1 SUMMARY

This subroutine finds a local minimizer of the differentiable function $F(x)$, of the n variables $x_1, ..., x_n$, subject to general linear constraints

(1) $$a_j^T x = b_j \quad j = 1, 2, ..., m_e$$
$$a_j^T x \le b_j \quad j = m_e + 1, ..., m$$

and simple bounds

(2) $$l_i \le x_i \le u_i \quad i = 1, 2, ..., n,$$

where the constraint coefficients a_{ij}, the right hand sides b_j and the bounds l_i and u_i are given. The user must provide a subroutine that calculates $F(x)$ and its gradient $\nabla F(x)$ for any feasible x. The method is described in the report " A tolerant algorithm for linearly constrained optimization calculations ", by M.J.D.Powell (Report DAMTP/1988/NA17, University of Cambridge. To be published in Mathematical Programming B, 1989).

ATTRIBUTES — Version: 1.0.0. **Types:** VE11A,VE11AD **Original date:** August 1989. **Origin:** M.J.D.Powell, D.A.M.T.P., University of Cambridge, **Licence:** A third-party licence for this package is available without charge.

2 HOW TO USE THE PACKAGE

2.1 Method of Use

The user is required to calculate the objective function $F(x)$ and its gradient $\nabla F(x)$

2.2 The Argument List

The single precision version

```
CALL VE11A(N,M,MEQ,A,IA,B,XL,XU,X,ACC,IACT,NACT,PAR,IPRINT,
           INFO,W,FGCALC)
```

The double precision version

```
CALL VE11AD(N,M,MEQ,A,IA,B,XL,XU,X,ACC,IACT,NACT,PAR,IPRINT,
            INFO,W,FGCALC)
```

N is an INTEGER variable that must be set by the user to n, the number of variables of the objective function. Restrictions:N > 0.

M is an INTEGER variable that must be set by the user to m, the number of general linear constraints (excluding simple bounds). Restriction: M ≥ 0.

MEQ is an INTEGER variable that must be set by the user to m_e, the number of general linear constraints that are equalities. Restriction: MEQ ≥ 0.

A is a two-dimensional REAL (DOUBLE PRECISION in the D version) array of dimensions (IA,M) in which the elements of the constraint gradients are stored. The user must set A(I, J) to a_{ij} in the expression (1) above.

IA is an INTEGER variable that must be set by the user to the first dimension of the array A. Restriction: IA ≥ N

B is a REAL (DOUBLE PRECISION in the D version) array in which the vector of right hand sides is stored. The user must set `B(J)` to b_j for $1 \le j \le m$.

`XL` and `XU` are `REAL` (`DOUBLE PRECISION` in the D version) arrays whose first `N` elements must be set to the bounds l_i and u_i, $i = 1, 2, ..., n$ respectively. If a variable is to be unconstrained, the user should assign very large positive and negative values in `XL` and `XU.` The i–th variable can be frozen by setting `XL(`i`)` to `XU(`i`)`.

`X` is a `REAL` (`DOUBLE PRECISION` in the D version) array. Initially its first `N` elements must provided starting values for the algorithm, and on exit it will have been automatically set to the vector that gives the least calculated value of the objective function, except that no values of $F(x)$ are calculated if the feasible region is empty. There is no need for the initial `X` to satisfy the linear constraints, but it is helpful to choose it as close as possible to the required solution.

`ACC` is a `REAL` (`DOUBLE PRECISION` in the D version) variable that must be set by the user to indicate the accuracy that is required from the calculation. Roughly speaking, the calculation ends when the 2-norm of the Kuhn-Tucker residual vector is at most `ACC.` In particular, in the unconstrained case, the condition for termination is the inequality $\| \nabla F(x) \|_2 \le$ `ACC.` The choice `ACC=0` is often convenient, because there are safeguards that give automatic termination when the required accuracy cannot be achieved.

`IACT` is an `INTEGER` array of length at least `M+2N.` Although it is used for working space during the calculation, `IACT` is included in the argument list in order to provide the indices of the active constraints at termination. Specifically, the indices of the final active constraints are `IACT(`k`),` $k = 1, 2,...,$ `NACT,` (see below). The value of `IACT(k)` is an integer from the interval $(1, m+2n)$. An integer in $(1, n)$ corresponds to an active general linear constraint. Any integers in $(m+1, m+n)$ correspond to active lower bounds on the components of x. Similarly, an integer in $(m+n+1, m+2n)$ corresponds to an active upper bound. For example, if `M=5, N=3, NACT` $\ge$ `2, IACT(1)=8` and `IACT(2)=9,` then the bounds $l_3 \le x_3$ and $x_1 \le u_1$ are in the final active set.

`NACT` is an `INTEGER` variable that is set by `VE11A/AD` to the final number of active constraints. As the gradients of the active constraints are always linearly independent, `NACT` must lie in the range $(0, n)$,

`PAR` is a `REAL` (`DOUBLE PRECISION` in the D version) array whose first `NACT` components, are set by `VE11A/AD` to estimates of the Lagrange multipliers of the active constraints at the solution of the optimization calculation.

`IPRINT` is an `INTEGER` variable whose value must be set by the user. It specifies the frequency and amount of printing during the execution of the optimization package. There is no printed output if `IPRINT=0.` Otherwise the value of $F(x)$ and the components of x and $\nabla F(x)$ are output for the first feasible point, whenever the tolerance parameter `TOL` is reduced, for the final vector of variables, and after every `|IPRINT|` intermediate iterations. If `IPRINT` $\le$ `0,` the current values of `IACT(`k`),` $k = 1, 2, ...,$ `NACT` , `PAR(`k`),` $k = 1, 2, ...,$ `NACT` and the residuals of the Kuhn-Tucker optimality equations, `RESKT(`i`)` $i = 1, 2, ...$ `N,` are also printed The reason for termination is also displayed when `IPRINT` is nonzero.

`INFO` is an `INTEGER` variable that has two purposes. If the users program sets it to a positive value initially then it is an upper bound on the number of calls to the subroutine that calculates $F(x)$ and $\nabla F(x)$ for any feasible x. Moreover, in all cases it is set by `VE11A/AD` to a number from the interval $(1, 8)$ to indicate the reason for return from the subroutine. For meanings of the different values see § 2.5.

`W` is a `REAL` workspace array (`DOUBLE PRECISION` in the D version) of length at least `M+11*N+N*N.` On exit the final components of $\nabla F(x)$ and `RESKT` can be found in `W(1),...,W(N)` and `W(N+1),...W(2 * N)` respectively.

`FGCALC` is a name of a subroutine supplied by the user to calculate the objective function and its gradient. Its calling sequence is as follows

```
        SUBROUTINE FGCALC(N,X,F,G)
```

where

`N` is an `INTEGER` variable that is the number of variables of the objective function. It must not be altered by the subroutine.

X is a REAL (DOUBLE PRECISION in the D version) array containing values of x for which the objective function $F(x)$ and gradient vector $\nabla F(x)$ are required. It must not be altered by the subroutine.

F is a REAL (DOUBLE PRECISION in the D version) variable which must be set by the subroutine to the value of the objective function $F(x)$.

G is a REAL (DOUBLE PRECISION in the D version) array which must be set to the components of the gradient vector $\nabla F(x)$ user.

2.3 Block data

The program that calls VE11A/AD must contain the declaration:

The single precision version

```
EXTERNAL VE11T
```

The double precision version

```
EXTERNAL VE11TD
```

VE11TD is a BLOCK DATA subprogram that defines the default values of several parameters.

2.4 Common

The subroutine contains one common area, which the user may wish to reference:

The single precision version

```
COMMON/VE11U/ LP,MP
```

The double precision version

```
COMMON/VE11UD/ LP,MP
```

LP is an INTEGER variable with default value 6. It is used as the unit number for the printing of error messages. This printing may be suppressed by setting LP to a non-positive value.

MP is an INTEGER variable with default value 6. It is used as the unit number for the printing of output other than error messages (see IPRINT option). This printing may be suppressed by setting LP to a non-positive value.

2.5 Messages

On exit from VE11A/AD, INFO will have one of the following values.

INFO =1 The final X is feasible and termination condition that depends on ACC is satisfied.

INFO =2 The final X is feasible and termination occurs because rounding errors seem to be preventing higher accuracy.

INFO =3 The final X is feasible and a line search fails to reduce the objective function, although a reduction is predicted by the current gradient vector. This return may also be due to rounding errors, but if the components of RESKT are large, it is likely that the users subroutine for calculating $\nabla F(x)$ is incorrect. One should also question the coding of the gradient when the final rate of convergence is slow.

INFO =4 In this case the calculation cannot begin, because there is a violation of at least one of the conditions $1 \le N$, $0 \le M$ and $0 \le MEQ \le M$, or because a variable has a lower bound that is greater than its upper bound.

INFO =5 This error return indicates that the equality constraints are inconsistent. These constraints include the freezing of any variables due to XL(i)=XU(i).

INFO =6 This error return indicates that the equality constraints and the bounds on the variables are incompatible.

INFO =7 In this case it is possible to satisfy all the equality constraints and bounds, but the general inequality constraints in the middle line of expression 1.1 cannot be satisfied too. When this return or an INFO=6 return occurs, the final values of the integers `IACT(`k`)`, $k=1,2,...,$ `NACT` are significant, because just the constraints with these indices prevent a change to the current variables `X(`i`)`, $i=1,2,...,$ `N` that reduces the sum of moduli of constraints violations,

`INFO=8` The limit on the number of calls of the subroutine that calculates $F(x)$ and $\nabla F(x)$ has been reached, and there would have been further calculation otherwise.

3 GENERAL INFORMATION

Use of common: uses common area `VE11U/UD` (see section 2.4).

Workspace: see argument `W`.

Other routines called directly: `VE11B/BD`, `VE11C/CD`, `VE11D/DD`, `VE11E/ED`, `VE11F/FD`, `VE11G/GD`, `VE11H/HD`, `VE11I/ID`, `VE11J/JD`, `VE11K/KD`, `VE11L/LD`, `VE11M/MD`, `VE11N/ND`, `VE11O/OD`, `VE11P/PD`, `VE11Q/QD`, `VE11R/RD`, `VE11S/SD`.

Input/output: No input; any output on unit `LP`.

4 METHOD

The underlying algorithm is iterative, each iteration making a line search in the space of variables, first to achieve feasibility and then to minimise the objective function. The most distinctive feature of the method is that each search direction is calculated so that it does not intersect the boundary of any inequality constraint that is satisfied and that has a small residual at the beginning of the line search. Thus maintaining feasibility does not necessitate tiny steplengths when the constraints include some near degeneracies, which can help efficiency greatly. It is particularly suitable if there are any degenerate or nearly degenerate constraints, because no search direction is allowed to move towards the boundary of a constraint that has a small residual at the initial point of a line search. Otherwise it is a typical active set method, see Gill, Murray and Wright (Practical Optimization, Academic Press, 1981) that employs BFGS updating of second derivative approximations and the matrix factorizations of Golfarb and Idnani, (“ A numerically stable dual method for solving strictly convex quadratic programs,” Mathematical Programming, vol 27, pp 1-33).

The method is described in detail by Powell (“ TOLMIN: A fortran package for linearly constrained optimization calculations, Report DAMTP/1989/NA2, University of Cambridge ”).

5 EXAMPLE OF USE

We now consider the pentagon problem. This calculation is an approximation to the problem of finding the positions of three points inside or on a regular pentagon such that the least distance between any two points is a large as possible. There are six variables, namely the two coordinates of each of the three points, and the fifteen linear conditions on the variables that should prevent the points from going beyond the edges of the pentagon. Specifically, the points are (x_1,x_2), (x_3,x_4) and (x_5,x_6), the constraints are the inequalities

$$x_i \cos\frac{2\pi j}{5} + x_{i+1}\sin\frac{2\pi j}{5} \le 1\ , \qquad i=1,3,5,\ \ j=0,1,2,3,4.$$

and we use the objective function

$$F(x)=[(x_1-x_3)^2+(x_2-x_4)^2]^{-8}+[(x_3-x_5)^2+(x_4-x_6)^2)]^{-8}+[(x_5-x_1)^2+(x_6-x_2)^2)]^{-8}$$

because, if we take the view that the exponent -8 is a suitable approximation to the exponent $-\infty$, then this linearly constrained optimization calculation is analogous to the given problem.

The initial values of the variables are chosen to be $(x_1,x_2) = (-1.0, 0.0)$, $(x_3,x_4) = (0.0,-1.0)$ and $(x_5,x_6) = (1.0, 1.0)$.

```
C
C     The pentagon problem.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(10,15),B(15),XL(6),XU(6),X(6),IACT(27),PAR(20),
     1  W(1000)
      COMMON /VE11UD/LP
      EXTERNAL FGCAL1
C
C     Set the initial values of ACC and IPRINT.
C
      ACC=1.0E-3
      IPRINT=0
      IA=10
C
C     Set the components of XL, XU and X.
C
      N=6
   10 DO 20 I=1,N
      XL(I)=-1.0E6
      XU(I)=1.0E6
   20 X(I)=0.5*DFLOAT(I-3)
      X(2)=0.0
      X(4)=-1.0
      X(6)=1.0
C
C     Set the constraints.
C
      M=0
      MEQ=0
      PI=4.0*DATAN(1.0D0)
      DO 40 K=1,5
      THETA=0.4*DFLOAT(K-1)*PI
      CTH=DCOS(THETA)
      STH=DSIN(THETA)
      DO 40 J=2,N,2
      M=M+1
      DO 30 I=1,N
   30 A(I,M)=0.0
      A(J-1,M)=CTH
      A(J,M)=STH
   40 B(M)=1.0
C
C     Call the optimization package.
C
      INFO=0
      IF(LP.GT.0)WRITE(LP,50) ACC,IPRINT
   50 FORMAT (//5X,'CALL OF VE11AD WITH ACC =',E14.4,
     1  '  AND IPRINT =',I4)
      CALL VE11AD (N,M,MEQ,A,IA,B,XL,XU,X,ACC,IACT,NACT,PAR,IPRINT,
     1  INFO,W,FGCAL1)
      IF(LP.GT.0)WRITE(LP,60) INFO
   60 FORMAT (/5X,'RETURN FROM TOLMIN WITH INFO =',I4)
      IF(LP.GT.0)WRITE(LP,70) (X(I),I=1,N)
   70 FORMAT (/5X,'THE COMPUTED POINTS OF THE PENTAGON PROBLEM',
     1  ' HAVE THE COORDINATES'/(5X,2F12.6))
```

```
      IF(LP.GT.0)WRITE(LP,80)
   80 FORMAT (/5X,'AND THE CONSTRAINTS ARE AS FOLLOWS')
      DO 100 K=1,NACT
      IX=MOD(IACT(K)-1,3)+1
      IE=(IACT(K)+2)/3
      IF(LP.GT.0)WRITE(LP,90) IX,IE
   90 FORMAT (9X,'POINT',I2,' IS ON EDGE',I2)
  100 CONTINUE
      IF(LP.GT.0)WRITE(LP,300)
  300 FORMAT (/12X,'X(I)',8X,'X(I)-XL(I)',5X,'XU(I)-X(I)')
      DO 310 I=1,N
  310 IF(LP.GT.0)WRITE(LP,320) X(I),X(I)-XL(I),XU(I)-X(I)
  320 FORMAT (5X,1P,3E15.7)
      IF (M .GT. 0) THEN
          DO 330 K=1,M
          DO 330 I=1,N
  330     B(K)=B(K)-A(I,K)*X(I)
          IF(LP.GT.0)WRITE(LP,340) (B(K),K=1,M)
  340     FORMAT (/5X,'FINAL CONSTRAINT RESIDUALS =',/(1P,6E12.4))
      END IF
C
C     Repeat the calculation once with ACC=0.0.
C
      IF (ACC .GT. 0.0) THEN
          ACC=0.0
          IPRINT=0
          GOTO 10
      END IF
      STOP
      END
      SUBROUTINE FGCAL1 (N,X,F,G)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(*),G(*)
C
C     Calculate the objective function and its gradient.
C
      WA=(X(1)-X(3))**2+(X(2)-X(4))**2
      WB=(X(3)-X(5))**2+(X(4)-X(6))**2
      WC=(X(5)-X(1))**2+(X(6)-X(2))**2
      F=1.0/(WA**8)+1.0/(WB**8)+1.0/(WC**8)
      G(1)=16.0*((X(3)-X(1))/(WA**9)+(X(5)-X(1))/(WC**9))
      G(2)=16.0*((X(4)-X(2))/(WA**9)+(X(6)-X(2))/(WC**9))
      G(3)=16.0*((X(5)-X(3))/(WB**9)+(X(1)-X(3))/(WA**9))
      G(4)=16.0*((X(6)-X(4))/(WB**9)+(X(2)-X(4))/(WA**9))
      G(5)=16.0*((X(1)-X(5))/(WC**9)+(X(3)-X(5))/(WB**9))
      G(6)=16.0*((X(2)-X(6))/(WC**9)+(X(4)-X(6))/(WB**9))
      RETURN
      END
```

The output is as follows:

```
     CALL OF VE11AD WITH ACC =    0.1000E-02  AND IPRINT =   0

     RETURN FROM TOLMIN WITH INFO =   1

     THE COMPUTED POINTS OF THE PENTAGON PROBLEM HAVE THE COORDINATES
        -1.100931    0.186000
         0.320566   -0.947304
```

```
          0.932139    0.748592

      AND THE CONSTRAINTS ARE AS FOLLOWS
          POINT 3 IS ON EDGE 2
          POINT 1 IS ON EDGE 3
          POINT 2 IS ON EDGE 5

             X(I)        X(I)-XL(I)     XU(I)-X(I)
       -1.1009309E+00  9.9999890E+05  1.0000011E+06
        1.8600036E-01  1.0000002E+06  9.9999981E+05
        3.2056564E-01  1.0000003E+06  9.9999968E+05
       -9.4730414E-01  9.9999905E+05  1.0000009E+06
        9.3213867E-01  1.0000009E+06  9.9999907E+05
        7.4859196E-01  1.0000007E+06  9.9999925E+05

      FINAL CONSTRAINT RESIDUALS =
   2.1009E+00  6.7943E-01  6.7861E-02  1.1633E+00  1.8019E+00  2.7756E-17
   1.1102E-16  1.8162E+00  1.3141E+00  2.1866E-01  7.0253E-01  2.1941E+00
   1.5171E+00  0.0000E+00  1.4239E+00


      CALL OF VE11AD WITH ACC =    0.0000E+00  AND IPRINT =   0

      RETURN FROM TOLMIN WITH INFO =   3

      THE COMPUTED POINTS OF THE PENTAGON PROBLEM HAVE THE COORDINATES
         -0.889184    0.477445
          0.179305   -0.993203
          1.000000    0.726542

      AND THE CONSTRAINTS ARE AS FOLLOWS
          POINT 3 IS ON EDGE 2
          POINT 1 IS ON EDGE 3
          POINT 2 IS ON EDGE 5
          POINT 3 IS ON EDGE 1

             X(I)        X(I)-XL(I)     XU(I)-X(I)
       -8.8918383E-01  9.9999911E+05  1.0000009E+06
        4.7744510E-01  1.0000005E+06  9.9999952E+05
        1.7930474E-01  1.0000002E+06  9.9999982E+05
       -9.9320254E-01  9.9999901E+05  1.0000010E+06
        1.0000000E+00  1.0000010E+06  9.9999900E+05
        7.2654247E-01  1.0000007E+06  9.9999927E+05

      FINAL CONSTRAINT RESIDUALS =
   1.8892E+00  8.2070E-01  0.0000E+00  8.2070E-01  1.8892E+00  4.1633E-17
   2.7756E-17  1.7289E+00  1.3820E+00  5.6127E-01  5.6127E-01  2.2361E+00
   1.7288E+00  2.7756E-17  1.3820E+00
```