Science & Technology
Facilities Council

VA27

PACKAGE SPECIFICATION HSL ARCHIVE

1 SUMMARY

To find a local minimum of a sum of squares of *m* nonlinear functions of *n* variables, that is to minimize

$$\sum_{i=1}^{m} [r_i(x_1, x_2, ..., x_n)]^2$$

Typically the r_i might be the residuals of a nonlinear least squares data fitting problem, as in the example of section 8. The user must be able to calculate the functions r_i and the partial derivatives $\partial r_i/\partial x_j$ for all *i* and *j*: this information is presented to `VA27A` by two user subroutines as described in section 3. The method is described by R. Fletcher (1971), ‘A modified Marquardt subroutine for nonlinear least squares’, Harwell report, AERE-R 6799; it is iterative so that an initial approximation to $x_1, x_2, ..., x_n$ must be supplied. The method allows the imposition of constraints in a limited way as described in section 6. It may also be possible to improve the performance of the method by scaling the variables in a realistic way (see section 5). An automatic choice can be made by `VA27A` or this can be overridden by the user if desired.

ATTRIBUTES — Version: 1.0.0. **Types:** `VA27A`, `VA27AD`. **Calls:** `FD05`, `MA10`, `SDOT/DDOT`. **Original date:** February 1994. **Origin:** R. Fletcher, Harwell. **Remark:** A version of `VA07` with a modified argument list.

2 HOW TO USE THE PACKAGE

2.1 The argument list

The single precision version

```
CALL VA27A(RESID,LSQ,M,N,X,R,SS,A,D,EPS,IPRINT,MAXFN,MODE,W)
```

The double precision version

```
CALL VA27AD(RESID,LSQ,M,N,X,R,SS,A,D,EPS,IPRINT,MAXFN,MODE,W)
```

`RESID` identifier of user subroutine – see section 3.

`LSQ` identifier of user subroutine – see section 3.

`M` an `INTEGER` set to the number of functions *m*.

`N` an `INTEGER` set to the number of variables *n*.

`X` a `REAL` (`DOUBLE PRECISION` in the D version) array of *n* elements, set so that the `X(I)` are the initial approximations to x_i. The best approximation to the minimum which is found will overwrite `X` on exit from `VA27A`.

`R` a `REAL` (`DOUBLE PRECISION` in the D version) array of *m* elements, which is such that on exit from `VA27A` `R(I)` contains the value of the residual $r_i(x_1, x_2, ..., x_n)$ corresponding to the values returned in `X`. `R` need not be set by the user on entry to `VA27A`.

`SS` a `REAL` (`DOUBLE PRECISION` in the D version) variable which on exit from `VA27A` contains the sum of squares of the `values` in `R(I)`. `SS` need not be set by the user on entry to `VA27A`.

`A` a `REAL` (`DOUBLE PRECISION` in the D version) array of at least n^2 elements, used by `VA27A` as working space.

`D` a `REAL` (`DOUBLE PRECISION` in the D version) array of *n* elements, only to be set if `MODE` = 3, and which controls scaling (see section 5).

`EPS` a `REAL` (`DOUBLE PRECISION` in the D version) array of *n* elements, set so that `EPS(I)` is the absolute accuracy to which x_i should approximate the solution.

`IPRINT` an `INTEGER` which controls the frequency and amount of printing – see section 4.

`MAXFN` an `INTEGER` giving an upper limit to the number of times `RESID` is called – see section 3.

`MODE` an `INTEGER` which governs the method of scaling the variables –see section 5.

`W` a `REAL` (`DOUBLE PRECISION` in the D version) array of length at least $4n+m$ which is used by the routine as workspace.

3 USER SUBROUTINES

The user must provide two subroutines, one to calculate the residuals, and the other to calculate derivatives. The user may choose any identifier for the subroutines and must pass them as the first two arguments to `VA27A`. An `EXTERNAL` statement must also appear in the user's MAIN program. (See the example in section 8). These subroutines should be written as follows.

```
SUBROUTINE RESID (M,N,X,R,IFL)
DIMENSION X(1),R(1)
   -----
   -----
```

[statements to evaluate $r_i(x_1,x_2,...,x_n)$
for i=1,2,...,m and store them in `R(I), I=1,2,...,M`.]

```
   -----
   -----
RETURN
END
```

If for any reason, one or more of the r_i cannot be evaluated at $x_1,x_2,...,x_n$ (for example if overflow or negative square root would occur), then the `INTEGER` variable `IFL` should be set to 1 and a `RETURN` made. This feature can also be used to impose constraints on the variables in a limited way – see section 6.

```
SUBROUTINE LSQ(M,N,X,R,A,V)
DIMENSION X(1),R(1),A(N,1),V(1)
  -----
  -----
```

[statements to set up the coefficients of the least squares normal equations, that is to evaluate

$$A_{ij} = \sum_{k=1}^{m} \frac{\partial r_k}{\partial x_i} \frac{\partial r_k}{\partial x_j} \qquad \text{for } i=1,2,...,n \text{ and } j=1,2,...,i$$

and

$$v_i = \sum_{k=1}^{m} r_k \frac{\partial r_k}{\partial x_i} \qquad \text{for } i=1,2,...,n$$

where r_k and $\partial r_k/\partial x_i$ are evaluated for $x_1,x_2,...,x_n$ as given in `X(1)`,`X(2)`,...,`X(N)`. In fact the values r_k, k=1,2,...,m will already have been evaluated at $x_1,x_2,...,x_n$ in an immediately previous call of `RESID`, and these values are available in `R(1),R(2),...,R(M)`]

```
  -----
  -----
RETURN
END
```

One way of programming `LSQ` is to declare an array of size `M`×`N` (`DR(M,N)` say), and to set `DR(K,I)` equal to $\partial r_k/\partial x_i$ for all `K=1,M` and all `I=1,N`. Then `A` and `V` can be evaluated using the statements

```
DO 1 I=1,N
  V(I)=SDOT(M,DR(1,I),1,R(1),1)
  DO 2 J=1,I
```

```
         A(I,J)=SDOT(M,DR(1,I),1,DR(1,J),1)
     2   CONTINUE
     1 CONTINUE
```

see the example of section 8. It may be possible to do this more efficiently without using `M*N` storage locations for `DR` – however note that `A` is overwritten by `VA27A` so that any constant elements in `A` must be reset. Because `A` is a symmetric matrix, *only the lower triangle need be set.*

In most problems, the calculation of derivatives in `LSQ` will involve terms (e.g. cosines, exponentials, etc.) which have already been calculated in `RESID.` It is usually very inefficient to recalculate such expressions and they should be passed from `RESID` to `LSQ` via a `COMMON` block – see the example in section 8. Alternatively `LSQ` can be written as if it were a secondary entry point to `RESID.`

For the sake of efficiency `LSQ` is only called if the sum of squares of residuals evaluated by `RESID` is an improvement on the best previously obtained. However in practice most calls of `RESID` are followed by a call of `LSQ.` This should be taken into account when setting the parameter `MAXFN`.

4 PRINTING

Printing starts on a new page with the text `ENTRY TO VA27A.` At the beginning of the first iteration and on every subsequent `IABS(IPRINT)` iterations. The numbers

```
IT  IR

SS

X(1),X(2),...,X(N)              (  8 to a line  )

V(1),V(2),...,V(N)              (       ”       )

R(1),R(2),..........,R(M)       (       ”       )
```

are printed as shown. `IT` is the previous number of iterations, `IR` is the number of calls of `RESID, X` is current best approximation, `R` and `SS` are the corresponding residuals and sum of squares, and `V` is the corresponding quantity defined in section 3(b), where in fact `2*V` is the gradient vector of the sum of squares. The same information is printed out on exit from `VA27A`, excepting that `V` is not given as it is not usually available.

Exceptions to the above occur if `IPRINT` = 0, when none of the above printing takes place, and if `IPRINT` < 0, when the printing of `R(1),...,R(M)` is suppressed. Furthermore diagnostics may be produced in certain error situations.

5 SCALING THE VARIABLES

At each iteration the equations

$$(A+\lambda\mathbf{D})\delta=-v$$

are solved to obtain a correction δ to the current approximation x. **D** is a constant diagonal matrix with $D_{ii} > 0$, and different choices of **D** correspond to different prescalings of the variables. D_{ii} is represented by the `I`-th element of the parameter `D` in the calling sequence of `VA27A` and may be specified in different ways by setting the parameter `MODE.` The following are permitted:

`MODE` = 1 (the normal setting): `D(I)` for `I=1,N` is set automatically by `VA27A` to `A(I,I)`, where `A` is the matrix calculated by `LSQ` from the given initial approximation, or to 1 if exceptionally `A(I,I) = 0`. This choice is described further in R.6799.

`MODE` = 2 `D(I)` is set automatically by `VA27A` to 1 (corresponding to no change of scale).

`MODE` = 3 `D(I)` is set by the user through the parameter `D` in the parameter list.

If `MODE` = 1 or 2 no user action is required as regards scaling.

6 CONSTRAINTS

When a `RETURN` is given in `RESID` with `IFL`=1, as described in section 3(a), then the iteration is repeated with a larger value of λ, causing a smaller correction to be made to the variables. This feature can be used to impose constraints in a limited and simple minded way. It is merely necessary in `RESID` to check whether the values $x_1, x_2,..., x_n$ violate any of the constraints on the variables, in which case the `INTEGER` variable `IFL` is set to 1 and a `RETURN` is made. This device is illustrated in the next section, and is worth trying when an unconstrained minimum is expected to exist, although success is by no means guaranteed.

7 GENERAL

Use of common: None.

Workspace: workspace is provided by the user through arguments `A` (n^2 elements) and `W` ($4n+m$ elements).

Restrictions: None.

8 AN EXAMPLE

Consider the problem of fitting the data $y(t_1), y(t_2),..., y(t_{25})$ by a function of the form

$$f(t) = a + bt + c\exp(-\tfrac{1}{2}(t-d)^2/e^2),$$

where *a*, *b*, *c*, *d* and *e* are parameters to be determined. The problem can be posed as that of choosing *a*, *b*,..., *e* so as to minimize

$$\sum_{i=1}^{25} [r_i(a,b,...,e)]^2$$

where

$$r_i(a,b,...,e) = a + bt_i + c\exp(-\tfrac{1}{2}(t_i-d)^2/e^2) - y(t_i).$$

Thus the problem has 25 residuals, 5 variables, and $t_1, t_2,..., t_{25}$ and $y(t_1), y(t_2),..., y(t_{25})$ are given data. The required partial derivatives are easily obtainable, namely

$$\partial\frac{r_i}{\partial}a = 1 \qquad \partial\frac{r_i}{\partial}b = t_i \qquad \partial\frac{r_i}{\partial}c = \exp(.....)$$

$$\partial\frac{r_i}{\partial}d = \frac{c(t_i\ d)}{e^2}\ exp(.....) \qquad \partial\frac{r_i}{\partial}e = \frac{c(t_i\ d)^2}{e^3}\ \exp(.....).$$

Note how the exponentials which occur in calculating r_i also occur in the calculation of the derivatives. Finally r_i will overflow if $e = 0$ and in fact a minimum subject to the constraint $e > 0$ is required.

The MAIN program for this problem is as follows

```
      REAL A(25),D(5),X(5),EPS(5),R(25),W(45)
      COMMON Y(25),T(25),EX(25),DR(25,5)
      EXTERNAL GAUSSR, GAUSSD
         -------
         -------
```

[statements to read $y(t_i) \rightarrow$ `Y(I)`, $t_i \rightarrow$ `T(I)`, to set `DR(I,1) = 1` and `DR(I,2) = T(I)`, for `I =` 1,2,...,25; also to set initial approximations to *a*,*b*,...,*e* into `X(1)`, `X(2)`, ..., `X(5)` and the respective tolerances into `EPS(1)`, `EPS(2)`, ..., `EPS(5)`.

```
         -------
         -------
      CALL VA27A(GAUSSR,GAUSSD,25,5,X,R,SS,A,D,EPS,1,100,1,W)
      STOP
      END
```

and the user subroutines are

```
      SUBROUTINE VAUSSR(M,N,X,R,IFL)
      DIMENSION V(1),R(1)
      COMMON Y(25),T(25),EX(25),DR(25,5)
      IF (X(5).LE.O.) THEN
        IFL=1
        RETURN
      ENDIF
      DO 1 I=1,M
        EX(I) = EXP(-.5-((T(I)-X(4))/X(5))**2)
        R(I) = X(1) + X(2)*T(I) + X(3)*EX(I)-Y(I)
    1 CONTINUE
      RETURN
      END
```

and

```
      SUBROUTINE GAUSSD(M,N,X,R,A,V)
      DIMENSION X(1),R(!),A(N,1),V(1)
      COMMON Y(25),T(25),EX(25),DR(25,5)
      DO 1 I=1,M
        DR(I,3) = EX(I)
        DR(I,4) = X(3)*(T(I)-X(4))*EX(I)/X(5)**2
        DR(I,5) = X(3)*(T(I)-X(4))**2*EX(I)/X(5)**3
    1 CONTINUE
      DO 3 I=1,N
        V(I)=FM02AS(M,DR(1,I),1,R(1),1)
        DO 2 J=1,I
          A(I,J)=FM02AS(M,DR(1,I),1,DR(1,J),1)
    2   CONTINUE
    3 CONTINUE
      RETURN
      END
```