## 1 SUMMARY

To minimize a function, $f(x_1, x_2, ..., x_n)$, of n independent variables, $x_1, x_2, ..., x_n$, subject to m linear inequality constraints

$$\sum_{j=1}^{n} c_{ij}\, x_j \geq d_i \quad \text{i = 1, 2, ..., m}$$

See section 8 for an example.

The user must supply a subroutine to calculate values of both the function $f$ and the gradient vector $g_j = \frac{\partial f}{\partial x_j}$ $j$=1,2,...,$n$.

The method used requires an initial approximation to the minimum position to be provided. The accuracy required and a lower bound for $f$ must also be specified.

Notes are given in section 6 on the use of the routine in parametric programming problems where in such cases it is required to vary the definition of $f$ or vary the constraint constants.

If the function is quadratic in the variables then use of this routine is inefficient. The user is referred to the routine `VE02A`.

**ATTRIBUTES — Version:** 1.0.0. **Types:** `VE01A`, `VE01AD` **Calls:** `FD05`, `SDOT/DDOT` and `FUNCT` (a user subroutine). **Original date:** December 1969. **Origin:** R. Fletcher, Harwell.

## 2 HOW TO USE THE PACKAGE

### 2.1 The argument list and calling sequence

*The single precision version*

```
        SUBROUTINE VE01A(N,X,F,G,H,U,EP,FB,M,K,C,IC,
                         D,LT,CP,ICP,AL,R,MXF,IPR,IEXIT)
```

*The double precision version*

```
        SUBROUTINE VE01AD(N,X,F,G,H,U,EP,FB,M,K,C,IC,
                          D,LT,CP,ICP,AL,R,MXF,IPR,IEXIT)
```

Arguments set by the user

`N` is an `INTEGER` and should be sent to n the number of variables.

`X` is a `REAL` array: the first n members should be sent to an initial approximation to the minimum position $x_j$ j = 1, 2, ..., n. The initial approximation given must be feasible (i.e. satisfy the constraints) if it does not the following action is taken: the violated constraints are listed, and computation proceeds on the assumption that the violations are negligible. (This can happen from entries of the type described in section 6). If the violations are not negligible then the results will be meaningless. A routine `LA02A` is available to enable feasible points to be calculated for this problem.

`U` is a `LOGICAL` variable: set `U=.TRUE.` initially.

`EP` is a `REAL` array: the first n members should be set to the absolute accuracy required in the corresponding members of the solution vector $x_j$ j = 1, 2, ..., n.

`FB` is a `REAL` variable which should be set to a lower bound on the value of the function *f*. If a value of *f* less than `FB` is detected the routine returns to the caller with `IEXIT=5.`

`M` is an `INTEGER` and should be set to m, the number of constraints.

`K` is an `INTEGER` used by the routine to indicate the number of active constraints at the minimum. It should be set to zero initially.

`C` is a `REAL` two dimensional array which should contain the coefficients of the constraint inequalities, i.e. $c_{ij}$ i = 1, 2, ..., m, j = 1, 2, ..., n.

`IC` is an `INTEGER` and should be set to the first dimension of the array `C` (e.g. suppose space is allocated for `C` by `DIMENSION C(10,4),` then `IC` would be set to 10).

`D` is a `REAL` array and the first m members should be set to the constraint right hand sides $d_i$ i = 1, 2, ..., m.

`MXF` is an `INTEGER` and should be set to an upper limit on the number of evaluations of the function *f*. If the routine finds it requires more than `MXF` evaluations it will return to the caller with `IEXIT` = 6 and `X` set to the best approximation to the minimum obtained so far.

`IPR` is an `INTEGER` used to control the printed output. Printing will occur every `IPR` iterations except when `IPR` = 0 when no iteration details will be printed.

`ICP` is an `INTEGER` giving the first dimension of `CP.`

Arguments set by the routine

`X` the first n members of `X` will be set to the best approximation to the minimum position.

`F` is a `REAL` variable and will be set to the value of the function *f* at the minimum position given in `X.`

`G` is a `REAL` array of length at least n. The first n members will be set to the vector value of the gradient $g_j = \frac{\partial f}{\partial x_j}$ j = 1, 2, ..., n at the minimum position given in `X.`

**NOTE:** `F` and `G` are in fact set by the user supplied routine `FUNCT.`

`K` will give the number of active constraints (i.e. those constraints for which equality holds at the minimum position given in `X).`

`LT` is an `INTEGER` array of lengths at least 2m+n. The first `K` elements will contain a list giving the row numbers of the active constraints. The remainder of the array is used by the routine as workspace.

`AL` is a `REAL` array of length at least n. The first `K` members will contain the Lagrange multipliers of the active constraints in the order given by the array `LT.` These may be used in checking the solution (see section 5).

`R` is a `REAL` array. The first m members will contain the constraint residuals at the minimum, i.e.

$$r_i = \sum_{j=1}^{n} c_{ij} x_j - d_i \quad i = 1,2,...,m.$$

`IEXIT` is an `INTEGER.` On exit from the routine `IEXIT` is set to an integer value in the range 1 to 6 indicating the reason for the return to the caller (see section 5 for details).

Other arguments which must be provided

`H` is a `REAL` array of length at least $\frac{1}{2}$n(n+11).

`CP` is a `REAL` two dimensional array which should be at least n by n.

## 3 USER SUPPLIED SUBROUTINE

The user must provide a subroutine

```
      SUBROUTINE FUNCT(N,X,F,G)
      DIMENSION X(1),G(1)
           ......
```

statements to evaluate $f(x_1,x_2,...,x_n)$ and $g_j = \dfrac{\partial f}{\partial x_j}$ j = 1, 2, ..., n and stored in `F` and `G(J)` `J` = 1,`N,` given $x_j$ in `X(J)`

```
            ......
       RETURN
       END
```

If the programming of `F` and `G` is at all complicated the user is strongly advised to check out this routine before using it with `VE01A.` It is worth noting that constraints such as $x_j \geq 0$ may not, due to round off errors, preclude negative values of $x_j$. Hence constructions such as `SQRT(X(J))` should be used with care.

## 4 GENERAL INFORMATION

**Use of common:** none.

**Workspace:** none.

**Input/output:** on each `IPR` iterations a block of printing gives:

No. of iterations, No. of calls to `FUNCT,K,F`

`X(1), X(2),` ..., `X(N),`

`G(1), G(2),` ..., `G(N)`

`LT(1), AL(1),` ..., `LT(K), AL(K).`

On exit additional printing gives the value of `IEXIT,` and the contents of two vectors which are useful for verifying the solution (see section 5). Diagnostic printing may occur for the following conditions:

(i) Degeneracy: In certain cases degenerate constraints may cause trouble. If so the value of `D(I)` for such constraints is reduced by a small amount to enable the iteration to continue.

(ii) Nonfeasible starting point: Constraints violated by the initial approximation passed to the routine in `X` are listed.

## 5 VERIFICATION OF THE SOLUTION

The reason for the exit from `VE01A` is given in `IEXIT` in accordance with the following codes:–

(1) normal exit: either $|x_j^* - x_j| < \varepsilon_j$ j = 1, 2, ..., n (where $x_j^*$ j = 1, 2, ..., n is the next predicted minimum position and $\varepsilon_j$ is the accuracy requirement given in `EP(J)),` or $f(x_1^*, x_2^*, ..., x_n^*) = f(x_1, x_2, ..., x_n)$. No exit is taken if the list of active constraints changes.

(2) $\sum_{j=1}^{n} (x_j^* - x_j) \dfrac{\partial f}{\partial x_j} \geq 0$: not possible without rounding or programming error; probable causes are: accumulation of round-off error, or elements of `EP` set too small, or `G` programmed incorrectly.

(3) Irregular behaviour of `F` and `G: probable` causes as for 2.

(4) Breakdown of interpolation: probable causes as for 2.

(5) A function value less than the lower bound given in `FB` detected.

(6) FUNCT called MXF times.

The code returned in IEXIT can very occasionally be misleading. The convergence criterion used may result in a return with IEXIT=1 when the minimum in fact has not yet been found to the accuracy specified. On the other hand too small a value in the elements of EP may result in a return with IEXIT=2 or 3 although the solution has been obtained to the best possible accuracy.

It is advisable then to check the solution to allow this to be done easily additional information is printed out on exit. This follows the printing of LT(1), AL(1), ...,LT(K), AL(K) and comprises of the n elements of a vector $\eta$ followed by the m elements of the constraint residual array R.

The vector $\eta = \bar{C}\alpha$, where $\alpha$ is the vector of Lagrange multipliers (given in AL) and $\bar{C}$ is an n by K matrix consisting of the transpose of the K rows of C corresponding to the active constraints identified in LT. The Lagrange multipliers satisfy the relation

$$\bar{C}\alpha = g \qquad (g_j = \frac{\partial f}{\partial x_j} \text{ given in } G(J)).$$

This enables the following conditions, necessary for a solution, to be checked:

| | | | | | |
|---|---|---|---|---|---|
| (i) | $\alpha_j$ | $\geq$ | 0 | j = 1, 2, ..., K | ($\alpha_j$ given in AL(J)) |
| (ii) | $\eta_j$ | = | $\partial \frac{f}{\partial} x_j$ | j = 1, 2, ..., n | ($\partial \frac{f}{\partial} x_j$ given in G(J)) |
| (iii) | R(I) | = | 0 | if Ith constraint active | |
| | | $\geq$ | 0 | if Ith constraint not active | |

The extent of the agreement in the equalities will give an indication of how accurately the solution has been obtained.

One remote cause of failure is that the $H$ matrix (see section 6) goes singular ($Hy$=0 for $y$ such that $\bar{C}^T y$=0). If suspected restart from the current approximation with U=.TRUE. and K=0.

## 6 MORE SOPHISTICATED USE OF VE01A, PARAMETRIC PROGRAMMING

(i)**Initial H matrix** The array H is used to store an approximation to the inverse hessian matrix of $f$ as rows of its upper triangle in H(5*N+1) to H(N*(N+11)/2). If such an initial approximation is known, and is positive definite, then it should be supplied to VE01A with U set to .FALSE. (if U is .TRUE. as is normally the case then VE01A sets H to the unit matrix).

(ii)**Function parametric programming** If small changes are made to the parameters which define FUNCT, from a problem which has already been solved, the information from the previous solution can be used to greatly improve the time taken to solve the adjusted problem. This information is given in X, H, K, and CP (the generalised inverse matrix $\bar{C}^+$), which should be passed on unamended. U should be set to .FALSE. on re-entry to VE01A.

(iii)**Right hand side parametric programming** If small changes are made to D then a new $x$ say $x$* can be calculated which satisfies the new constraints. $x$* is given by

$$x* = x - [\bar{C}^+]^T r*$$

where r* are the new residuals of the previously active constraints. Statements similar to the following are required:–

```
      DO 1 I=1,K
      Z=D(LT(I))
      DO 2 J=1,N
2     Z=Z+C(LT(I),J)*X(J)
1     RSTAR(I)=Z
      DO 3 I=1,N
```

```
          Z=X(I)
          DO 4 J=1,K
    4     Z=Z-CP(J,I)*RSTAR(J)
    3     X(I)=Z
```

If the new X thus obtained does not violate any previously non-active constraints, then continue as in (ii). Otherwise proceed as in (iv).

(iv)**Constraint parametric programming** If small changes are made to C, or if the point determined in (iii) is no longer feasible, then a new feasible point must be supplied. However, information about the inverse hessian of *f* retained in H can be supplied to VE01A, but first the matrix $CC^+$ must be added to H to replace singular behaviour by unit matrix like behaviour in the space of the constraint normals. Statements as follows are required:–

```
          KK=5*N+1
          DO 1 I=1,N
          DO 4 J=I,N
          Z=0.
          DO 2 JJ=1,K
    2     Z=Z+C(LT(JJ),I)*CP(JJ,J)
          H(KK)=H(KK)+Z
    4     KK=KK+1
    1     CONTINUE
```

Then VE01A can be entered with U =.FALSE. and K=0.

## 7 METHOD

A generalization of Davidon's method, based on using an approximation *H* to the inverse hessian matrix of *f*, but enabling linear inequality constraints to be dealt with by projection techniques.

## 8 EXAMPLE

Minimize the function $f(x_1,x_2,x_3)=x_1\exp(x_2)+(x_3-x_2)^4$ subject to the conditions $x_1+x_2+x_3\geq 0$, $x_1\geq 1$. The partial derivatives $\partial f\backslash\partial x_j=g_j$ $j$=1,3 will be required. These are –

$g_1=\exp(x_2)$; $g_2=x_1\exp(x_2)-4(x_3-x_{2)}^3$; $g_3=4(x_3-x_2)^3$.

The subroutine FUNCT that is required to evaluate *f* and $g_j$ $j$=1,3 would be as follows –

```
      SUBROUTINE FUNCT(N,X,F,G)
      DIMENSION X(N),G(N)
      G(1)=EXP(X(2))
      Q=(X(3)-X(2))**3
      F=X(1)*G(1)+(X(3)-X(2))*Q
      G(2)=X(1)*G(1)-4.*Q
      G(3)=4.*Q
      RETURN
      END
```

The space allocation for the arrays could be –

```
DIMENSION C(2,3),D(2),X(3),EP(3),G(3),LT(9),
          AL(3),R(3),H(21),CP(3,3)
```

The input parameters to `VE01A` could then be –

```
N=3; X(J)=1. J=1,3; U=.TRUE.; EP(J)=.001 J=1,3; FB=0.; M=2; K=0;
C(1,J)=1.,1.,1.
C(2,J)=1.,0.,0. J=1,3; IC=3; D(I)=0.,0.1 I=1,2; MXF=50; IPR=1;
ICP=3
```

where we have taken the initial values for *x* to be 1.,1.,1. and we are asking for an absolute accuracy of .001 in each of the parameters *x*. The function *f*(*x*) cannot be negative so we set `FB` to 0.

The final print out from `VE01A` was:–

```
  EXIT FROM VE01A, IEXIT=2
   5  14  2  0.875542E-01
 0.999969E-01 -0.159992E 00 0.599926E-01 0.852150E 00 0.426289E-01 0.425834E-01
   2  0.809544E 00  1  0.426062E-01
 0.852150E 00  0.426062E-01  0.426062E-01
-0.299886E-05 -0.309944E-05
```

which indicates that there were

5 iterations

14 calls to FUNCT

2 constraints active

and the function value is `0.875542E-01`

The second line gives the 3*x* values followed by the 3 partial derivatives. The third line gives the active constraints, i.e. constraint No. followed by the Lagrange multiplier for the constraint.

A test for the accuracy of the result is to compare the 3 numbers in the fourth line ($\eta$) with the last 3 numbers in the second line (the partial derivatives). In this example the comparison is good showing that a good result has been reached despite that `IEXIT=2` indicates that a certain amount of round-off error had occurred.

The last line gives the constraint residuals.