1 SUMMARY

To find the **minimum of a general function** $f(\mathbf{x})$ of n variables $\mathbf{x}=x_1,x_2,...,x_n$ subject to the variables satisfying $2n$ **simple bounds**

$$l_i \le x_i \le u_i \qquad i=1,2,...,n$$

and $m-2n$ **linear constraints**

$$\sum_{j=1}^{n} c_{ji}x_j \ge d_i \qquad i=1,2,...,m-2n$$

where any of the bounds and constraints may be made strict equalities. **Derivatives** $\partial f/\partial x_j$ j=1, 2,..., n are required.

The method is given in R.Fletcher, AERE TP.431. There are several modes of operation and certain modes require the user to supply an initial feasible solution. Scaling can be specified and a subroutine must be provided to compute values of the function $f(\mathbf{x})$ and its derivatives.

ATTRIBUTES — Version: 1.0.0. **Remark:** See also `VE01`. **Types:** `VE03A; VE03AD`. **Calls:** `FD05`, `LA02`, `_DOT` and `VE02`. **Original date:** June 1971. **Origin:** R.Fletcher, Harwell.

2 HOW TO USE THE PACKAGE

The method used by `VE03` is iterative and requires an initial feasible point $\mathbf{x}$ to be calculated (i.e. one which satisfies all the bounds and linear constraints). Such a point can be supplied by the user or calculated automatically — see Section 2.5. The user must provide a subroutine to calculate $f(\mathbf{x})$ for any feasible x, and also the vector of first derivatives $\mathbf{g}(\mathbf{x})$ (where $g_i(\mathbf{x})=\partial f/\partial x_i$) — see Section 2.2.

The subroutine returns to the user the best feasible point $\mathbf{x}$ which has been found, and the corresponding $f(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$. Some information about which constraints are *active* (i.e. are equalities at the solution and therefore restricting the function from being reduced) is also given, together with the residuals of the linear constraints.

If a sequence of problems is being solved in which $f(\mathbf{x})$ is modified slightly each time, but the bounds and constraints remain the same, then increased efficiency is possible — see Section 2.5.

2.1 Argument list

The single precision version

```
      CALL VE03A(FUNCT,N,M,C,IC,D,BDL,BDU,
     *  F,G,GM,IGM,CN,X,K,KE,H,IH,LT,HL,LP,S,
     *  HK,HH,EPS,MAXFN,IPRINT,MODE)
```

The double precision version

```
      CALL VE03AD(FUNCT,N,M,C,IC,D,BDL,BDU,
     *  F,G,GM,IGM,CN,X,K,KE,H,IH,LT,HL,LP,S,
     *  HK,HH,EPS,MAXFN,IPRINT,MODE)
```

`FUNCT` the name of a subroutine provided by the user described in Section 2.2. It must be declared in an `EXTERNAL` statement in the calling program.

`N` an `INTEGER` variable set by the user to the total number of variables n.

`M` an `INTEGER` variable set by the user to the total number of constraints. The upper and lower bounds each count as n constraints and must be given explicitly, so $m \ge 2n$.

C is a REAL two-dimensional array to be set to the coefficients on the left hand side of the constraints c_{ji}, j=1, 2,..., n and i=1, 2,..., m–2n. C must have n rows and m–2n columns. The i-th column of C contains the coefficient vector $\mathbf{c}_i$ of the constraint $\mathbf{c}_i^T\mathbf{x} \geq d_i$. If there are no general constraints, whence m=2n, C will not be referred to and a dummy argument (GM say) can be inserted in the calling sequence.

IC is an INTEGER variable which must be set by the user to the first dimension of C in the dimension statement which assigns space to C. **Restriction:** IC ≥ n.

D is a REAL (DOUBLE PRECISION in the D version) array of m–2n elements which must be set by the user to d_i, i=1, 2,..., m–2n, the right hand sides of the constraints. If m=2n, D is not referred to and a dummy argument (G say) can be inserted in the calling sequence.

BDL is a REAL (DOUBLE PRECISION in the D version) array of n elements which must be set by the user to the lower bounds l_i, i=1, 2,..., n, e.g. BDL(I) is set to the lower bound l_i on the i-th variable. All n elements must be set. Any elements corresponding to unbounded variables should be set to a large negative value, such as -10^{30}.

BDU is a REAL (DOUBLE PRECISION in the D version) array of n elements which must be set by the user to the upper bounds on the variables u_i, i=1, 2,..., n, e.g. BDU(i) is set to the upper bound on the i-th variable. All n elements must be set. Any elements corresponding to unbounded variables should be set to a large positive value, such as 10^{30}.

F is a REAL (DOUBLE PRECISION in the D version) variable which on exit from VE03 contains the least value of $f(\mathbf{x})$ which has been found.

G is a REAL (DOUBLE PRECISION in the D version) array of n elements which on exit from VE03 contains the derivatives $\mathbf{g}(\mathbf{x})$ which correspond to F above.

GM is a REAL (DOUBLE PRECISION in the D version) two-dimensional array of at least n rows and n columns representing the $n\times n$ symmetric matrix Γ (see Harwell report TP 431) which approximates the Hessian matrix of second derivatives of $f(\mathbf{x})$. If MODE=1 or 2 (see Section 2.5) all elements of GM(I,J), for all I,J=1, 2, ..., N must be set, and GM(I,J) must equal GM(J,I) for all I,J. Any information is useful, but failing this set Γ equal to the null matrix or the identity matrix and the approximation will be improved automatically. If MODE=3, see Section 2.5.

IGM is an INTEGER variable which must be set by the user to the first dimension of GM in the dimension statement which assigns space to GM. **Restriction:** IGM ≥ n.

CN is a REAL (DOUBLE PRECISION in the D version) array of m–2n elements which on exit from VE03 contains the residuals of the linear constraints, that is CN(I) contains $\mathbf{c}_i^T\mathbf{x}-d_i$ where $\mathbf{x}$ is the contents of X below. When m=2n, CN is not set and a dummy (G say) can be supplied in the calling sequence.

X is a REAL (DOUBLE PRECISION in the D version) array of length at least n which is set depending on the value of MODE. If MODE=1 the user must set X to a feasible point, e.g. a feasible point $\mathbf{x}=x_1,x_2,...,x_n$ must be set in X(1), X(2), ...,X(N). If MODE=2 or 3 refer to Section 2.5 for setting instructions.

On exit from VE03, X contains the point $\mathbf{x}$ for which $f(\mathbf{x})$ is the least function value found.

K is an INTEGER variable which must be set to zero if MODE=1. If MODE=2 or 3 see section 2.5. On exit from VE03, K gives the number of active constraints.

KE is an INTEGER variable set to the combined number of bounds and constraints which are to be designated as equalities. If KE > 0, use of MODE=1 is not permitted. If MODE=3 see section 2.5.

H is a REAL (DOUBLE PRECISION in the D version) two-dimensional array of at least 2n rows and 2n columns used as working space. If MODE=3 see section 2.5.

IH is an INTEGER variable which must be set by the user to the first dimension of H in the dimension statement which assigns space to H. **Restriction:** IH ≥ 2n.

LT is an INTEGER array of at least 2n+m elements, partly used as working space. It need not be set unless MODE=2

or 3, in which case see section 2.5.

On exit from VE03, the first K elements of LT will give the index numbers of the active constraints in the following order: lower bounds $1 \rightarrow n$, upper bounds $n+1 \rightarrow 2n$, constraints $2n+1 \rightarrow m$.

HL is a REAL (DOUBLE PRECISION in the D version) array of n^2 elements used as working space. Not to be set unless MODE=3 whence see section 2.5.

LP is an INTEGER array of at least n elements used as working space. Not to be set unless MODE=3 whence see section 2.5.

S is a REAL (DOUBLE PRECISION in the D version) array of at least n elements which controls scaling and is to be set in accordance with section 2.5.

HK is a REAL (DOUBLE PRECISION in the D version) variable used as an upper bound to the change made in any scaled variable. Must be set on entry: typically to 0.1 of the largest estimated error in any scaled variable. HK < 2×EPS. HK will be adjusted suitably during the iteration, and its setting is not critical. It is altered by the subroutine.

HH is a REAL (DOUBLE PRECISION in the D version) variable to be set to an estimate of the maximum error in any scaled variable. Set HH on the large side if in doubt. Again its setting is not critical.

EPS is a REAL (DOUBLE PRECISION in the D version) variable to be set to the absolute accuracy required in all the scaled variables.

MAXFN is an INTEGER variable set to the maximum number of times which FUNCT can be called.

IPRINT is an INTEGER variable set to control printing — see Section 2.4.

MODE is an INTEGER variable set to control the mode of operation of VE03 — see Section 2.5.

2.2 The user subroutine

The user must supply a subroutine, to which he assigns an identifier of his choice (XYZ say). This should be as follows:

```
SUBROUTINE XYZ(N,X,F,G)
DIMENSION X(*),G(*)
   ......
     statements to evaluate f(x) and g(x) where x_i is given in X(i). f(x) should be left in F and g_i(x)=∂f/∂x_i in
     G(I) for all i=1,2,...,n.
   .....
RETURN
END
```

If the derivation of **g** is at all complicated, the user is strongly advised to check the derivatives by differences using relationships such as

$$g_1(\mathbf{x}) = \frac{f(\mathbf{x}+h\mathbf{e}_1) - f(\mathbf{x})}{h}$$

where $\mathbf{e}_1$ is the vector (1,0,0,...,0), and so on.

2.3 Printing

Printing starts on a new page with the text ENTRY TO VE03. At the beginning of the first iteration, and on every subsequent IPRINT iterations the numbers

```
IT    ICT
F
X(1),X(2),...,X(N)          (8 to a line)
G(1),G(2),...,G(N)          (8 to a line)
```

are printed as shown. In addition to the definitions of Section 2.1 IT is the previous number of iterations, and ICT the number of times the user subroutine has been called. The same information is printed out on exit, with additionally

```
CN(1),CN(2),...,CN(M-2*N)        (8 to a line)
```

if M > 2*N. However no printing at all will take place if IPRINT=0. If there is no feasible point to the constraints the diagnostic NO FEASIBLE POINT will be printed.

2.4 Scaling

The variables in the problem may be scaled so that changes of similar magnitude are made in each, and this can be expected to improve the rate of convergence. The user supplies a vector S(1), S(2),...,S(N) and the subroutine scales any changes, DL(I) say, in the variables by dividing by S(I). The largest element of S(I) should be 1, and all elements should be greater than 0. Thus the size of the elements of S(I) should be proportional to the likely changes to the variables.

2.5 Modes of operation

The subroutine will operate in three modes to the setting of the parameter MODE.

MODE=1 is the usual setting if the user can supply a feasible point, and if all the constraints are inequalities.

MODE=2 should be set if a feasible point is required to be determined automatically.

MODE=3 concerns the case when a similar problem has just been solved with a slightly modified $f(x)$ but with the same constraints.

When using MODE=2 the setting of parameters X, K, KE and LT should follow the advice given in the specification sheet for setting these parameters on entry to the feasible point subroutine LA02. When using MODE=3 the contents of GM, X, K, KE, H, LT, HL and LP should be left undisturbed from the previous call of VE03.

3 GENERAL INFORMATION

Use of common: None.

Workspace: See under 'restrictions' below.

Other routines called directly: VE02A/AD, LA02A/AD, FD05, and _DOT.

Restrictions: VE03 calls MB01 which has a restriction of 100 variables which in turn imposes a restriction of 50 variables on VE03. Furthermore VE03 uses named common for work space and this storage is restricted in that $n \leq 50$ and $m \leq 250$. To increase these restrictions above these amounts (to NB and MB say), then MB01 must be recompiled with an upper limit of 2*NB variables, and the following named common statements included in the user's MAIN program (where MM=MAX0(7*NB,2*NB+MB))

```
COMMON/VE03B/ DL(MM)
COMMON/VE03C/ B(N)
COMMON/VE03D/ BL(N)
COMMON/VE03E/ BU(N)
COMMON/VE03F/ U(N)
COMMON/VE03G/ V(N)
COMMON/VE03H/ LB(N)
```

If only the M limit is to be increased, then of course MB01 need not be recompiled, and only the first of the above named common statements need be included.

4 METHOD

The method is that described by R. Fletcher 'An efficient, globally convergent, algorithm for unconstrained and linearly constrained optimization problems', see report TP 431. Some modifications have been made to the method, which have been incorporated into this subroutine, and are described briefly in Section 2.6. The method is iterative and requires an initial feasible point **x** to be calculated (i.e. one which satisfies all the bounds and linear constraints).

Modifications to TP 431

In TP 431 it was suggested that a multiplier θ be introduced into the formula by which Γ (i.e. GM) was updated, in order to maintain positive definiteness. It was found on writing a Fortran subroutine to do this that this process could give rise to very slow convergence. Hence the multiplier θ has not to be used with the result that Γ can be a general indefinite matrix. A corresponding difficulty is caused in that the global minimum of the quadratic program should be found. However for the convergence proofs to be valid it is only necessary that a local minimum of the quadratic program be found which is sufficiently small in a certain sense. The measure of smallness involves solving a linear program over the same constraint set at each iteration. Modifications to make the program operate in this way have been introduced, so the performance figures given in TP 431 are still valid. Although the modifications have generated considerable extra code, there is good reason to think that any extra computing time required will be negligible.

A further modification has also been made to the ideas which enable the quadratic program to be solved more efficiently. The estimated number of operations per iteration is now $2\times n^3$ as against $2.5\times n^3$ with the modification suggested in TP 431 and $4\times n^3$ when using direct solution.