NAG Library Routine Document

C05NBF

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

C05NBF is an easy-to-use routine that finds a solution of a system of nonlinear equations by a modification of the Powell hybrid method.

2 Specification

```
SUBROUTINE C05NBF (FCN, N, X, FVEC, XTOL, WA, LWA, IFAIL)

INTEGER             N, LWA, IFAIL
REAL (KIND=nag_wp) X(N), FVEC(N), XTOL, WA(1)
EXTERNAL            FCN
```

3 Description

The system of equations is defined as:

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

C05NBF is based on the MINPACK routine HYBRD1 (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 approximated by forward differences, but these are not used 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*

FCN must return the values of the functions f_i at a point x.

The specification of FCN is:

```
SUBROUTINE FCN (N, X, FVEC, IFLAG)

INTEGER             N, IFLAG
REAL (KIND=nag_wp) X(N), FVEC(N)
```

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 must be evaluated.

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

On exit: the function values $f_i(x)$ (unless IFLAG is set to a negative value by FCN).

4: IFLAG – INTEGER *Input/Output*

On entry: IFLAG > 0.

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. This value will be returned through IFAIL.

FCN must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which C05NBF 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: 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: XTOL ≥ 0.0.

6: WA(1) – REAL (KIND=nag_wp) array *Input*
7: LWA – INTEGER *Input*

These parameters are no longer accessed by C05NBF. Workspace is provided internally by dynamic allocation instead.

8: 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 < 0

You have set IFLAG negative in FCN. The value of IFAIL will be the same as your setting of IFLAG.

IFAIL $= 1$

On entry, N ≤ 0,
or XTOL < 0.0.

IFAIL $= 2$

There have been at least $200 \times (\mathrm{N} + 1)$ evaluations of FCN. 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. 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 C05NBF from a different starting point may avoid the region of difficulty.

IFAIL $= -999$

Internal memory allocation failed.

7 Accuracy

If $\hat{x}$ is the true solution, C05NBF tries to ensure that

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

If this condition is satisfied with 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 C05NBF 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 test assumes that the functions are reasonably well behaved. If this condition is not satisfied, then C05NBF may incorrectly indicate convergence. The validity of the answer can be checked, for example, by rerunning C05NBF with a lower value for XTOL.

8 Further Comments

Local workspace arrays of fixed lengths are allocated internally by C05NBF. The total size of these arrays amounts to $\mathrm{N} \times (3 \times \mathrm{N} + 13)/2$ real elements.

The time required by C05NBF 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 C05NBF to process each call of FCN is about $11.5 \times n^2$. Unless FCN can be evaluated quickly, the timing of C05NBF will be strongly influenced by the time spent in FCN.

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

```
!   C05NBF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module c05nbfe_mod

!     C05NBF 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        :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter        :: three = 3.0_nag_wp
      Real (Kind=nag_wp), Parameter        :: two = 2.0_nag_wp
      Integer, Parameter                   :: n = 9, nout = 6
    Contains
      Subroutine fcn(n,x,fvec,iflag)

!       .. Scalar Arguments ..
        Integer, Intent (Inout)              :: iflag
        Integer, Intent (In)                 :: n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out)     :: fvec(n)
        Real (Kind=nag_wp), Intent (In)      :: x(n)
!       .. Executable Statements ..
        fvec(1:n) = three*x(1:n) - two*x(1:n)*x(1:n) + one
        fvec(2:n) = fvec(2:n) - x(1:(n-1))
        fvec(1:(n-1)) = fvec(1:(n-1)) - two*x(2:n)

        Return

      End Subroutine fcn
    End Module c05nbfe_mod
    Program c05nbfe

!     C05NBF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: c05nbf, dnrm2, nag_wp, x02ajf
      Use c05nbfe_mod, Only: fcn, n, nout, one
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: fnorm, xtol
      Integer                              :: i, ifail, j, lwa
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: fvec(:), x(:)
      Real (Kind=nag_wp)                   :: wa(1)
!     .. Intrinsic Procedures ..
      Intrinsic                            :: sqrt
!     .. Executable Statements ..
      Write (nout,*) 'C05NBF Example Program Results'
```

```
      Allocate (fvec(n),x(n))

!     The following starting values provide a rough solution.

      x(1:n) = -one
      xtol = sqrt(x02ajf())

      ifail = -1
      Call c05nbf(fcn,n,x,fvec,xtol,wa,lwa,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:)
        Write (nout,*)
        Write (nout,*) 'Approximate solution'
        Write (nout,*)
        Write (nout,99998)(x(i),i=1,n)
      End Select

99999 Format (1X,A,E12.4)
99998 Format (1X,3F12.4)
    End Program c05nbfe
```

9.2 Program Data

None.

9.3 Program Results

```
 C05NBF 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
```