# VF04

## 1 SUMMARY

This subroutine is used to calculate a local minimum of a general function of several variables subject to nonlinear equality and inequality constraints. Linear constraints may, of course, be included but they are not treated specially in this implementation.

The objective function is

$$F(x_1,x_2,...,x_n)$$

and the constraints are

$$c_i(x_1,x_2,...,x_n)=0,\ i=1,2,...,k$$

and

$$c_i(x_1,x_2,...,x_n)\geq 0,\ i=k+1,...,m.$$

The user is required to provide initial values of the variables and a subroutine for evaluating $F$ and $c_i$, i=1,2,...,m. Derivatives are **not** required. An iterative method is used in which each major iteration minimizes a multiplier penalty function (see Section 4). Convergence is forced by adjustment of the penalty and/or multiplier parameters. The subroutine may also be used to solve unconstrained problems (m=0), and will usually be more efficient than VA10. It may be necessary to scale the problem so that the values of the variables all have magnitude about unity.

**ATTRIBUTES** — **Version:** 1.0.0. **Types:** VF04A.; VF04AD. **Original date:** March 1986. **Origin:** I.D.Coope, University of Canterbury, New Zealand. **Remark:** If derivatives are easily coded then VF03AD is recommended.

## 2 HOW TO USE THE PACKAGE

The subroutine is designed to be easy to use, with few parameters in the calling sequence. However, extra parameters can be accessed through the use of a named COMMON area. In many cases users with simple, well-scaled problems need not be aware of these extra parameters and may prefer to skip Sections 2.3 and 2.4 below. A simple example is given in Section 5.

### 2.1 Argument list

*The single precision version*

```
CALL VF04A(FCSUB,N,M,K,X,F,ACC,W)
```

*The double precision version*

```
CALL VF04AD(FCSUB,N,M,K,X,F,ACC,W)
```

FCSUB is the name of the subroutine provided by the user which is considered in Section 2.2. It must be declared in an EXTERNAL statement.

N is an INTEGER variable that must be set by the user to the number, $n$, of variables. Its value is not altered by the subroutine. **Restriction**: N>0.

M is an INTEGER variable that must be set by the user to the number, $m$, of constraints on the variables. Its value is not altered by the subroutine. A zero value is allowed. **Restriction**: M≥0.

K is an INTEGER variable that must be set by the user to the number, $k$, of equality constraints on the variables. Its value is not altered by the subroutine. A zero value is allowed. **Restriction**: M≥K≥0.

X is a REAL (DOUBLE PRECISION in the D version) array of length $n$ for the values of the variables. Initially it must contain estimates of the values of the variables. VF04A/AD returns the solution in X.

F is a REAL (DOUBLE PRECISION in the D version) variable, which on return from VF04A/AD contains the final value of the objective function $F$ corresponding to X.

ACC is a REAL (DOUBLE PRECISION in the D version) variable that must be set by the user. It controls the accuracy of the calculation, and it is not altered by the subroutine. The calculation finishes when the objective function is predicted to be within ACC of its optimum value. Some constraints may be violated slightly but the estimated change in $F(x_1,x_2,...,x_n)$ that would occur on correcting these violations is less than ACC.

W is a REAL (DOUBLE PRECISION in the D version) array that is used for workspace. It must have length at least $n^2+8n+8m$. On a return from VF04A/AD values of the constraint functions, $c_i(x_1,x_2,...,x_n)$, i=1,2,...,m, are set in W(I), I=1,2,...,M.

### 2.2 User Subroutine

```
SUBROUTINE FCSUB(N,X,F,C)
```

The name of this subroutine must be the first argument in the call to VF04A/AD. N,X and F correspond to the arguments with these names in section 2.1. The elements of X are given to this subroutine by VF04A/AD and it must set the value of the objective function in F, and values of the constraint functions, if any, in C(*). An example is given in Section 5.

### 2.3 Common

VF04 contains the following named COMMON area:

*The single precision version*

```
COMMON/VF04D/IPRINT,LP,MAXITN,IH,IERR,MODE
```

*The double precision version*

```
COMMON/VF04DD/IPRINT,LP,MAXITN,IH,IERR,MODE
```

The six integers IPRINT,LP,MAXITN,IH,IERR,MODE have default values set in BLOCK DATA VF04C/CD. In many cases they can be ignored, but sometimes they are useful.

IPRINT is an INTEGER, with default value –1, to indicate how much printing is required. There is no output if IPRINT is zero. If it is positive then, in addition to any diagnostics, the values of $x_i$, $i$=1,2,...,$n$, $F(x_1,x_2,...,x_n)$ and $c_i$, $i$=1,2,...,$m$, are printed every IPRINT minor iterations. Lagrange multiplier estimates and penalty parameters are also printed at the end of each unconstrained minimization of the penalty function. Any negative value for IPRINT causes diagnostics and final values only to be printed.

LP is an INTEGER, with default value 6 which gives the UNIT number for printing. If LP≤0 there is no printing.

MAXITN is an INTEGER which bounds the number of major or minor iterations of the calculation. If it is zero it has no effect. A positive value causes VF04A/AD to terminate automatically when MAXITN minor iterations (line searches) have been completed. If it is negative, termination occurs when –MAXITN major iterations (unconstrained minimizations) have been completed. Normal termination may occur earlier. The default value is –30.

IH is an INTEGER with default value –6 which is be used to guide the choice of finite difference step-size for estimating derivative information. The actual step-size used is usually $10^{IH}$ but this value may be adjusted if it seems to be out of scale. The value of IH is not altered by the subroutine.

IERR is an INTEGER, which is used to record the status of a return from VF04A/AD. It has the value IERR=0 if the required accuracy has been achieved, but it may also have the values 1,2,3,4,5. They indicate error conditions and are explained in Section 2.4 below. Its value should be inspected if IPRINT=0 (no printing) has been set by the user.

MODE is an INTEGER with a default value of zero, in which case initialisation of Lagrange multiplier estimates, penalty parameters and second derivative information is performed automatically by VF04A/AD. If MODE is negative then no initialisation is performed; it is assumed that VF04A/AD has already been called once to solve a similar problem. If MODE≥1 then the user must set the initial values of the elements W(2M+I), K=I,2,...,M, before the call to VF04A/AD. These values are used to penalise constraint violations and they are increased automatically by the algorithm, if necessary, but they are never decreased. Large values have the effect of keeping constraint violations small but efficiency may be impaired if the initial values are too large. If this option is not invoked then default values W(2M+I)=10.0, I=1,2,...,M, are used. If MODE≥2 then initial estimates of Lagrange multipliers for the constraint functions $c_i$, $i$=1,2,...,$m$, must also be set by the user in the array elements W(M+I), I=1,2,...,M, before the call to VF04A/AD. If this option is not invoked then default values W(M+I)=0.0, I=1,2,...,M, are used.

### 2.4 Errors and diagnostic messages

The required accuracy is achieved when IERR=0 on a return from the subroutine. However, if IERR>0 then an error has occurred and if IPRINT≠0 then a diagnostic message is automatically printed. The following error returns are possible:

IERR=1 The calculation has terminated because the value of MAXITN has been reached. If this seems to be caused by the subroutine changing the variables very slowly, you should seek advice. Otherwise, increase the size of MAXITN.

IERR=2 The subroutine has failed to find a vector of variables that satisfies the constraints. It occurs if the constraints are inconsistent, but it may also occur if excessive accuracy is requested.

IERR=3 This error return occurs when the problem has an unbounded solution. It may also occur if the problem is badly scaled or if the initial values of the penalty parameters are too small. In the latter case the remedy is to make another call to VF04A/AD with MODE=1, after increasing the penalty parameters.

IERR=4 The line search procedure requires more than ten steps. This error is usual if the chosen finite difference interval is inappropriate because then the objective function of the line search may increase along a direction which is not a direction of descent, because of the inexact derivatives. This error return may also occur if the required accuracy cannot be achieved, in which case the changes in function values are dominated by computer rounding errors.

IERR=5 One or more of the conditions $n$>0, $m$≥$k$≥0, is violated by the values input in N, M, K.

## 3 GENERAL INFORMATION

**Use of common:** The subroutine uses common block VF04D/DD and BLOCK DATA VF04C/CD.

**Other routines called directly:** Uses VF04B/BD as a private subroutine.

**Input/output:** Output is under the control of argument LP.

**Restrictions:** N>0, M≥K≥0.

## 4 METHOD

The subroutine uses a special implementation of the BFGS method to minimize a Lagrange multiplier penalty function. The penalty function for the equality constrained case (m=k) is

$$P(\mathbf{x},\lambda,\sigma) = F(\mathbf{x}) - \sum_{i=1}^{m}\lambda_i c_i(\mathbf{x}) + \tfrac{1}{2}\sum_{i=1}^{m}\sigma_i c_i^2(\mathbf{x}).$$

A major iteration consists of minimizing the penalty function for fixed values of the penalty parameters, $\sigma_i$, i=1,...,m, and multiplier parameters, $\lambda_i$, i=1,...,m. Then if the constraint violations are not sufficiently small the penalty and/or multiplier parameters are adjusted automatically and another unconstrained minimization is begun.

The algorithm is similar to VF01AD in operation, except that derivatives of the penalty function are estimated by finite difference formulae, and a simpler formula is used to revise the Lagrange multiplier estimates.

## 5 EXAMPLE OF USE

We illustrate use of the subroutine on the problem with objective function

$$x_1^2 + x_2^2 + x_3$$

and constraints

$$x_1 x_2 = x_3$$

$$x_3 \geq 1$$

taking the point (1,2,3) as an initial guess. The following code finds the solution (1,1,1) after 6 major iterations of VF04AD.

```
      DOUBLE PRECISION X(3),F,ACC,W(100)
      EXTERNAL EXAMPL
      DATA X /1.0D0, 2.0D0, 3.0D0/
      N=3
      M=2
      K=1
      ACC=1.0D-7
      CALL VF04AD(EXAMPL,N,M,K,X,F,ACC,W)
      STOP
      END

      SUBROUTINE EXAMPL(N,X,F,C)
      DOUBLE PRECISION X(N),F,C(*)
      F   = X(1)**2 + X(2)**2 + X(3)
      C(1)= X(1)*X(2) - X(3)
      C(2)= X(3) - 1.0D0
      RETURN
      END
```

```
 ENTRY TO VF04AD;    REQUESTED ACCURACY IN FUNCTION VALUE IS   0.1D-06
 PROBLEM HAS  3 VARIABLES,  1 EQUALITY  AND  1 INEQUALITY CONSTRAINTS


 THE RESULTS FROM  VF04AD ARE AS FOLLOWS
 ---------------------------------------
 UNCONSTRAINED  MINIMIZATION  NUMBER   6
 ---------------------------------------
 ITERATION   22     FUNCTION CALLS   135
 ---------------------------------------
 F = 3.000000019875850
 VARIABLES
     0.100000000222D+01    0.100000000668D+01    0.100000000209D+01
 CONSTRAINT RESIDUAL(S)
     0.680851686141D-08    0.208627204401D-08
 LAGRANGE MULTIPLIER(S)
     0.199999999500D+01    0.300000017607D+01
 PENALTY PARAMETER(S)
     0.100000000000D+02    0.100000000000D+03
```