NAG Library Routine Document

C05RBF

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

C05RBF is an easy-to-use 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 C05RBF (FCN, N, X, FVEC, FJAC, XTOL, IUSER, RUSER, IFAIL)
INTEGER             N, IUSER(*), IFAIL
REAL (KIND=nag_wp)  X(N), FVEC(N), FJAC(N,N), XTOL, RUSER(*)
EXTERNAL            FCN
```

3 Description

The system of equations is defined as:

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

C05RBF is based on the MINPACK routine HYBRJ1 (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 $= 2$, 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 $= 1$, FJAC contains the value of $\dfrac{\partial f_i}{\partial x_j}$ at the point x, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$, and must not be changed.

On exit: if IFLAG $= 2$ on entry, FJAC(i,j) must contain the value of $\dfrac{\partial f_i}{\partial x_j}$ 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 C05RBF. 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 $= 1$ or 2.

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.

FCN must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which C05RBF 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: $\mathrm{XTOL} \geq 0.0$.

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

IUSER and RUSER are not used by C05RBF, but are passed directly to FCN and may be used to pass information to this routine as an alternative to using COMMON global variables.

9: 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: $\mathrm{IFAIL} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

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

Errors or warnings detected by the routine:

$\mathrm{IFAIL} = 2$

There have been at least $100 \times (\mathrm{N} + 1)$ calls to FCN. Consider restarting the calculation from the point held in X.

$\mathrm{IFAIL} = 3$

No further improvement in the solution is possible. XTOL is too small: $\mathrm{XTOL} = \langle value \rangle$.

$\mathrm{IFAIL} = 4$

The iteration is not making good progress. This failure exit 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 C05RBF from a different starting point may avoid the region of difficulty.

$\mathrm{IFAIL} = 5$

IFLAG was set negative in FCN. $\mathrm{IFLAG} = \langle value \rangle$.

$\mathrm{IFAIL} = 11$

On entry, $\mathrm{N} = \langle value \rangle$.
Constraint: $\mathrm{N} > 0$.

$\mathrm{IFAIL} = 12$

On entry, $\mathrm{XTOL} = \langle value \rangle$.
Constraint: $\mathrm{XTOL} \geq 0.0$.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

If $\hat{x}$ is the true solution, C05RBF tries to ensure that

$$\|x - \hat{x}\|_2 \leq \mathrm{XTOL} \times \|\hat{x}\|_2.$$

If this condition is satisfied with $\mathrm{XTOL} = 10^{-k}$, then the larger components of x have k significant decimal digits. There is a danger that the smaller components of x may have large relative errors, but the fast rate of convergence of C05RBF 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 C05RBF 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 C05RBF with a lower value for XTOL.

8 Parallelism and Performance

C05RBF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

C05RBF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

Local workspace arrays of fixed lengths are allocated internally by C05RBF. The total size of these arrays amounts to $n \times (n + 13)/2$ real elements.

The time required by C05RBF 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 C05RBF 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 C05RBF 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.

10 Example

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

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

10.1 Program Text

```
!   C05RBF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

    Module c05rbfe_mod

!     C05RBF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                               :: fcn
!     .. Parameters ..
      Integer, Parameter, Public           :: n = 9, nout = 6
    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/=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
!       Set iflag negative to terminate execution for any reason.
        iflag = 0
        Return
      End Subroutine fcn
    End Module c05rbfe_mod
    Program c05rbfe
```

```
!     C05RBF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: c05rbf, dnrm2, nag_wp, x02ajf
      Use c05rbfe_mod, Only: fcn, n, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                    :: fnorm, xtol
      Integer                               :: i, ifail
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable       :: fjac(:,:), fvec(:), x(:)
      Real (Kind=nag_wp)                    :: ruser(1)
      Integer                               :: iuser(1)
!     .. Intrinsic Procedures ..
      Intrinsic                             :: sqrt
!     .. Executable Statements ..
      Write (nout,*) 'C05RBF Example Program Results'

      Allocate (fjac(n,n),fvec(n),x(n))

!     The following starting values provide a rough solution.

      x(1:n) = -1.0E0_nag_wp

      xtol = sqrt(x02ajf())

      ifail = -1
      Call c05rbf(fcn,n,x,fvec,fjac,xtol,iuser,ruser,ifail)

      If (ifail==0 .Or. ifail==2 .Or. ifail==3 .Or. ifail==4) Then
        If (ifail==0) Then
!         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'
        Else
          Write (nout,*)
          Write (nout,*) 'Approximate solution'
        End If
        Write (nout,*)
        Write (nout,99998)(x(i),i=1,n)
      End If

99999 Format (1X,A,E12.4)
99998 Format (1X,3F12.4)
    End Program c05rbfe
```

10.2 Program Data

None.

10.3 Program Results

```
 C05RBF 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
```