D02BJF – NAG Fortran Library Routine Document

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.

Note. This routine was introduced into the NAG Fortran Library at Mark 19 and may therefore not be available to all users of the NAG Fortran SMP Library.

1 Purpose

D02BJF integrates a system of first-order ordinary differential equations over an interval with suitable initial conditions, using a fixed order Runge–Kutta method (RK), until a user-specified function, if supplied, of the solution is zero, and returns the solution at points specified by the user, if desired.

2 Specification

```
      SUBROUTINE D02BJF(X, XEND, N, Y, FCN, TOL, RELABS, OUTPUT, G, W,
     1                  IFAIL)
      real              X, XEND, Y(N), TOL, G, W(20*N)
      INTEGER           N, IFAIL
      CHARACTER*1       RELABS
      EXTERNAL          FCN, OUTPUT, G
```

3 Description

The routine advances the solution of a system of ordinary differential equations

$$y_i' = f_i(x, y_1, y_2, \ldots, y_n), \quad i = 1, 2, \ldots, n,$$

from $x = \mathrm{X}$ to $x = \mathrm{XEND}$ using a fixed order Runge–Kutta method. The system is defined by a subroutine FCN supplied by the user, which evaluates f_i in terms of x and $y = (y_1, y_2, \ldots, y_n)$. The initial values of $y = (y_1, y_2, \ldots, y_n)$ must be given at $x = \mathrm{X}$.

The solution is returned via the user-supplied subroutine OUTPUT at points specified by the user, if desired: this solution is obtained by C^1 interpolation on solution values produced by the method. As the integration proceeds a check can be made on the user-specified function $g(x, y)$ to determine an interval where it changes sign. The position of this sign change is then determined accurately by C^1 interpolation to the solution. It is assumed that $g(x, y)$ is a continuous function of the variables, so that a solution of $g(x, y) = 0$ can be determined by searching for a change in sign in $g(x, y)$. The accuracy of the integration, the interpolation and, indirectly, of the determination of the position where $g(x, y) = 0$, is controlled by the parameters TOL and RELABS.

4 References

[1] Shampine L F (1994) *Numerical solution of ordinary differential equations* Chapman and Hall

5 Parameters

1: X — ***real*** *Input/Output*

On entry: the initial value of the independent variable x.

On exit: if g is supplied by the user, it contains the point where $g(x, y) = 0$, unless $g(x, y) \neq 0$ anywhere on the range X to XEND, in which case, X will contain XEND (and the error indicator IFAIL = 6 is set); if g is not supplied by the user it contains XEND. However, if an error has occurred, it contains the value of x at which the error occurred.

2: XEND — ***real*** *Input*

On entry: the final value of the independent variable. If XEND < X, integration will proceed in the negative direction.

Constraint: XEND $\neq$ X.

3: N — INTEGER *Input*

On entry: the number of equations, n.

Constraint: N > 0.

4: Y(N) — ***real*** array *Input/Output*

On entry: the initial values of the solution $y_1, y_2, \ldots, y_n$ at $x = \mathrm{X}$.

On exit: the computed values of the solution at the final point $x = \mathrm{X}$.

5: FCN — SUBROUTINE, supplied by the user. *External Procedure*

FCN must evaluate the functions f_i (i.e., the derivatives y_i') for given values of its arguments $x, y_1, \ldots, y_n$.

Its specification is:

```
      SUBROUTINE FCN(X, Y, F)
      real           X, Y(*), F(*)
```

1: X — ***real*** *Input*

On entry: the value of the independent variable x.

2: Y(∗) — ***real*** array *Input*

On entry: the value of the variable y_i, for $i = 1, 2, \ldots, n$.

3: F(∗) — ***real*** array *Output*

On exit: the value of f_i, for $i = 1, 2, \ldots, n$.

FCN must be declared as EXTERNAL in the (sub)program from which D02BJF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

6: TOL — ***real*** *Input*

On entry: a **positive** tolerance for controlling the error in the integration. Hence TOL affects the determination of the position where $g(x, y) = 0$, if g is supplied.

D02BJF has been designed so that, for most problems, a reduction in TOL leads to an approximately proportional reduction in the error in the solution. However, the actual relation between TOL and the accuracy achieved cannot be guaranteed. The user is strongly recommended to call D02BJF with more than one value for TOL and to compare the results obtained to estimate their accuracy. In the absence of any prior knowledge, the user might compare the results obtained by calling D02BJF with RELABS set to 'D' and with each of TOL $= 10.0^{-p}$ and TOL $= 10.0^{-p-1}$ where p correct significant digits are required in the solution, y. The accuracy of the value x such that $g(x, y) = 0$ is indirectly controlled by varying TOL. The user should experiment to determine this accuracy.

Constraint: $10.0 \times$ ***machine precision*** < TOL < 0.01.

7: RELABS — CHARACTER*1 *Input*

On entry: the type of error control. At each step in the numerical solution an estimate of the local error, EST, is made. For the current step to be accepted the following condition must be satisfied:

$$\mathrm{EST} = \max(e_i/(\tau_r \times \max(|y_i|, \tau_a))) \leq 1.0$$

where τ_r and τ_a are defined by

RELABS	τ_r	τ_a
'M'	TOL	1.0
'A'	ϵ_r	TOL/ϵ_r
'R'	TOL	ϵ_a
'D'	TOL	ϵ_a

where ϵ_r and ϵ_a are small machine-dependent numbers and e_i is an estimate of the local error at y_i, computed internally. If the condition is not satisfied, the step size is reduced and the solution is recomputed on the current step. If the user wishes to measure the error in the computed solution in terms of the number of correct decimal places, then RELABS should be set to 'A' on entry, whereas if the error requirement is in terms of the number of correct significant digits, then RELABS should be set to 'R'. If the user prefers a mixed error test, then RELABS should be set to 'M', otherwise if the user has no preference, RELABS should be set to the default 'D'. Note that in this case 'D' is taken to be 'R'.

Constraint: RELABS = 'M', 'A', 'R', 'D'.

8: OUTPUT — SUBROUTINE, supplied by the user. *External Procedure*

OUTPUT permits access to intermediate values of the computed solution (for example to print or plot them), at successive user-specified points. It is initially called by D02BJF with XSOL = X (the initial value of x). The user must reset XSOL to the next point (between the current XSOL and XEND) where OUTPUT is to be called, and so on at each call to OUTPUT. If, after a call to OUTPUT, the reset point XSOL is beyond XEND, D02BJF will integrate to XEND with no further calls to OUTPUT; if a call to OUTPUT is required at the point XSOL = XEND, then XSOL must be given precisely the value XEND.

Its specification is:

```
SUBROUTINE OUTPUT(XSOL, Y)
real               XSOL, Y(*)
```

1: XSOL — ***real*** *Input/Output*

On entry: the output value of the independent variable x.

On exit: the user must set XSOL to the next value of x at which OUTPUT is to be called.

2: Y($*$) — ***real*** array *Input*

On entry: the computed solution at the point XSOL.

If the user does not wish to access intermediate output, the actual argument OUTPUT **must** be the dummy routine D02BJX. (D02BJX is included in the NAG Fortran Library and so need not be supplied by the user. The name may be implementation-dependent: see the Users' Note for your implementation for details.)

OUTPUT must be declared as EXTERNAL in the (sub)program from which D02BJF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

9: G — ***real*** FUNCTION, supplied by the user. *External Procedure*

G must evaluate the function $g(x, y)$ for specified values x, y. It specifies the function g for which the first position x where $g(x, y) = 0$ is to be found.

Its specification is:

```
real FUNCTION G(X, Y)
real          X, Y(*)
```

1: X — ***real*** *Input*

On entry: the value of the independent variable x.

2: Y(∗) — ***real*** array *Input*

On entry: the value of the variable y_i, for $i = 1, 2, \ldots, n$.

If the user does not require the root finding option, the actual argument G **must** be the dummy routine D02BJW. (D02BJW is included in the NAG Fortran Library and so need not be supplied by the user. The name may be implementation-dependent: see the Users' Note for your implementation for details.)
G must be declared as EXTERNAL in the (sub)program from which D02BJF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

10: W(20∗N) — ***real*** array *Workspace*

11: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

6 Errors 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 detected by the routine:

IFAIL = 1

On entry, TOL $\geq$ 0.01,
or TOL is too small
or N $\leq$ 0,
or RELABS $\neq$ 'M', 'A', 'R' or 'D',
or X = XEND.

IFAIL = 2

With the given value of TOL, no further progress can be made across the integration range from the current point x = X. (See Section 8 for a discussion of this error exit.) The components Y(1),Y(2),...,Y(N) contain the computed values of the solution at the current point x = X. If the user has supplied g, then no point at which $g(x, y)$ changes sign has been located up to the point x = X.

IFAIL = 3

TOL is too small for D02BJF to take an initial step. X and Y(1),Y(2),...,Y(N) retain their initial values.

IFAIL = 4

XSOL has not been reset or XSOL lies behind X in the direction of integration, after the initial call to OUTPUT, if the OUTPUT option was selected.

IFAIL = 5

A value of XSOL returned by OUTPUT has not been reset or lies behind the last value of XSOL in the direction of integration, if the OUTPUT option was selected.

IFAIL = 6

At no point in the range X to XEND did the function $g(x, y)$ change sign, if g was supplied. It is assumed that $g(x, y) = 0$ has no solution.

IFAIL = 7

A serious error has occurred in an internal call to an interpolation routine. Check all subroutine calls and array dimensions. Seek expert help.

7 Accuracy

The accuracy of the computation of the solution vector Y may be controlled by varying the local error tolerance TOL. In general, a decrease in local error tolerance should lead to an increase in accuracy. Users are advised to choose RELABS = 'D' unless they have a good reason for a different choice.

If the problem is a root-finding one, then the accuracy of the root determined will depend on the properties of $g(x, y)$ and on the values of TOL and RELABS. The user should try to code G without introducing any unnecessary cancellation errors.

8 Further Comments

If more than one root is required, then to determine the second and later roots D02BJF may be called again starting a short distance past the previously determined roots. Alternatively the user may construct his own root finding code using D02PDF, D02PXF and C05AZF.

If the routine fails with IFAIL = 3, then it can be called again with a larger value of TOL if this has not already been tried. If the accuracy requested is really needed and cannot be obtained with this routine, the system may be very stiff (see below) or so badly scaled that it cannot be solved to the required accuracy.

If the routine fails with IFAIL = 2, it is probable that it has been called with a value of TOL which is so small that a solution cannot be obtained on the range X to XEND. This can happen for well-behaved systems and very small values of TOL. The user should, however, consider whether there is a more fundamental difficulty. For example:

(a) in the region of a singularity (infinite value) of the solution, the routine will usually stop with IFAIL = 2, unless overflow occurs first. Numerical integration cannot be continued through a singularity, and analytic treatment should be considered;

(b) for 'stiff' equations where the solution contains rapidly decaying components, the routine will use very small steps in x (internally to D02BJF) to preserve stability. This will exhibit itself by making the computing time excessively long, or occasionally by an exit with IFAIL = 2. Runge–Kutta methods are not efficient in such cases, and the user should try D02EJF.

9 Example

We illustrate the solution of four different problems. In each case the differential system (for a projectile) is

$$\begin{aligned} y' &= \tan\phi \\ v' &= \frac{-0.032\tan\phi}{v} - \frac{0.02v}{\cos\phi} \\ \phi' &= \frac{-0.032}{v^2} \end{aligned}$$

over an interval X = 0.0 to XEND = 10.0 starting with values $y = 0.5$, $v = 0.5$ and $\phi = \pi/5$. We solve each of the following problems with local error tolerances 1.0E−4 and 1.0E−5.

(i) To integrate to $x = 10.0$ producing intermediate output at intervals of 2.0 until a root is encountered where $y = 0.0$.
(ii) As (i) but with no intermediate output.
(iii) As (i) but with no termination on a root-finding condition.
(iv) As (i) but with no intermediate output and no root-finding termination condition.

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     D02BJF Example Program Text
*     Mark 18 Release. NAG Copyright 1997.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          N, IW
      PARAMETER        (N=3,IW=20*N)
*     .. Scalars in Common ..
      real             H, XEND
      INTEGER          K
*     .. Local Scalars ..
      real             PI, TOL, X
      INTEGER          I, IFAIL, J
*     .. Local Arrays ..
      real             W(IW), Y(N)
*     .. External Functions ..
      real             D02BJW, G, X01AAF
      EXTERNAL         D02BJW, G, X01AAF
*     .. External Subroutines ..
      EXTERNAL         D02BJF, D02BJX, FCN, OUT
*     .. Intrinsic Functions ..
      INTRINSIC        real
*     .. Common blocks ..
      COMMON           XEND, H, K
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D02BJF Example Program Results'
      XEND = 10.0e0
      PI = X01AAF(0.0e0)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Case 1: intermediate output, root-finding'
      DO 20 J = 4, 5
         TOL = 10.0e0**(-J)
         WRITE (NOUT,*)
         WRITE (NOUT,99999) ' Calculation with TOL =', TOL
         X = 0.0e0
         Y(1) = 0.5e0
         Y(2) = 0.5e0
         Y(3) = PI/5.0e0
         K = 4
         H = (XEND-X)/real(K+1)
         WRITE (NOUT,*) '     X         Y(1)         Y(2)         Y(3)'
         IFAIL = 0
*
         CALL D02BJF(X,XEND,N,Y,FCN,TOL,'Default',OUT,G,W,IFAIL)
*
         WRITE (NOUT,99998) '  Root of Y(1) = 0.0 at', X
         WRITE (NOUT,99997) '  Solution is', (Y(I),I=1,N)
```

```
   20 CONTINUE
      WRITE (NOUT,*)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Case 2: no intermediate output, root-finding'
      DO 40 J = 4, 5
         TOL = 10.0e0**(-J)
         WRITE (NOUT,*)
         WRITE (NOUT,99999) ' Calculation with TOL =', TOL
         X = 0.0e0
         Y(1) = 0.5e0
         Y(2) = 0.5e0
         Y(3) = PI/5.0e0
         IFAIL = 0
*
         CALL D02BJF(X,XEND,N,Y,FCN,TOL,'Default',D02BJX,G,W,IFAIL)
*
         WRITE (NOUT,99998) '  Root of Y(1) = 0.0 at', X
         WRITE (NOUT,99997) '  Solution is', (Y(I),I=1,N)
   40 CONTINUE
      WRITE (NOUT,*)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Case 3: intermediate output, no root-finding'
      DO 60 J = 4, 5
         TOL = 10.0e0**(-J)
         WRITE (NOUT,*)
         WRITE (NOUT,99999) ' Calculation with TOL =', TOL
         X = 0.0e0
         Y(1) = 0.5e0
         Y(2) = 0.5e0
         Y(3) = PI/5.0e0
         K = 4
         H = (XEND-X)/real(K+1)
         WRITE (NOUT,*) '     X         Y(1)         Y(2)         Y(3)'
         IFAIL = 0
*
         CALL D02BJF(X,XEND,N,Y,FCN,TOL,'Default',OUT,D02BJW,W,IFAIL)
*
   60 CONTINUE
      WRITE (NOUT,*)
      WRITE (NOUT,*)
      WRITE (NOUT,*)
     +'Case 4: no intermediate output, no root-finding ( integrate to XE
     +ND)'
      DO 80 J = 4, 5
         TOL = 10.0e0**(-J)
         WRITE (NOUT,*)
         WRITE (NOUT,99999) ' Calculation with TOL =', TOL
         X = 0.0e0
         Y(1) = 0.5e0
         Y(2) = 0.5e0
         Y(3) = PI/5.0e0
         WRITE (NOUT,*) '     X         Y(1)         Y(2)         Y(3)'
         WRITE (NOUT,99996) X, (Y(I),I=1,N)
         IFAIL = 0
*
         CALL D02BJF(X,XEND,N,Y,FCN,TOL,'Default',D02BJX,D02BJW,W,IFAIL)
*
         WRITE (NOUT,99996) X, (Y(I),I=1,N)
   80 CONTINUE
      STOP
*
99999 FORMAT (1X,A,e8.1)
99998 FORMAT (1X,A,F7.3)
99997 FORMAT (1X,A,3F13.4)
99996 FORMAT (1X,F8.2,3F13.4)
      END
*
      SUBROUTINE OUT(X,Y)
*     .. Parameters ..
      INTEGER         NOUT
      PARAMETER       (NOUT=6)
      INTEGER         N
      PARAMETER       (N=3)
*     .. Scalar Arguments ..
      real            X
*     .. Array Arguments ..
      real            Y(N)
*     .. Scalars in Common ..
      real            H, XEND
      INTEGER         I
*     .. Local Scalars ..
      INTEGER         J
*     .. Intrinsic Functions ..
      INTRINSIC       real
*     .. Common blocks ..
      COMMON          XEND, H, I
*     .. Executable Statements ..
      WRITE (NOUT,99999) X, (Y(J),J=1,N)
      X = XEND - real(I)*H
      I = I - 1
      RETURN
*
99999 FORMAT (1X,F8.2,3F13.4)
      END
*
      SUBROUTINE FCN(T,Y,F)
*     .. Parameters ..
      INTEGER         N
      PARAMETER       (N=3)
*     .. Scalar Arguments ..
      real            T
*     .. Array Arguments ..
      real            F(N), Y(N)
*     .. Intrinsic Functions ..
      INTRINSIC       COS, TAN
*     .. Executable Statements ..
      F(1) = TAN(Y(3))
      F(2) = -0.032e0*TAN(Y(3))/Y(2) - 0.02e0*Y(2)/COS(Y(3))
      F(3) = -0.032e0/Y(2)**2
      RETURN
      END
*
```

```
      real FUNCTION G(T,Y)
*     .. Parameters ..
      INTEGER         N
      PARAMETER       (N=3)
*     .. Scalar Arguments ..
      real            T
*     .. Array Arguments ..
      real            Y(N)
*     .. Executable Statements ..
      G = Y(1)
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 D02BJF Example Program Results

 Case 1: intermediate output, root-finding

  Calculation with TOL = 0.1E-03
      X         Y(1)         Y(2)         Y(3)
    0.00       0.5000       0.5000       0.6283
    2.00       1.5493       0.4055       0.3066
    4.00       1.7423       0.3743      -0.1289
    6.00       1.0055       0.4173      -0.5507
   Root of Y(1) = 0.0 at  7.288
   Solution is       0.0000       0.4749      -0.7601

  Calculation with TOL = 0.1E-04
      X         Y(1)         Y(2)         Y(3)
    0.00       0.5000       0.5000       0.6283
    2.00       1.5493       0.4055       0.3066
    4.00       1.7423       0.3743      -0.1289
    6.00       1.0055       0.4173      -0.5507
   Root of Y(1) = 0.0 at  7.288
   Solution is       0.0000       0.4749      -0.7601


 Case 2: no intermediate output, root-finding

  Calculation with TOL = 0.1E-03
   Root of Y(1) = 0.0 at  7.288
   Solution is       0.0000       0.4749      -0.7601

  Calculation with TOL = 0.1E-04
   Root of Y(1) = 0.0 at  7.288
   Solution is       0.0000       0.4749      -0.7601
```

```
Case 3: intermediate output, no root-finding

 Calculation with TOL = 0.1E-03
     X         Y(1)         Y(2)         Y(3)
    0.00       0.5000       0.5000       0.6283
    2.00       1.5493       0.4055       0.3066
    4.00       1.7423       0.3743      -0.1289
    6.00       1.0055       0.4173      -0.5507
    8.00      -0.7460       0.5130      -0.8537
   10.00      -3.6283       0.6333      -1.0515

 Calculation with TOL = 0.1E-04
     X         Y(1)         Y(2)         Y(3)
    0.00       0.5000       0.5000       0.6283
    2.00       1.5493       0.4055       0.3066
    4.00       1.7423       0.3743      -0.1289
    6.00       1.0055       0.4173      -0.5507
    8.00      -0.7459       0.5130      -0.8537
   10.00      -3.6282       0.6333      -1.0515


Case 4: no intermediate output, no root-finding ( integrate to XEND)

 Calculation with TOL = 0.1E-03
     X         Y(1)         Y(2)         Y(3)
    0.00       0.5000       0.5000       0.6283
   10.00      -3.6283       0.6333      -1.0515

 Calculation with TOL = 0.1E-04
     X         Y(1)         Y(2)         Y(3)
    0.00       0.5000       0.5000       0.6283
   10.00      -3.6282       0.6333      -1.0515
```