NAG Library Routine Document

C05PBF/C05PBA

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

C05PBF/C05PBA 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

2.1 Specification for C05PBF

```
SUBROUTINE C05PBF (FCN, N, X, FVEC, FJAC, LDFJAC, XTOL, WA, LWA, IFAIL)

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

2.2 Specification for C05PBA

```
SUBROUTINE C05PBA (FCN, N, X, FVEC, FJAC, LDFJAC, XTOL, WA, LWA, IUSER,         &
                   RUSER, IFAIL)

INTEGER            N, LDFJAC, LWA, IUSER(*), IFAIL
REAL (KIND=nag_wp) X(N), FVEC(N), FJAC(LDFJAC,N), XTOL, WA(1), RUSER(*)
EXTERNAL           FCN
```

3 Description

The system of equations is defined as:

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

C05PBF/C05PBA 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 calculated, but it is not recalculated 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 for C05PBF is:

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

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

The specification of FCN for C05PBA is:

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

INTEGER            N, LDFJAC, IFLAG, IUSER(*)
REAL (KIND=nag_wp) X(N), FVEC(N), FJAC(LDFJAC,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(LDFJAC,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: LDFJAC – INTEGER *Input*

On entry: the first dimension of the array FJAC as declared in the (sub)program from which C05PBF/C05PBA is called.

6: 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. This value will be returned through IFAIL.

Note: *the following are additional parameters for specific use with C05PBA. Users of C05PBF therefore need not read the remainder of this description.*

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

FCN is called with the parameters IUSER and RUSER as supplied to C05PBF/C05PBA. You are free to use the arrays IUSER and RUSER to supply information to FCN as an alternative to using COMMON global variables.

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. 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 C05PBF/C05PBA 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(LDFJAC,N) – REAL (KIND=nag_wp) array *Output*

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

6: LDFJAC – INTEGER *Input*

On entry: the first dimension of the array FJAC as declared in the (sub)program from which C05PBF/C05PBA is called.

Constraint: LDFJAC $\geq$ N.

7: 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 $\geq$ 0.0.

8: WA(1) – REAL (KIND=nag_wp) array *Input*
9: LWA – INTEGER *Input*

These parameters are no longer accessed by C05PBF/C05PBA. Workspace is provided internally by dynamic allocation instead.

10: IFAIL – INTEGER *Input/Output*

Note: *for C05PBA, IFAIL does not occur in this position in the parameter list. See the additional parameters described below.*

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).

Note: *the following are additional parameters for specific use with C05PBA. Users of C05PBF therefore need not read the remainder of this description*.

10: IUSER(∗) – INTEGER array *User Workspace*
11: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

IUSER and RUSER are not used by C05PBF/C05PBA, but are passed directly to FCN and may be used to pass information to this routine as an alternative to using COMMON global variables.

12: IFAIL – INTEGER *Input/Output*

Note: see the parameter description for IFAIL above.

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

A negative value of IFAIL indicates an exit from C05PBF/C05PBA because 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 LDFJAC < N,
or XTOL < 0.0,

IFAIL = 2

There have been 100 × (N + 1) evaluations of 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. 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 C05PBF/C05PBA 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, C05PBF/C05PBA tries to ensure that

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

If this condition is satisfied with $\text{XTOL} = 10^{-k}$, 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 C05PBF/C05PBA 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 and the Jacobian are coded consistently and that the functions are reasonably well behaved. If these conditions are not satisfied, then C05PBF/C05PBA may incorrectly indicate convergence. The coding of the Jacobian can be checked using C05ZAF. If the Jacobian is coded correctly, then the validity of the answer can be checked by rerunning C05PBF/C05PBA with a lower value for XTOL.

8 Further Comments

Local workspace arrays of fixed lengths are allocated internally by C05PBF/C05PBA. The total size of these arrays amounts to $\text{N} \times (\text{N} + 13)/2$ real elements.

The time required by C05PBF/C05PBA 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 C05PBF/C05PBA is about $11.5 \times n^2$ to process each evaluation of the functions and about $1.3 \times n^3$ to process each evaluation of the Jacobian. Unless FCN can be evaluated quickly, the timing of C05PBF/C05PBA 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

```
!   C05PBF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module c05pbfe_mod

!     C05PBF 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
      Integer, Parameter                   :: ldfjac = n
    Contains
      Subroutine fcn(n,x,fvec,fjac,ldfjac,iflag)

!       .. Scalar Arguments ..
        Integer, Intent (Inout)              :: iflag
        Integer, Intent (In)                 :: ldfjac, n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout)   :: fjac(ldfjac,n), fvec(n)
        Real (Kind=nag_wp), Intent (In)      :: x(n)
!       .. Local Scalars ..
        Integer                              :: k
!       .. Executable Statements ..
        If (iflag/=2) Then
          fvec(1:n) = (three-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)
        Else
          fjac(1:n,1:n) = 0.0_nag_wp

          fjac(1,1) = three - two*two*x(1)
          fjac(1,2) = -two
          Do k = 2, n - 1
            fjac(k,k) = three - two*two*x(k)
            fjac(k,k-1) = -one
            fjac(k,k+1) = -two
          End Do
          fjac(n,n-1) = -one
          fjac(n,n) = three - two*two*x(n)

        End If

        Return

      End Subroutine fcn
    End Module c05pbfe_mod
    Program c05pbfe

!     C05PBF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: c05pbf, dnrm2, nag_wp, x02ajf
      Use c05pbfe_mod, Only: fcn, ldfjac, n, nout, one
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: fnorm, xtol
      Integer                              :: ifail, j, lwa
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: fjac(:,:), fvec(:), x(:)
      Real (Kind=nag_wp)                   :: wa(1)
!     .. Intrinsic Procedures ..
      Intrinsic                            :: sqrt
!     .. Executable Statements ..
      Write (nout,*) 'C05PBF Example Program Results'

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

!     The following starting values provide a rough solution.

      x(1:n) = -one

      xtol = sqrt(x02ajf())

      ifail = -1
      Call c05pbf(fcn,n,x,fvec,fjac,ldfjac,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(j),j=1,n)
      End Select

99999 Format (1X,A,E12.4)
99998 Format (1X,3F12.4)
    End Program c05pbfe
```

9.2 Program Data

None.

9.3 Program Results

```
 C05PBF 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
```