. The matrix is represented in the product form LDL^T where L is a lower triangular matrix with unit diagonals and D is a diagonal matrix. The lower triangle of L is stored by columns in H excepting that the unit diagonal elements are replaced by the corresponding elements of D. The setting of H on entry is controlled by the parameter MODE (q.v.).

W — A REAL ARRAY of 4*N elements used as working space.

DFN — A REAL number which must be set so as to give VA09A an estimate of the likely reduction to be obtained in $F(\underline{x})$. DFN is used only on the first iteration so an order of magnitude estimate will suffice. The information can be provided in different ways depending upon the sign of DFN which should be set in one of the following ways:

DFN>0 the setting of DFN itself will be taken as the likely reduction to be obtained in $F(\underline{x})$.

DFN=0 it will be assumed that an estimate of the minimum value of $F(\underline{x})$ has been set in argument F, and the likely reduction in $F(\underline{x})$ will be computed according to the initial function value.

DFN<0 a multiple |DFN| of the modulus of the initial function value will be taken as an estimate of the likely reduction.

EPS — A REAL ARRAY of N elements to be set on entry to the accuracy required in each element of X.

MODE — An INTEGER which controls the setting of the initial estimate of the hessian matrix in the parameter H. The following settings of MODE are permitted.

MODE=1 An estimate corresponding to a unit matrix is set in H by VA09A.

MODE=2 VA09A assumes that the hessian matrix itself has been set in H by columns of its lower triangle, and the conversion to LDL^T form is carried out by VA09A. The hessian matrix must be positive definite.

MODE=3 VA09A assumes that the hessian matrix has been set in H in product form. This is convenient when using the H matrix from one problem as an initial estimate for another, in which case the contents of H are passed on unchanged.

MAXFN — An INTEGER set to the maximum number of calls of FUNCT permitted.

IPRINT — An INTEGER controlling printing. Printing occurs every |IPRINT| iterations and also on exit, in the form

```
Iteration No, No of calls of FUNCT,IEXIT (on exit only)
Function value
X(1),X(2),...,X(N)        8 to a line (5 in VA09AD)
G(1),G(2),...,G(N)        8 to a line (5 in VA09AD)
```

The values of X and G can be suppressed on intermediate iterations by setting IPRINT<0. All intermediate printing can be suppressed by setting IPRINT=MAXFN+1. All printing can be suppressed by setting IPRINT=0.

IEXIT An INTEGER giving the reason for exit from VA09A. This will be set by VA09A as follows

IEXIT=0 (MODE=2 only). The estimate of the hessian matrix is not positive definite.

IEXIT=1 The normal exit in which $|DX(1)|<EPS(I)$ for all I=1,2,...N, where DX(I) is the change in X on an iteration.

IEXIT=2 $G^{T}DX \geq 0$. Not possible without rounding error. Probable cause is that EPS is set too small for computer word length.

IEXIT=3 FUNCT called MAXFN times.

3. User Subroutine

The user must provide a subroutine headed

```
SUBROUTINE XXX(N,X,F,G)
REAL X(1),G(1)                    (REAL*8 in VA09AD)
```

where XXX is an identifier chosen by the user.

This subroutine should use the variables $\underline{x}$ supplied in X(1), X(2),...,X(N) to evaluate the function and gradient vector and place them in F and G(1),G(2),...,G(N) respectively. XXX must be passed to VA09A as VA09A's first argument, see section 2, and appear in an EXTERNAL statement in the program that calls VA09A.

4. General

Use of COMMON : none

Workspace: N*(N+1)/2 words + 4N words provided by the user in H and W.

Other routines: none

Input/Output: controlled by the user through IPRINT. All output is on stream 6 (line printer).

Restrictions: none

System dependence: none

Date of routine: April, 1972.

5. Method

The method used is a quasi-Newton method described by Fletcher (Computer Journal, Vol. 13, p.317, 1970), and is a modification of earlier methods of this type, such as that implemented by VA01A. The method is superior to that of VA01A on three counts.

(1) It uses a formula to update the hessian approximation H which has proved to be more efficient and reliable.

(2) It uses a 'crude' line search which has been shown to be more efficient than an 'accurate' line search.

(3) It represents H by the product LDL^T, which enables the positive definiteness of H to be guaranteed, even in the presence of round-off error.

```
0001                  SUBROUTINE VA09A(FUNCT,N,X,F,G,H,W,DFN,EPS,MODE,MAXFN,IPRINT,
                     1      IEXIT)
0002                  REAL X(1),G(1),H(1),W(1),EPS(1)
0003                  IF(IPRINT.NE.0)PRINT 1000
0004                1000 FORMAT('1ENTRY TO VA09A'/)
0005                  NP=N+1
0006                  NI=N-1
0007                  NN=N*NP/2
0008                  IS=N
0009                  IU=N
0010                  IV=N+N
0011                  IB=IV+N
0012                  IEXIT=0
0013                  IF(MODE.EQ.3)GOTO15
0014                  IF(MODE.EQ.2)GOTO10
0015                  IJ=NN+1
0016                  DO 5 I=1,N
0017                  DO 6 J=1,I
0018                  IJ=IJ-1
0019                6 H(IJ)=0.
0020                5 H(IJ)=1.
0021                  GOTO15
0022               10 CONTINUE
0023                  IJ=1
0024                  DO 11 I=2,N
0025                  Z=H(IJ)
0026                  IF(Z.LE.0.)RETURN
0027                  IJ=IJ+1
0028                  II=IJ
0029                  DO 11 J=I,N
0030                  ZZ=H(IJ)
0031                  H(IJ)=H(IJ)/Z
0032                  JK=IJ
0033                  IK=II
0034                  DO 12 K=I,J
0035                  JK=JK+NP-K
0036                  H(JK)=H(JK)-H(IK)*ZZ
0037               12 IK=IK+1
0038               11 IJ=IJ+1
0039                  IF(H(IJ).LE.0.)RETURN
0040               15 CONTINUE
```

```
0041                IJ=NP
0042                DMIN=H(1)
0043                DO 16 I=2,N
0044                IF(H(IJ).GE.DMIN)GOTO16
0045                DMIN=H(IJ)
0046             16 IJ=IJ+NP-I
0047                IF(DMIN.LE.0.)RETURN
0048                Z=F
0049                ITN=0
0050                CALL FUNCT(N,X,F,G)
0051                IFN=1
0052                DF=DFN
0053                IF(DFN.EQ.0.)DF=F-Z
0054                IF(DFN.LT.0.)DF=ABS(DF*F)
0055                IF(DF.LE.0.)DF=1.
0056             20 CONTINUE
0057                IF(IPRINT.EQ.0)GOTO21
0058                IF(MOD(ITN,IPRINT).NE.0)GOTO21
0059                PRINT 1001,ITN,IFN
0060          1001 FORMAT(24I5)
0061                PRINT 1002,F
0062          1002 FORMAT((8E15.7))
0063                IF(IPRINT.LT.0)GOTO21
0064                PRINT 1002,(X(I),I=1,N)
0065                PRINT 1002,(G(I),I=1,N)
0066             21 CONTINUE
0067                ITN=ITN+1
0068                W(1)=-G(1)
0069                DO 22 I=2,N
0070                IJ=I
0071                I1=I-1
0072                Z=-G(I)
0073                DO 23 J=1,I1
0074                Z=Z-H(IJ)*W(J)
0075             23 IJ=IJ+N-J
0076             22 W(I)=Z
0077                W(IS+N)=W(N)/H(NN)
0078                IJ=NN
0079                DO 25 I=1,N1
0080                IJ=IJ-1
0081                Z=0.
```

```
0082                DO 26 J=1,I
0083                Z=Z+H(IJ)*W(IS+NP-J)
0084             26 IJ=IJ-1
0085             25 W(IS+N-I)=W(N-I)/H(IJ)-Z
0086                GS=0.
0087                DO 29 I=1,N
0088             29 GS=GS+W(IS+I)*G(I)
0089                IEXIT=2
0090                IF(GS.GE.0.)GOTO92
0091                GSO=GS
0092                ALPHA=-2.*DF/GS
0093                IF(ALPHA.GT.1.)ALPHA=1.
0094                DF=F
0095                TOT=0.
0096             30 CONTINUE
0097                IEXIT=3
0098                IF(IFN.EQ.MAXFN)GOTO92
0099                ICON=0
0100                IEXIT=1
0101                DO 31 I=1,N
0102                Z=ALPHA*W(IS+I)
0103                IF(ABS(Z).GE.EPS(I))ICON=1
0104             31 X(I)=X(I)+Z
0105                CALL FUNCT(N,X,FY,W)
0106                IFN=IFN+1
0107                GYS=0.
0108                DO 32 I=1,N
0109             32 GYS=GYS+W(I)*W(IS+I)
0110                IF(FY.GE.F)GOTO40
0111                IF(ABS(GYS/GSO).LE..9)GOTO50
0112                IF(GYS.GT.0.)GOTO40
0113                TOT=TOT+ALPHA
0114                Z=10.
0115                IF(GS.LT.GYS)Z=GYS/(GS-GYS)
0116                IF(Z.GT.10.)Z=10.
0117                ALPHA=ALPHA*Z
0118                F=FY
0119                GS=GYS
0120                GOTO30
0121             40 CONTINUE
0122                DO 41 I=1,N
```

```
0123           41 X(I)=X(I)-ALPHA*W(IS+I)
0124              IF(ICON.EQ.0)GOTO92
0125              Z=3.*(F-FY)/ALPHA+GYS+GS
0126              ZZ=SQRT(Z**2-GS*GYS)
0127              Z=1.-(GYS+ZZ-Z)/(2.*ZZ+GYS-GS)
0128              ALPHA=ALPHA*Z
0129              GOTO30
0130           50 CONTINUE
0131              ALPHA=TOT+ALPHA
0132              F=FY
0133              IF(ICON.EQ.0)GOTO90
0134              DF=DF-F
0135              DGS=GYS-GS0
0136              LINK=1
0137              IF(DGS+ALPHA*GS0.GT.0.)GOTO52
0138              DO 51 I=1,N
0139           51 W(IU+I)=W(I)-G(I)
0140              SIG=1./(ALPHA*DGS)
0141              GOTO70
0142           52 CONTINUE
0143              ZZ=ALPHA/(DGS-ALPHA*GS0)
0144              Z=DGS*ZZ-1.
0145              DO 53 I=1,N
0146           53 W(IU+I)=Z*G(I)+W(I)
0147              SIG=1./(ZZ*DGS**2)
0148              GOTO70
0149           60 CONTINUE
0150              LINK=2
0151              DO 61 I=1,N
0152           61 W(IU+I)=G(I)
0153              IF(DGS+ALPHA*GS0.GT.0.)GOTO62
0154              SIG=1./GS0
0155              GOTO70
0156           62 CONTINUE
0157              SIG=-ZZ
0158              GOTO70
0159           65 CONTINUE
0160              DO 66 I=1,N
0161           66 G(I)=W(I)
0162              GOTO20
0163           70 CONTINUE
```

```
0164                W(IV+1)=W(IU+1)
0165                DO 71 I=2,N
0166                IJ=I
0167                I1=I-1
0168                Z=W(IU+I)
0169                DO 72 J=1,I1
0170                Z=Z-H(IJ)*W(IV+J)
0171             72 IJ=IJ+N-J
0172             71 W(IV+I)=Z
0173                IJ=1
0174                DO 75 I=1,N
0175                Z=H(IJ)+SIG*W(IV+I)**2
0176                IF(Z.LE.0.)Z=DMIN
0177                IF(Z.LT.DMIN)DMIN=Z
0178                H(IJ)=Z
0179                W(IB+I)=W(IV+I)*SIG/Z
0180                SIG=SIG-W(IB+I)**2*Z
0181             75 IJ=IJ+NP-I
0182                IJ=1
0183                DO 80 I=1,N1
0184                IJ=IJ+1
0185                I1=I+1
0186                DO 80 J=I1,N
0187                W(IU+J)=W(IU+J)-H(IJ)*W(IV+I)
0188                H(IJ)=H(IJ)+W(IB+I)*W(IU+J)
0189             80 IJ=IJ+1
0190                GOTO(60,65),LINK
0191             90 CONTINUE
0192                DO 91 I=1,N
0193             91 G(I)=W(I)
0194             92 CONTINUE
0195                IF(IPRINT.EQ.0)RETURN
0196                PRINT 1001,ITN,IFN,IEXIT
0197                PRINT 1002,F
0198                PRINT 1002,(X(I),I=1,N)
0199                PRINT 1002,(G(I),I=1,N)
0200                RETURN
0201                END
```

APPENDIX 2

A FORTRAN subroutine VA10A for minimization when only the function can be calculated explicitly

1. Purpose

To find the minimum of a function $F(\underline{x})$ of n variables $\underline{x}$. It is assumed that the function is differentiable, although it is not necessary to supply a formula for the derivatives. The method used is a quasi-Newton method in which derivatives are estimated by differences and is described in R. Fletcher, 'FORTRAN subroutines for minimization by quasi-Newton methods', A.E.R.E.R.7125 (1972). The subroutine complements VA04A but some comparisons (R. Fletcher, loc.cit) indicate that VA04A is less efficient than VA10A and more affected by round off error. VA04A also uses twice as much storage as VA10A. It is therefore suggested that VA10A be used in the first instance on any problem.

VA10A should not be used when explicit expressions are available for derivatives (use VA09A) nor when the function is a sum of squares (use VA05A, VA02A or one of the NS routines as appropriate).

2. Argument List

```
CALL VA10A(FUNCT,N,X,F,G,H,W,DFN,XM,HH,EPS,MODE,MAXFN,
           IPRINT,IEXIT)
```

FUNCT — An IDENTIFIER of the users subroutine - see section 3.

N — An INTEGER to be set to the number of variables ($N \geq 2$).

X — A REAL ARRAY of N elements in which the current estimate of the solution is stored. An initial approximation must be set in X on entry to VA10A and the best estimate obtained will be returned on exit.

F — A REAL number in which the best value of $F(\underline{x})$ corresponding to X above will be returned.

G — A REAL ARRAY of N elements which is used to store an estimate of the gradient vector $\nabla F(\underline{x})$. Not to be set on entry.

H — A REAL ARRAY of N*(N+1)/2 elements in which an estimate of the hessian matrix $\partial^2 F/(\partial x_i \partial x_j)$ is stored. The matrix is represented in the product form LDL^T where L is a lower triangular matrix with unit diagonals and D is a diagonal matrix. The lower triangle of L is stored by columns in H excepting that the unit diagonal elements are replaced by the corresponding elements of D. The setting of H on entry is controlled by the parameter MODE (q.v.).

W — A REAL ARRAY of 4*N elements used as working space.

DFN — A REAL number which must be set so as to give VA10A an estimate of the likely reduction to be obtained in $F(\underline{x})$. DFN is used only on the first iteration so an order of magnitude estimate will suffice. The information can be provided in different ways depending upon the sign of DFN which should be set in one of the following ways:

DFN>0 the setting of DFN itself will be taken as the likely reduction to be obtained in $F(\underline{x})$.

DFN=0 it will be assumed that an estimate of the minimum value of $F(\underline{x})$ has been set in argument F, and the likely reduction in $F(\underline{x})$ will be computed according to the initial function value.

DFN<0 a multiple |DFN| of the modulus of the initial function value will be taken as an estimate of the likely reduction.

XM — A REAL ARRAY of N elements to be set on entry so that XM(I) > 0 contains an indication of the magnitude of X(I). This quantity need not be set precisely as it is merely used in scaling the problem.

HH — A REAL number to be set so that HH*XM(I) contains a step length to be used in calculating G(I) by differences. Set HH equal to $2^{-t/2}$ where t is the number of significant binary digits in the calculation of F. If F contains only small errors the setting HH=1E-3 is appropriate for VA10A and HH=1E-6 for VA10AD.

EPS — A REAL number to be set on entry so that the accuracy required in X(I) is EPS*XM(I) for all I.

MODE — An INTEGER which controls the setting of the initial estimate of the hessian matrix in the parameter H. The following settings of MODE are permitted.

MODE=1 An estimate corresponding to a unit matrix is set in H by VA10A.

MODE=2 VA10A assumes that the hessian matrix itself has been set in H by columns of its lower triangle, and the conversion to LDL^T form is carried out by VA10A. The hessian matrix must be positive definite.

MODE=3 VA10A assumes that the hessian matrix has been set in H in product form. This is convenient when using the H matrix from one problem as an initial estimate for another, in which case the contents of H are passed on unchanged.

MAXFN An INTEGER set to the maximum number of calls of FUNCT permitted. Up to 2N more calls may be taken if the limit is exceeded whilst evaluating a gradient vector by differences.

IPRINT An INTEGER controlling printing. Printing occurs every |IPRINT| iterations and also on exit, in the form

Iteration No, No of calls of FUNCT,IEXIT (on exit only).
Function value
X(1),X(2),...,X(N) 8 to a line (5 in VA10AD)
G(1),G(2),...,G(N) 8 to a line (5 in VA10AD)

The values of X and G can be suppressed on intermediate iterations by setting IPRINT<0. All intermediate printing can be suppressed by setting IPRINT=MAXFN+1. All printing can be suppressed by setting IPRINT=0.

IEXIT An INTEGER giving the reason for exit from VA10A. This will be set by VA10A as follows

IEXIT=0 (MODE=2 only). The estimate of the hessian matrix is not positive definite.

IEXIT=1 The normal exit in which $|DX(I)|<EPS(I)$ for all I=1,2,...,N, where DX(I) is the change in X on an iteration.

IEXIT=2 $G^TDX>0$. Either due to rounding errors because EPS is set too small for the computer word length, or to the truncation error in the finite difference formula for G being dominant.

IEXIT=3 FUNCT called MAXFN times.

3. User Subroutine

The user must provide a subroutine headed

```
SUBROUTINE XXX(N,X,F)
REAL X(1)                    (REAL*8 in VA10AD)
```

where XXX is an identifier chosen by the user.

This subroutine should use the variables $\underline{x}$ supplied in X(1),X(2),....X(N) to evaluate the function and place it in F. XXX must be passed to VA10A as VA10A's first argument, see section 2, and appear in an EXTERNAL statement in the program that calls VA10A.

4. General

Use of COMMON: none

Workspace: N words + N*(N+1)/2 words + 4N words provided by the user in G,H and W.

Other routines: none

Input/Output: controlled by the user through IPRINT. All output is on stream 6 (line printer)

Restrictions: none

System dependence: none

Date of routine: April 1972.

May, 1972

```
0001                  SUBROUTINE VA10A(FUNCT,N,X,F,G,H,W,DFN,XM,HH,EPS,MODE,MAXFN,
                     1     IPRINT,IEXIT)
0002                  REAL X(1),G(1),H(1),W(1),XM(1)
0003                  IF(IPRINT.NE.0)PRINT 1000
0004                1000 FORMAT('1ENTRY TO VA10A'/)
0005                  NP=N+1
0006                  N1=N-1
0007                  NN=N*NP/2
0008                  IS=N
0009                  IU=N
0010                  IV=N+N
0011                  IB=IV+N
0012                  IDIFF=1
0013                  IEXIT=0
0014                  IF(MODE.EQ.3)GOTO15
0015                  IF(MODE.EQ.2)GOTO10
0016                  IJ=NN+1
0017                  DO 5 I=1,N
0018                  DO 6 J=1,I
0019                  IJ=IJ-1
0020                6 H(IJ)=0.
0021                5 H(IJ)=1.
0022                  GOTO15
0023               10 CONTINUE
0024                  IJ=1
0025                  DO 11 I=2,N
0026                  Z=H(IJ)
0027                  IF(Z.LE.0.)RETURN
0028                  IJ=IJ+1
0029                  I1=IJ
0030                  DO 11 J=I,N
0031                  ZZ=H(IJ)
0032                  H(IJ)=H(IJ)/Z
0033                  JK=IJ
0034                  IK=I1
0035                  DO 12 K=I,J
0036                  JK=JK+NP-K
0037                  H(JK)=H(JK)-H(IK)*ZZ
0038               12 IK=IK+1
0039               11 IJ=IJ+1
0040                  IF(H(IJ).LE.0.)RETURN
```

```
0041            15 CONTINUE
0042               IJ=NP
0043               DMIN=H(1)
0044               DO 16 I=2,N
0045               IF(H(IJ).GE.DMIN)GOTO16
0046               DMIN=H(IJ)
0047            16 IJ=IJ+NP-I
0048               IF(DMIN.LE.0.)RETURN
0049               Z=F
0050               ITN=0
0051               CALL FUNCT(N,X,F)
0052               IFN=1
0053               DF=DFN
0054               IF(DFN.EQ.0.)DF=F-Z
0055               IF(DFN.LT.0.)DF=ABS(DF*F)
0056               IF(DF.LE.0.)DF=1.
0057            17 CONTINUE
0058               DO 19 I=1,N
0059            19 W(I)=X(I)
0060               LINK=1
0061               IF(IDIFF-1)100,100,110
0062            18 CONTINUE
0063               IF(IFN.GE.MAXFN)GOTO90
0064            20 CONTINUE
0065               IF(IPRINT.EQ.0)GOTO21
0066               IF(MOD(ITN,IPRINT).NE.0)GOTO21
0067               PRINT 1001,ITN,IFN
0068          1001 FORMAT(24I5)
0069               PRINT 1002,F
0070          1002 FORMAT((8E15.7))
0071               IF(IPRINT.LT.0)GOTO21
0072               PRINT 1002,(X(I),I=1,N)
0073               PRINT 1002,(G(I),I=1,N)
0074            21 CONTINUE
0075               ITN=ITN+1
0076               W(1)=-G(1)
0077               DO 22 I=2,N
0078               IJ=I
0079               I1=I-1
0080               Z=-G(I)
0081               DO 23 J=1,I1
```

```
0082                Z=Z-H(IJ)*W(J)
0083             23 IJ=IJ+N-J
0084             22 W(I)=Z
0085                W(IS+N)=W(N)/H(NN)
0086                IJ=NN
0087                DO 25 I=1,N1
0088                IJ=IJ-1
0089                Z=0.
0090                DO 26 J=1,I
0091                Z=Z+H(IJ)*W(IS+NP-J)
0092             26 IJ=IJ-1
0093             25 W(IS+N-I)=W(N-I)/H(IJ)-Z
0094                Z=0.
0095                GS0=0.
0096                DO 29 I=1,N
0097                IF(Z*XM(I).GE.ABS(W(IS+I)))GOTO29
0098                Z=ABS(W(IS+I))/XM(I)
0099             29 GS0=GS0+G(I)*W(IS+I)
0100                AEPS=EPS/Z
0101                IEXIT=2
0102                IF(GS0.GE.0.)GOTO92
0103                ALPHA=-2.*DF/GS0
0104                IF(ALPHA.GT.1.)ALPHA=1.
0105                FF=F
0106                TOT=0.
0107                INT=0
0108                IEXIT=1
0109             30 CONTINUE
0110                IF(IFN.GE.MAXFN)GOTO90
0111                DO 31 I=1,N
0112             31 W(I)=X(I)+ALPHA*W(IS+I)
0113                CALL FUNCT(N,W,F1)
0114                IFN=IFN+1
0115                IF(F1.GE.F)GOTO40
0116                F2=F
0117                TOT=TOT+ALPHA
0118             32 CONTINUE
0119                DO 33 I=1,N
0120             33 X(I)=W(I)
0121                F=F1
0122                IF(INT-1)35,49,50
```

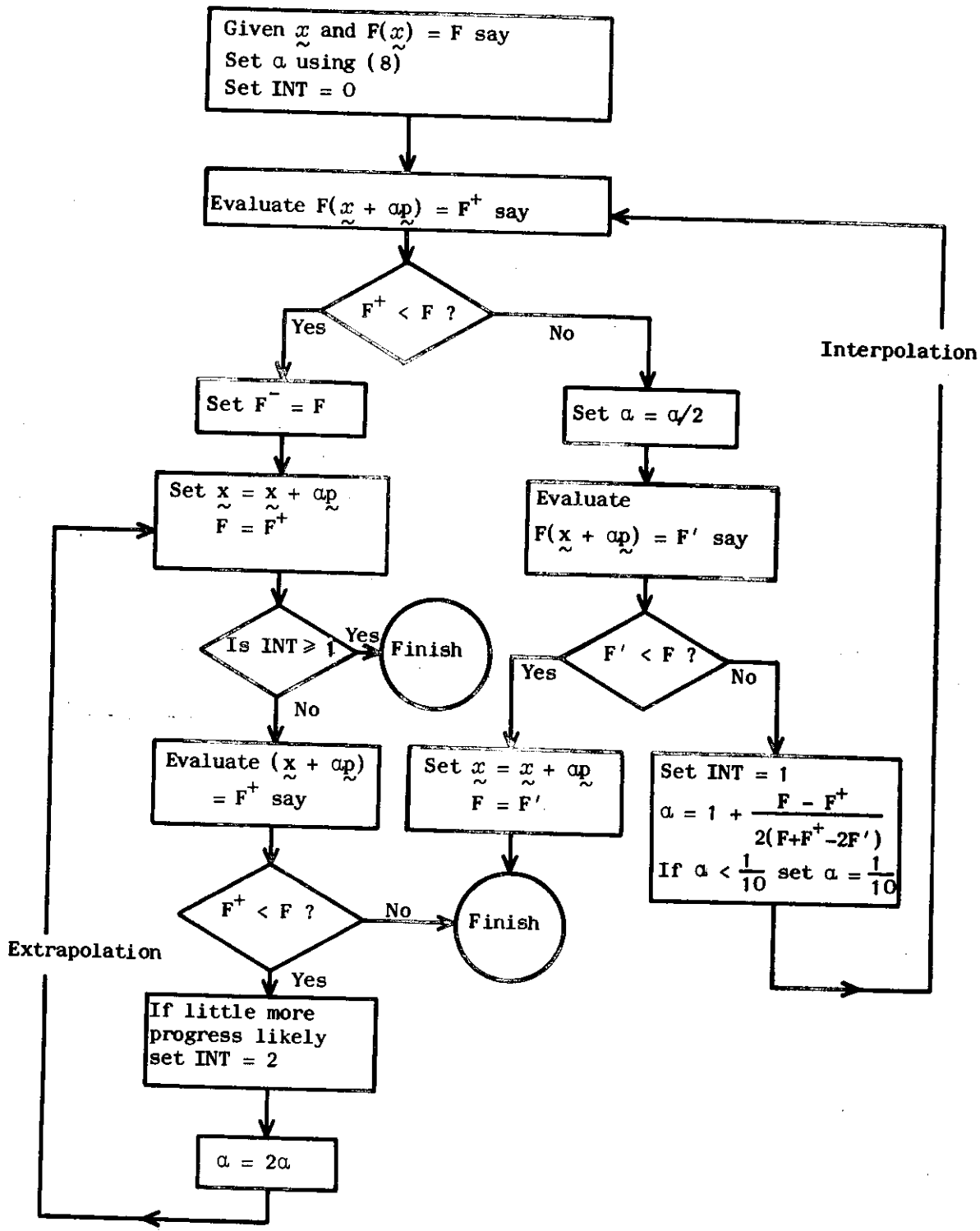
```
0123          35 CONTINUE
0124             IF(IFN.GE.MAXFN)GOTO90
0125             DO 34 I=1,N
0126          34 W(I)=X(I)+ALPHA*W(IS+I)
0127             CALL FUNCT(N,W,F1)
0128             IFN=IFN+1
0129             IF(F1.GE.F)GOTO50
0130             IF(F1+F2.GE.F+F.AND.7.*F1+5.*F2.GT.12.*F)INT=2
0131             TOT=TOT+ALPHA
0132             ALPHA=2.*ALPHA
0133             GOTO32
0134          40 CONTINUE
0135             IF(ALPHA.LT.AEPS)GOTO92
0136             IF(IFN.GE.MAXFN)GOTO90
0137             ALPHA=.5*ALPHA
0138             DO 41 I=1,N
0139          41 W(I)=X(I)+ALPHA*W(IS+I)
0140             CALL FUNCT(N,W,F2)
0141             IFN=IFN+1
0142             IF(F2.GE.F)GOTO45
0143             TOT=TOT+ALPHA
0144             F=F2
0145             DO 42 I=1,N
0146          42 X(I)=W(I)
0147             GOTO49
0148          45 CONTINUE
0149             Z=.1
0150             IF(F1+F.GT.F2+F2)Z=1.+.5*(F-F1)/(F+F1-F2-F2)
0151             IF(Z..T..1)Z=.1
0152             ALPHA=Z*ALPHA
0153             INT=1
0154             GOTO30
0155          49 CONTINUE
0156             IF(TOT.LT.AEPS)GOTO92
0157          50 CONTINUE
0158             ALPHA=TOT
0159             DO 56 I=1,N
0160             W(I)=X(I)
0161          56 W(IB+I)=G(I)
0162             LINK=2
0163             IF(IDIFF-1)100,100,110
```

```
0164          54 CONTINUE
0165             IF(IFN.GE.MAXFN)GOTO90
0166             GYS=0.
0167             DO 55 I=1,N
0168             W(I)=W(IB+I)
0169          55 GYS=GYS+G(I)*W(IS+I)
0170             DF=FF-F
0171             DGS=GYS-GS0
0172             IF(DGS.LE.0.)GOTO20
0173             LINK=1
0174             IF(DGS+ALPHA*GS0.GT.0.)GOTO52
0175             DO 51 I=1,N
0176          51 W(IU+I)=G(I)-W(I)
0177             SIG=1./(ALPHA*DGS)
0178             GOTO70
0179          52 CONTINUE
0180             ZZ=ALPHA/(DGS-ALPHA*GS0)
0181             Z=DGS*ZZ-1.
0182             DO 53 I=1,N
0183          53 W(IU+I)=Z*W(I)+G(I)
0184             SIG=1./(ZZ*DGS**2)
0185             GOTO70
0186          60 CONTINUE
0187             LINK=2
0188             DO 61 I=1,N
0189          61 W(IU+I)=W(I)
0190             IF(DGS+ALPHA*GS0.GT.0.)GOTO62
0191             SIG=1./GS0
0192             GOTO70
0193          62 CONTINUE
0194             SIG=-ZZ
0195          70 CONTINUE
0196             W(IV+1)=W(IU+1)
0197             DO 71 I=2,N
0198             IJ=I
0199             I1=I-1
0200             Z=W(IU+I)
0201             DO 72 J=1,I1
0202             Z=Z-H(IJ)*W(IV+J)
0203          72 IJ=IJ+N-J
0204          71 W(IV+I)=Z
```

```
0205              IJ=1
0206              DO 75 I=1,N
0207              Z=H(IJ)+SIG*W(IV+I)**2
0208              IF(Z.LE.0.)Z=DMIN
0209              IF(Z.LT.DMIN)DMIN=Z
0210              H(IJ)=Z
0211              W(IB+I)=W(IV+I)*SIG/Z
0212              SIG=SIG-W(IB+I)**2*Z
0213           75 IJ=IJ+NP-I
0214              IJ=1
0215              DO 80 I=1,N1
0216              IJ=IJ+1
0217              I1=I+1
0218              DO 80 J=I1,N
0219              W(IU+J)=W(IU+J)-H(IJ)*W(IV+I)
0220              H(IJ)=H(IJ)+W(IB+I)*W(IU+J)
0221           80 IJ=IJ+1
0222              GOTO(60,20),LINK
0223           90 CONTINUE
0224              IEXIT=3
0225              GOTO94
0226           92 CONTINUE
0227              IF(IDIFF.EQ.2)GOTO94
0228              IDIFF=2
0229              GOTO17
0230           94 CONTINUE
0231              IF(IPRINT.EQ.0)RETURN
0232              PRINT 1001,ITN,IFN,IEXIT
0233              PRINT 1002,F
0234              PRINT 1002,(X(I),I=1,N)
0235              PRINT 1002,(G(I),I=1,N)
0236              RETURN
0237          100 CONTINUE
0238              DO 101 I=1,N
0239              Z=HH*XM(I)
0240              W(I)=W(I)+Z
0241              CALL FUNCT(N,W,F1)
0242              G(I)=(F1-F)/Z
0243          101 W(I)=W(I)-Z
0244              IFN=IFN+N
0245              GOTO(18,54),LINK
```

```
0246          110 CONTINUE
0247              DO 111 I=1,N
0248              Z=HH*XM(I)
0249              W(I)=W(I)+Z
0250              CALL FUNCT(N,W,F1)
0251              W(I)=W(I)-Z-Z
0252              CALL FUNCT(N,W,F2)
0253              G(I)=(F1-F2)/(2.*Z)
0254          111 W(I)=W(I)+Z
0255              IFN=IFN+N+N
0256              GOTO(18,54),LINK
0257              END
```

Figure 1



Linear search subproblem for the no derivative problem