NAG Library Routine Document

C05RCF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

C05RCF is a comprehensive routine that finds a solution of a system of nonlinear equations by a modification of the Powell hybrid method. You must provide the Jacobian.

2 Specification

```
SUBROUTINE C05RCF (FCN, N, X, FVEC, FJAC, XTOL, MAXFEV, MODE, DIAG, FACTOR,     &
                   NPRINT, NFEV, NJEV, R, QTF, IUSER, RUSER, IFAIL)
INTEGER            N, MAXFEV, MODE, NPRINT, NFEV, NJEV, IUSER(*), IFAIL
REAL (KIND=nag_wp) X(N), FVEC(N), FJAC(N,N), XTOL, DIAG(N), FACTOR,             &
                   R(N*(N+1)/2), QTF(N), RUSER(*)
EXTERNAL           FCN
```

3 Description

The system of equations is defined as:

$$f_i(x_1, x_2, \ldots, x_n) = 0, \qquad i = 1, 2, \ldots, n.$$

C05RCF is based on the MINPACK routine HYBRJ (see Moré *et al.* (1980)). It chooses the correction at each step as a convex combination of the Newton and scaled gradient directions. The Jacobian is updated by the rank-1 method of Broyden. At the starting point, the Jacobian is requested, but it is not asked for again until the rank-1 method fails to produce satisfactory progress. For more details see Powell (1970).

4 References

Moré J J, Garbow B S and Hillstrom K E (1980) User guide for MINPACK-1 *Technical Report ANL-80-74* Argonne National Laboratory

Powell M J D (1970) A hybrid method for nonlinear algebraic equations *Numerical Methods for Nonlinear Algebraic Equations* (ed P Rabinowitz) Gordon and Breach

5 Parameters

1: FCN – SUBROUTINE, supplied by the user. *External Procedure*

Depending upon the value of IFLAG, FCN must either return the values of the functions f_i at a point x or return the Jacobian at x.

The specification of FCN is:

```
SUBROUTINE FCN (N, X, FVEC, FJAC, IUSER, RUSER, IFLAG)

INTEGER            N, IUSER(*), IFLAG
REAL (KIND=nag_wp) X(N), FVEC(N), FJAC(N,N), RUSER(*)
```

1: N – INTEGER *Input*

On entry: n, the number of equations.

2: X(N) – REAL (KIND=nag_wp) array *Input*

On entry: the components of the point x at which the functions or the Jacobian must be evaluated.

3: FVEC(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: if IFLAG $= 0$ or 2, FVEC contains the function values $f_i(x)$ and must not be changed.

On exit: if IFLAG $= 1$ on entry, FVEC must contain the function values $f_i(x)$ (unless IFLAG is set to a negative value by FCN).

4: FJAC(N,N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: if IFLAG $= 0$, FJAC(i,j) contains the value of $\frac{\partial f_i}{\partial x_j}$ at the point x, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$. When IFLAG $= 0$ or 1, FJAC must not be changed.

On exit: if IFLAG $= 2$ on entry, FJAC(i,j) must contain the value of $\frac{\partial f_i}{\partial x_j}$ at the point x, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$, (unless IFLAG is set to a negative value by FCN).

5: IUSER($*$) – INTEGER array *User Workspace*
6: RUSER($*$) – REAL (KIND=nag_wp) array *User Workspace*

FCN is called with the parameters IUSER and RUSER as supplied to C05RCF. You are free to use the arrays IUSER and RUSER to supply information to FCN as an alternative to using COMMON global variables.

7: IFLAG – INTEGER *Input/Output*

On entry: IFLAG $= 0$, 1 or 2.

IFLAG $= 0$
X, FVEC and FJAC are available for printing (see NPRINT).

IFLAG $= 1$
FVEC is to be updated.

IFLAG $= 2$
FJAC is to be updated.

On exit: in general, IFLAG should not be reset by FCN. If, however, you wish to terminate execution (perhaps because some illegal point X has been reached), then IFLAG should be set to a negative integer value.

FCN must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which C05RCF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

2: N – INTEGER *Input*

On entry: n, the number of equations.

Constraint: N > 0.

3: X(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: an initial guess at the solution vector.

On exit: the final estimate of the solution vector.

4: FVEC(N) – REAL (KIND=nag_wp) array *Output*

On exit: the function values at the final point returned in X.

5: FJAC(N,N) – REAL (KIND=nag_wp) array *Output*

On exit: the orthogonal matrix Q produced by the QR factorization of the final approximate Jacobian.

6: XTOL – REAL (KIND=nag_wp) *Input*

On entry: the accuracy in X to which the solution is required.

Suggested value: $\sqrt{\epsilon}$, where ϵ is the ***machine precision*** returned by X02AJF.

Constraint: $\text{XTOL} \geq 0.0$.

7: MAXFEV – INTEGER *Input*

On entry: the maximum number of calls to FCN with $\text{IFLAG} \neq 0$. C05RCF will exit with $\text{IFAIL} = 2$, if, at the end of an iteration, the number of calls to FCN exceeds MAXFEV.

Suggested value: $\text{MAXFEV} = 100 \times (\text{N} + 1)$.

Constraint: $\text{MAXFEV} > 0$.

8: MODE – INTEGER *Input*

On entry: indicates whether or not you have provided scaling factors in DIAG.

If $\text{MODE} = 2$ the scaling must have been specified in DIAG.

Otherwise, if $\text{MODE} = 1$, the variables will be scaled internally.

Constraint: $\text{MODE} = 1$ or 2.

9: DIAG(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: if $\text{MODE} = 2$, DIAG must contain multiplicative scale factors for the variables.

If $\text{MODE} = 1$, DIAG need not be set.

Constraint: if $\text{MODE} = 2$, $\text{DIAG}(i) > 0.0$, for $i = 1, 2, \ldots, n$.

On exit: the scale factors actually used (computed internally if $\text{MODE} = 1$).

10: FACTOR – REAL (KIND=nag_wp) *Input*

On entry: a quantity to be used in determining the initial step bound. In most cases, FACTOR should lie between 0.1 and 100.0. (The step bound is $\text{FACTOR} \times \|\text{DIAG} \times \text{X}\|_2$ if this is nonzero; otherwise the bound is FACTOR.)

Suggested value: $\text{FACTOR} = 100.0$.

Constraint: $\text{FACTOR} > 0.0$.

11: NPRINT – INTEGER *Input*

On entry: indicates whether (and how often) special calls to FCN, with IFLAG set to 0, are to be made for printing purposes.

$\text{NPRINT} \leq 0$
: No calls are made.

$\text{NPRINT} > 0$
: FCN is called at the beginning of the first iteration, every NPRINT iterations thereafter and immediately before the return from C05RCF.

12: NFEV – INTEGER *Output*

On exit: the number of calls made to FCN to evaluate the functions.

13: NJEV – INTEGER *Output*

On exit: the number of calls made to FCN to evaluate the Jacobian.

14: $R(N \times (N+1)/2)$ – REAL (KIND=nag_wp) array *Output*

On exit: the upper triangular matrix R produced by the QR factorization of the final approximate Jacobian, stored row-wise.

15: QTF(N) – REAL (KIND=nag_wp) array *Output*

On exit: the vector $Q^{\mathrm{T}}f$.

16: IUSER($*$) – INTEGER array *User Workspace*
17: RUSER($*$) – REAL (KIND=nag_wp) array *User Workspace*

IUSER and RUSER are not used by C05RCF, but are passed directly to FCN and may be used to pass information to this routine as an alternative to using COMMON global variables.

18: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 2$

There have been at least MAXFEV calls to FCN to evaluate the functions. Consider restarting the calculation from the final point held in X.

IFAIL $= 3$

No further improvement in the approximate solution X is possible; XTOL is too small.

IFAIL $= 4$

The iteration is not making good progress, as measured by the improvement from the last five Jacobian evaluations.

IFAIL $= 5$

The iteration is not making good progress, as measured by the improvement from the last ten iterations.

IFAIL $= 6$

You have set IFLAG negative in FCN.

IFAIL $= 11$

On entry, $\mathrm{N} \le 0$.

IFAIL $= 12$

On entry, $\mathrm{XTOL} < 0.0$.

IFAIL $= 13$

On entry, $\mathrm{MODE} \ne 1$ or 2.

IFAIL $= 14$

On entry, $\mathrm{FACTOR} \le 0.0$.

IFAIL $= 15$

On entry, $\mathrm{MODE} = 2$ and $\mathrm{DIAG}(i) \le 0.0$ for some i, for $i = 1, 2, \ldots, \mathrm{N}$.

IFAIL $= 18$

On entry, $\mathrm{MAXFEV} \le 0$.

IFAIL $= -999$

Internal memory allocation failed.

A value of IFAIL $= 4$ or 5 may indicate that the system does not have a zero, or that the solution is very close to the origin (see Section 7). Otherwise, rerunning C05RCF from a different starting point may avoid the region of difficulty.

7 Accuracy

If $\hat{x}$ is the true solution and D denotes the diagonal matrix whose entries are defined by the array DIAG, then C05RCF tries to ensure that

$$\|D(x - \hat{x})\|_2 \le \mathrm{XTOL} \times \|D\hat{x}\|_2.$$

If this condition is satisfied with $\mathrm{XTOL} = 10^{-k}$, then the larger components of Dx have k significant decimal digits. There is a danger that the smaller components of Dx may have large relative errors, but the fast rate of convergence of C05RCF usually obviates this possibility.

If XTOL is less than ***machine precision*** and the above test is satisfied with the ***machine precision*** in place of XTOL, then the routine exits with IFAIL $= 3$.

Note: this convergence test is based purely on relative error, and may not indicate convergence if the solution is very close to the origin.

The convergence test assumes that the functions and the Jacobian are coded consistently and that the functions are reasonably well behaved. If these conditions are not satisfied, then C05RCF may incorrectly indicate convergence. The coding of the Jacobian can be checked using C05ZDF. If the Jacobian is coded correctly, then the validity of the answer can be checked by rerunning C05RCF with a lower value for XTOL.

8 Further Comments

Local workspace arrays of fixed lengths are allocated internally by C05RCF. The total size of these arrays amounts to $4 \times n$ real elements.

The time required by C05RCF to solve a given problem depends on n, the behaviour of the functions, the accuracy requested and the starting point. The number of arithmetic operations executed by C05RCF is approximately $11.5 \times n^2$ to process each evaluation of the functions and approximately $1.3 \times n^3$ to process each evaluation of the Jacobian. The timing of C05RCF is strongly influenced by the time spent evaluating the functions.

Ideally the problem should be scaled so that, at the solution, the function values are of comparable magnitude.

9 Example

This example determines the values $x_1, \ldots, x_9$ which satisfy the tridiagonal equations:

$$\begin{aligned} (3 - 2x_1)x_1 - 2x_2 &= -1, \\ -x_{i-1} + (3 - 2x_i)x_i - 2x_{i+1} &= -1, \qquad i = 2, 3, \ldots, 8 \\ -x_8 + (3 - 2x_9)x_9 &= -1. \end{aligned}$$

9.1 Program Text

```
!   C05RCF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module c05rcfe_mod

!     C05RCF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter        :: factor = 100.0_nag_wp
      Integer, Parameter                   :: maxfev = 1000, mode = 2, n = 9,  &
                                              nout = 6, nprint = 0
    Contains
      Subroutine fcn(n,x,fvec,fjac,iuser,ruser,iflag)

!       .. Parameters ..
        Real (Kind=nag_wp), Parameter        ::                                &
      coeff(5) = (/-1.0_nag_wp,3.0_nag_wp,-2.0_nag_wp,-2.0_nag_wp,-1.0_nag_wp/)
!       .. Scalar Arguments ..
        Integer, Intent (Inout)              :: iflag
        Integer, Intent (In)                 :: n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout)   :: fjac(n,n), fvec(n), ruser(*)
        Real (Kind=nag_wp), Intent (In)      :: x(n)
        Integer, Intent (Inout)              :: iuser(*)
!       .. Local Scalars ..
        Integer                              :: k
!       .. Executable Statements ..
        If (iflag==0) Then

!         Insert print statements here when NPRINT is positive.

          Continue
        Else If (iflag/=2) Then
          fvec(1:n) = (coeff(2)+coeff(3)*x(1:n))*x(1:n) - coeff(5)
          fvec(2:n) = fvec(2:n) + coeff(1)*x(1:(n-1))
          fvec(1:(n-1)) = fvec(1:(n-1)) + coeff(4)*x(2:n)
        Else
          fjac(1:n,1:n) = 0.0_nag_wp

          fjac(1,1) = coeff(2) + 2.0_nag_wp*coeff(3)*x(1)
          fjac(1,2) = coeff(4)
          Do k = 2, n - 1
            fjac(k,k-1) = coeff(1)
            fjac(k,k) = coeff(2) + 2.0_nag_wp*coeff(3)*x(k)
            fjac(k,k+1) = coeff(4)
          End Do
          fjac(n,n-1) = coeff(1)
          fjac(n,n) = coeff(2) + 2.0_nag_wp*coeff(3)*x(n)

        End If
```

```
        Return

      End Subroutine fcn
    End Module c05rcfe_mod
    Program c05rcfe

!     C05RCF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: c05rcf, dnrm2, nag_wp, x02ajf
      Use c05rcfe_mod, Only: factor, fcn, maxfev, mode, n, nout, nprint
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                 :: fnorm, xtol
      Integer                            :: ifail, j, nfev, njev
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable    :: diag(:), fjac(:,:), fvec(:),     &
                                            qtf(:), r(:), x(:)
      Real (Kind=nag_wp)                 :: ruser(1)
      Integer                            :: iuser(1)
!     .. Intrinsic Procedures ..
      Intrinsic                          :: sqrt
!     .. Executable Statements ..
      Write (nout,*) 'C05RCF Example Program Results'

      Allocate (diag(n),fjac(n,n),fvec(n),qtf(n),r(n*(n+1)/2),x(n))

!     The following starting values provide a rough solution.

      x(1:n) = -1.0_nag_wp

      xtol = sqrt(x02ajf())
      diag(1:n) = 1.0_nag_wp

      ifail = -1
      Call c05rcf(fcn,n,x,fvec,fjac,xtol,maxfev,mode,diag,factor,nprint,nfev, &
        njev,r,qtf,iuser,ruser,ifail)

      Select Case (ifail)
      Case (0)
!       The NAG name equivalent of dnrm2 is f06ejf
        fnorm = dnrm2(n,fvec,1)
        Write (nout,*)
        Write (nout,99999) 'Final 2-norm of the residuals =', fnorm
        Write (nout,*)
        Write (nout,*) 'Final approximate solution'
        Write (nout,*)
        Write (nout,99998)(x(j),j=1,n)
      Case (2:5)
        Write (nout,*)
        Write (nout,*) 'Approximate solution:'
        Write (nout,*)
        Write (nout,99998)(x(j),j=1,n)
      End Select

99999 Format (1X,A,E12.4)
99998 Format (1X,3F12.4)
    End Program c05rcfe
```

9.2 Program Data

None.

9.3 Program Results

```
C05RCF Example Program Results

Final 2-norm of the residuals =  0.1193E-07

Final approximate solution

     -0.5707     -0.6816     -0.7017
     -0.7042     -0.7014     -0.6919
     -0.6658     -0.5960     -0.4164
```