1 SUMMARY

This subroutine is used to calculate the least value of a function of several variables subject to general constraints on the values of the variables.

The objective function is

$$F(x_1, x_2, ..., x_n)$$

and the constraints are

$$c_i(x_1, x_2, ..., x_n) = 0, \qquad i=1,2,...,m'$$

and

$$c_i(x_1, x_2, ..., x_n) \geq 0, \qquad i=m'+1,\ m'+2,...,m.$$

The user is required to provide initial values of the variables and code for evaluating F and c_i and their first derivatives $\partial F/\partial x_j$, $\partial c_i/\partial x_j$, i=1,2,...,m and j=1,2,...,n. It is sometimes necessary to scale the problem so that the values of the variables and the derivative vectors of the functions F and c_i, i=1,2,...,m, all have magnitude about unity.

An iterative method is used. Each iteration minimizes a quadratic approximation to the Lagrangian function subject to linear approximations to the constraints. The second derivative matrix of the Lagrangian function is estimated automatically. A line search procedure utilising the 'watchdog technique' is used to force convergence when the initial values of the variables are far from the solution.

ATTRIBUTES — Version: 1.0.0. **Types:** VF13A; VF13AD. **Calls:** VE17. **Original date:** April 1987. **Origin:** M.J.D.Powell, Cambridge University. **Licence:** A third-party licence for this package is available without charge. **Remark:** This is an improved version of VF12 and supersedes it. It also supersedes VF03.

2 HOW TO USE THE PACKAGE

'Reverse Communication' is used to provide VF13A/AD with values of F, c_i, $\partial F/\partial x_j$ and $\partial c_i/\partial x_j$, i=1,2,...,m and j=1,2,...,n, which means that computer instructions for their calculation must be present in the part of the user's program that calls VF13A/AD initially. When the subroutine requires the values of these terms, it sets the appropriate values of the variables in the vector X and it returns to the user's program with a flag, called INF, set to zero. The user's program must set the values of F, c_i, $\partial F/\partial x_j$ and $\partial c_i/\partial x_j$, i=1,2,...,m and j=1,2,...,n, in F, C, G and CN respectively, and it may not alter the values of any of the other arguments of VF13A/AD. Then VF13A/AD is called again with INF still equal to zero. The initial call of VF13A/AD is indicated by INF<0. Before the initial call it is necessary to calculate F, C, G and CN for the initial value of X. An example is shown in Section 5.

2.1 Argument list

The single precision version

```
      CALL VF13A(N,M,MEQ,X,F,G,C,CN,LCN,MAXFUN,
     *           ACC,IPRINT,INF,W,LW,IW)
```

The double precision version

```
      CALL VF13AD(N,M,MEQ,X,F,G,C,CN,LCN,MAXFUN,
     *            ACC,IPRINT,INF,W,LW,IW)
```

N is an INTEGER variable that must be set by the user to the number, n, of variables. Its value is not altered by the subroutine. **Restriction**: N>0.

M is an INTEGER variable that must be set by the user to the number, m, of constraints on the variables. Its value is not altered by the subroutine. A zero value is allowed. **Restriction**: M≥0.

MEQ is an INTEGER variable that must be set by the user to the number, m', of equality constraints on the variables. Its value is not altered by the subroutine. Of course, a zero value is allowed. **Restriction**: M≥MEQ≥0.

X is a REAL (DOUBLE PRECISION in the D version) array of length n for the values of the variables. Initially it must contain the initial estimates of the variables. On a return from the subroutine with INF=0, the array contains the variables for the required calculation of F, c_i, i=1,2,...,m and their first derivatives. On a return with INF>0, the array contains the final values of the variables.

F is a REAL (DOUBLE PRECISION in the D version) variable, which the user must set to $F(x_1, x_2, ..., x_n)$ each time the subroutine is called. On a return with INF>0, F contains the final value of the objective function.

G is a REAL (DOUBLE PRECISION in the D version) array of length n, in which the user must set the first derivatives $\partial F/\partial x_j$, j=1,2,...,n, each time the subroutine is called. On a return with INF>0, G contains the final values of these first derivatives.

C is a REAL (DOUBLE PRECISION in the D version) array of length m, in which the user must set the values of the constraint functions $c_i(x_1, x_2, ..., x_n)$, i=1,2,...,m, each time the subroutine is called. On a return with INF>0, C provides the final values of the constraint functions.

CN is a REAL (DOUBLE PRECISION in the D version) array of dimensions (LCN,M). The user must set CN(J,I) to the value of $\partial c_i/\partial x_j$, i=1,2,...,m and j=1,2,...,n, each time the subroutine is called. On a return with INF>0 the array contains the final values of these derivatives. The subroutine makes use of the space CN(N+1,I), I=1,2,...,M.

LCN is an INTEGER variable which the user must set to the first dimension of CN. Its value must be at least (N+1), and it is not altered by the subroutine.

MAXFUN is an INTEGER variable that limits the number of function and gradient evaluations that are called for. An error return is made if this number exceeds MAXFUN. The value of MAXFUN is not altered by the subroutine.

ACC is a REAL (DOUBLE PRECISION in the D version) variable that must be set by the user, that controls the accuracy of the calculation, and that is not altered by the subroutine. The calculation finishes when the objective function is predicted to be within ACC of its final value, and allowance is made for the expected changes in $F(x_1, x_2, ..., x_{n)}$ due to any constraint violations. Thus constraint violations are controlled also.

IPRINT is an INTEGER variable that must be set by the user to indicate how much printing is required, and that is not altered by the subroutine. There is no output if IPRINT is negative, only diagnostics are printed if it is zero, and, if IPRINT is positive, then, in addition to any diagnostics, the values of x_i, i=1,2,...,n, $F(x_1, x_2, ..., x_{n)}$ and c_i, i=1,2,...,m, are printed every IPRINT iterations.

INF is an INTEGER variable that must be set by the user to a negative value initially, in order to indicate the start of the calculation. The value of INF on each return indicates the reason for the return. If it is zero, then the user's program has to provide new values of F, c_i, $\partial F/\partial x_j$ and $\partial c_i/\partial x_j$, and then recall the subroutine with INF still set to zero. If INF is positive the calculation is finished. It has the value INF=1 if the required accuracy is achieved, but it may also have the values 2,3,...,8. They indicate error conditions, and are explained in Section 2.4 below. INF also allows options that control step-lengths and constraint violations. These options are invoked with the special initial values INF=−101, INF=−110 and INF=−111, and are explained in Section 2.5 below. Any other negative value of INF causes default values to be set initially.

W is a REAL (DOUBLE PRECISION in the D version) array of length LW that is used for workspace. The number of elements that are used is $5n^2/2+43n/2+14+6m$. Note that $(6m+1500)$ storage locations are sufficient for values of n less than 20.

LW is an INTEGER variable that must be set by the user to the length of the array W. Its value is not altered by the subroutine, and it must be sufficient for the working space that is required.

IW is an INTEGER array of length *n*+1 that is used for workspace.

2.3 Common

The following two common blocks are included:

The single precision version

```
COMMON/VF13D/ BOUND,LP
COMMON/VF13E/ DUMMY(14),IDUMMY(11)
```

The double precision version

```
COMMON/VF13DD/ BOUND,LP
COMMON/VF13ED/ DUMMY(14),IDUMMY(11)
```

BOUND is a REAL (DOUBLE PRECISION in the D version) variable with default value 1.0D6 set by BLOCK DATA VF13F/FD, which limits the change made to any variable during an iteration unless INF=-101 or INF=-111 initially.

LP is an INTEGER, with default value 6 set by BLOCK DATA VF13F/FD, which gives the stream number for messages.

DUMMY is a REAL (DOUBLE PRECISION in the D version) array of length 14 used to hold private variables. It must not be altered between calls for the same problem.

IDUMMY is an INTEGER array of length 11 used to hold private variables. It must not be altered between calls for the same problem.

2.4 Errors and diagnostic messages

The required accuracy is achieved when INF=1, however, the following error returns may occur:

INF=2 The limit on the calculation due to the value of MAXFUN has been reached. If this seems to be because the subroutine is changing the variables very slowly, you should seek advice. Otherwise, increase the value of MAXFUN.

INF=3 The line search procedure requires more than five steps. This error is usual if the derivatives are coded incorrectly, because then the objective function of the line search may increase along a direction which, according to the first derivatives, is a direction of descent. Therefore, you should check the derivatives carefully, paying particular attention to the components along the search direction that caused the error return, which is the straight line through the last five values of $x_1, x_2, ..., x_n$ at which F, c_i, i=1,2,...,m, and their derivatives were calculated.

This error return will also occur when the required accuracy cannot be achieved, in which case you will find that the changes in function values are dominated by computer rounding errors. Thirdly, if the error return occurs very early in the calculation there is a possibility that an initial automatic choice of VF13A/AD is inappropriate, namely setting the initial estimate of the second derivative matrix of the Lagrangian function to the unit matrix. The remedy is to scale the variables and $F(x_1, x_2, ..., x_{n)}$ so that first derivatives and expected changes in variables have magnitude about one.

INF=4 An uphill search direction is calculated. This can only be due to computer rounding errors, but sometimes their effect is exaggerated by the auxiliary subroutine VE17AD. The required accuracy cannot be achieved.

INF=5 The subroutine has failed to find a vector of variables that satisfies the constraints. It occurs if the constraints are inconsistent, but it may also occur if the constraint gradients are calculated incorrectly, or if the constraints are highly curved and there are large constraint violations at the initial value of $x_1, x_2, ..., x_n$, or if the problem is badly scaled.

INF=6 The variable metric matrix that is used to calculate the search direction is not positive definite. This can only

occur through computer rounding errors and may be because the problem has an unbounded solution or because the problem is badly scaled.

INF=7 The array W is too small. The printed diagnostic indicates the minimum satisfactory size.

INF=8 One of the conditions $n>0$, $m\geq m'\geq 0$ is violated by the values input in N, M, MEQ.

2.5 Options for controlling step-lengths and constraint violations

Any negative value of INF may be used to signal the start of the calculation but the following special values allow the user to initialise some parameters that are used to control step-lengths and constraint violations.

INF=-101 Bounds on the magnitude of the changes in the variables x_i, i=1,2,...,n, that will be allowed in a single iteration must be set by the user in the array elements W(M+I), I=1,2,...,N, before the first call to VF13A/AD. If this option is not invoked then default values W(M+I)=BOUND, I=1,2,...,N, are used, as described in Section 2.3.

INF=-110 The user must set the initial values of the elements W(K), K=1,2,...,M, before the first call to VF13A/AD. These values are used to weight constraint violations and they are increased automatically by the algorithm, if necessary, but they are never decreased. Large values have the effect of keeping constraint violations small but efficiency may be impaired if the initial values are too large. If this option is not invoked then default values W(K)=1.0D-20, K=1,2,...,M, are used.

INF=-111 This option combines both of the above options.

3 GENERAL INFORMATION

Use of common: The subroutine uses common blocks VF13D/DD and VF13E/ED and BLOCK DATA VF13F/FD.

Other routines called directly: VE17 is called.

Input/output: Output is under the control of argument LP.

Restrictions: M≥MEQ≥0, N>0.

4 METHOD

VF13 is a revised version of VF03 which calls the more efficient quadratic programming subroutine VE17 in place of VF02. A description of the method is given by M.J.D.Powell, 'Extensions to subroutine VF02', in System Modeling and Optimization, Lecture Notes in Control and Information Sciences 38, eds. R.F. Drenick and F. Kozin, Springer-Verlag (Berlin) (1982), pp.529-538. and by R.M. Chamberlain, C. Lemarechal, H.C. Pedersen and M.J.D. Powell in 'The watchdog technique for forcing convergence in algorithms for constrained optimization', Mathematical Programming Study 16 (1982), pp.1-17.

The quadratic programming procedure used is described by M.J.D. Powell in 'On the quadratic programming algorithm of Goldfarb and Idnani', Mathematical Programming Study 25 (1985), pp.46-61.

The algorithm can be regarded as an extension of a variable metric method for unconstrained optimization to the constrained case. In methods of this type a positive definite matrix occurs as the second derivative matrix of a quadratic objective function, whose first derivative at the starting point of an iteration is made equal to the gradient of $F(x_1,x_2,...,x_n)$. The direction to the minimum of the quadratic is the search direction of the iteration in the space of the variables. The positive definite matrix is revised automatically so that the final rate of convergence of the whole process is superlinear. When constraints are present in the main calculation, then linear approximations to the constraints are included in the part of the process that calculates the search direction. Thus this calculation is a convex quadratic programming problem. The revision of the positive definite matrix takes account of any constraint curvature.

Constraint violations are controlled in each iteration by forcing a reduction in a line search objective function. Two

line search functions are employed because use of the standard line search objective function sometimes causes curved constraint boundaries to be followed very closely, impeding substantial progress. Therefore, the watchdog technique allows a relaxed line search to be used on some iterations. When the relaxed line search is used, constraint violations may increase initially but usually a much better approximation is obtained subsequently. In case the relaxed line search does not eventually provide a better approximation the watchdog technique incorporates a backtracking facility to the best point reached so far. When backtracking occurs the relaxed line search is not used again for the next ten iterations. Thus convergence is still assured but efficiency may be greatly increased on difficult problems.

An advantage of the Variable Metric method over the Augmented Lagrangian and Penalty Function methods is that the linear approximations to the constraints restrict the freedom in the calculation of the search direction in a way that is sensible and that does not depend on the current positive definite matrix. Thus, when many constraints are active at the final solution, less information is required from the positive definite matrix, so it becomes easier to find one that is suitable. Hence very large gains over the number of function and gradient evaluations that are required by `VF01A/AD` are obtained when the number of active constraints is greater than $\frac{1}{2}n$. However, the work of solving a quadratic programming problem on each iteration is considerably greater than the work per line search of `VF01A/AD`, so the saving in iterations is worthwhile only when most of the computing time is spent in calculating functions and their first derivatives.

5 EXAMPLE OF USE

We illustrate use of the subroutine on the problem with objective function

$$x_1^2 + x_2^2 + x_3$$

and constraints

$$x_1 x_2 = x_3$$

$$x_3 \geq 1$$

taking the point (1,2,3) as an initial guess. The following code finds the solution (1,1,1) after 19 calls of VF13A/AD.

```
      DOUBLE PRECISION X(3),F,G(3),C(2),CN(10,2),ACC,W(200)
      INTEGER IW(18)
      LW=200
      LCN=10
      N=3
      M=2
      MEQ=1
      MAXFUN=100
      ACC=1.0D-7
      IPRINT=1
      INF=-1
      DO 10 I=1,N
        X(I)=DBLE(I)
   10 CONTINUE
   20 F=X(1)**2+X(2)**2+X(3)
      G(1)=2.0D0*X(1)
      G(2)=2.0D0*X(2)
      G(3)=1.0D0
      C(1)=X(1)*X(2)-X(3)
      CN(1,1)=X(2)
      CN(2,1)=X(1)
      CN(3,1)=-1.0D0
      C(2)=X(3)-1.0D0
      CN(1,2)=0.0D0
      CN(2,2)=0.0D0
      CN(3,2)=1.0D0
      CALL VF13AD(N,M,MEQ,X,F,G,C,CN,LCN,
     +            MAXFUN,ACC,IPRINT,INF,W,LW,IW)
      IF(INF.EQ.0)GO TO 20
      STOP
      END
```