1 SUMMARY

This subroutine seeks a vector $\mathbf{x}=(x_1,x_2,...,x_n)$ that **minimizes the quadratic function**

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} = \tfrac{1}{2}\sum_{i,j} a_{ij}\, x_i\, x_j - \sum_i b_i\, x_i$$

subject to the m linear constraints

$$x_i \geq l_i,\ i=1,2,...,n \tag{1}$$

$$x_i \leq u_i,\ i=1,2,...,n \tag{2}$$

$$\mathbf{c}_j^T\mathbf{x} \geq d_j,\ j=1,2,...,m-2n \tag{3}$$

any of which may be required to be satisfied as equalities.

The algorithm works by first finding a feasible point (a point satisfying all the constraints) and then solving a sequence of problems in which $f(\mathbf{x})$ is minimized subject to a varying set of constraints being active (satisfied as equalities). There are options for starting with n active constraints (sensible for the general case) or with none (sensible for the convex case, for which few constraints may be active at the solution). There are also facilities for parametric programming (solving sequences of slightly different problems).

The user must specify which constraints are equalities. These must be linearly independent and are satisfied at every point.

Note that a very large negative l_i or a very large positive u_i may cause roundoff problems. Where a variable x_i has no lower bound or no upper bound, the values chosen for l_i and u_i should be reasonably comparable (e.g. ten times bigger) than the eventual size of x_i.

If the matrix **A** is positive semi-definite, a global solution is found. However, if **A** is indefinite, the procedure may find a local solution which is not the global solution to the problem.

ATTRIBUTES — Version: 1.0.0. **Types:** `VE02A; VE02AD.` **Calls:** `FD05`, `LA02` and `MB01`. **Original date:** July 1970. **Origin:** R.Fletcher, ex-Harwell.

2 HOW TO USE THE PACKAGE

2.1 Argument List

The single precision version

```
CALL VE02A(N,M,A,IA,B,C,IC,D,BDL,BDU,X,K,KE,H,IH,LT,MODE)
```

The double precision version

```
CALL VE02AD(N,M,A,IA,B,C,IC,D,BDL,BDU,X,K,KE,H,IH,LT,MODE)
```

`N` is an `INTEGER` variable which the user must set to the number of variables. It is not altered by the subroutine. **Restriction:** $n>0$.

`M` is an `INTEGER` variable which the user must set to the total number of constraints. It is not altered by the subroutine. **Restriction:** $m \geq 2n$.

`A` is a `REAL` (`DOUBLE PRECISION` in the D version) two-dimensional array which the user must set to hold the coefficients a_{ij} of the symmetric matrix **A**. It is not altered by the subroutine. **Restriction:** $a_{ij}=a_{ji}$, $i,j=1,2,...,n$ (symmetry).

IA is an INTEGER variable which the user must set to the first dimension of A in the DIMENSION statement which allocates space to A. It is not altered by the subroutine. **Restriction:** IA $\geq n$.

B is a REAL (DOUBLE PRECISION in the D version) array of length n which the user must set to hold the coefficients b_i of the vector **b**. It is not altered by the subroutine.

C is a two-dimensional REAL (DOUBLE PRECISION in the D version) array which the user must set so that C(i,j), i=1,2,...n hold the components of the constraint vector $\mathbf{c}_j$, for j=1,...,m–2n. It is not altered by the subroutine.

IC is an INTEGER variable which the user must set to the first dimension of C in the DIMENSION statement which allocates space to C. It is not altered by the subroutine. **Restriction:** IC $\geq n$.

D is a REAL (DOUBLE PRECISION in the D version) array, of length m–2n, which the user must set to hold d_j, j=1,2,...,m–2n. It is not altered by the subroutine.

BDL is a REAL (DOUBLE PRECISION in the D version) array of length n, which the user must set to hold l_i, i=1,2,...,n.

BDU is a REAL (DOUBLE PRECISION in the D version) array of length n, which the user must set to hold u_i, i=1,2,...,n.

X is a REAL (DOUBLE PRECISION in the D version) array of length max(2n+m,7n). The user must set the first n components to an initial estimate of the solution and on a successful return from the subroutine X(i), i=1,2,...,n hold the components of the solution found, X(n+i), i=1,2,...,k hold the Lagrange multipliers (rates of change of the minimum of f(**x**) with respect to changes in the right-hand sides of the active constraints) and X(n+k+1) holds the value of the function f. The rest of X is used as workspace.

K is an INTEGER variable which the user must set to the number k of constraints he or she wishes to be active (satisfied as equalities) initially. On successful return it contains the number of active constraints at the solution. **Restriction:** KE $\leq$ K $\leq n$.

KE is an INTEGER variable which the user must set to the number of constraints that are equalities. **Restriction:** 0 $\leq$ KE $\leq k$.

H is a REAL (DOUBLE PRECISION in the D version) work-array of dimensions (IH,2n).

IH is an INTEGER variable which the user must set to the first dimension of H in the DIMENSION statement that allocates space to it. It is not altered by the subroutine. **Restriction:** IH $\geq$ 2n.

LT is an INTEGER array of length 2n+m. The user must set LT(j), $j \leq$ KE, to indicate which constraints are equalities and LT(j), KE $< j \leq$ K to indicate which inequality constraints he or she wishes to be satisfied initially as equalities. Values LT(j) are interpreted as follows:

constraint (1) is active for x_i (with i = LT(j)) if $1 \leq$ LT(j) $\leq n$,

constraint (2) is active for x_i (with i = LT(j)–n) if n+1 $\leq$ LT(j) $\leq$ 2n,

constraint (3) is active for $\mathbf{c}_i$, d_i (with i = LT(j)–2n) if 2n+1 $\leq$ LT(j) $\leq m$.

On output LT(j), j=1,2,...,k contain the index numbers of the active constraints, to be interpreted in the same way.

The rest of LT is used for workspace.

MODE is an INTEGER variable, which the user must set to one of the following values according to his or her requirements. It is not changed by the subroutine.

1. This is for solving a problem with no special features. It starts by calling LA02A/AD to give a feasible point with n constraints active (satisfied as equalities). The initial values of K and X(i), i=1,2,...,n are passed to LA02A/AD.

2 or 3. This is for solving a convex problem (**A** positive-definite, i.e. $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ unless **x**=**0**) for which **A** is well-conditioned ($\mathbf{A}^{-1}$ of moderate size) and which has no constraints specified as equalities (i.e. KE must be zero), by commencing with no active constraints. If MODE=2, an initial feasible point is found

by calling LA02A/AD as for mode 1. If MODE=3 then X(i), i=1,2,...,n is used as starting point unless it is not feasible, in which case LA02A/AD is called as for MODE=2.

4. This is for solving a problem that differs only slightly from one already solved (parametric programming). Arguments N, M, K, KE, LT must be unchanged since a previous successful call.
5. This is for solving a problem that differs only in the values of **b**, **l**, **u**, **d** and inactive constraint normals $\mathbf{c}_j$ (LT($4n+j$)=1 on return from VE02A/AD). Arguments N, M, A, IA, IC, K, KE, H, IH, LT must be unchanged since a previous successful call.

2.2 The common area and diagnostic messages

The single-precision version:

```
COMMON/VE02B/LP,IFLAG
```

The double-precision version:

```
COMMON/VE02BD/LP,IFLAG
```

LP is an INTEGER variable with default value 6, set by BLOCK DATA, that specifies the stream number for diagnostic messages. If the user resets LP ≤ 0, then no diagnostic messages are printed.

IFLAG is an INTEGER variable which on return may have one of the following values:

0 successful return

1 the problem has no solution

2 the given equality constraints are linearly dependent

3 disastrous loss of accuracy, probably caused by a non-convex problem on a MODE 2 or 3 entry

4 warning that the solution may be a degenerate local minimum (i.e. small changes to problem might lead to a large improvement in f)

5 equality constraints present on a MODE 2 or 3 entry.

3 GENERAL INFORMATION

Use of common: see section 2.2.

Workspace: see description of arguments X, H and LT.

Other routines called directly: FD05, LA02A/AD and MB01C/CD are called.

Input/output: Error messages are printed under the control of common variable LP (see section 2.2).

Restrictions:

$n > 0$

$m \geq n$

IA $\geq n$

IC $\geq n$

$0 \leq$ KE $\leq$ K $\leq n$

IH $\geq 2n$

$1 \leq$ LT(i) $\leq m$, i=1,2,...,K.

4 METHOD

The method has already been sketched in section 1. It is described in more detail by R. Fletcher in 'A General Quadratic Programming Algorithm', AERE-TP.401 and in 'A Fortran Subroutine for General Quadratic Programming', AERE-R.6370. It finds feasible points by calling LA02A/AD, which is described in detail in R. Fletcher's 'The calculation of feasible points for linearly constrained optimization problems', AERE-R.6354.

The method is so designed that the computation of ill-conditioned operators corresponding to an intermediate basis in the iteration is avoided. The main source of errors is thus from the successive updating of the operators which enable each intermediate solution to be determined. To negate this accumulation, the final solution is recalculated by inverting ab-initio the system of linear equations which determine the solution. This solution is then checked as for MODE 4. This reinversion is not carried out when the parametric programming modes (4 and 5) are being used, because under these circumstances it is expected that the solution will not have been obtained by successive updating of the operators.

Execution times vary with the number of iterations required, but are at least comparable with the time required to solve a set of linear equations in *n* variables and might typically take 10 times this amount. Savings can be obtained by using parametric programming when appropriate, especially in MODE 5 when the amount of computation should reduce by a factor of *n*.