NAG Fortran Library Routine Document

D02GAF

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

D02GAF solves the two-point boundary-value problem with assigned boundary-values for a system of ordinary differential equations, using a deferred correction technique and a Newton iteration.

2 Specification

```
      SUBROUTINE D02GAF (U, V, N, A, B, TOL, FCN, MNP, X, Y, NP, W, LW, IW,
     1                   LIW, IFAIL)
      INTEGER            N, MNP, NP, LW, IW(LIW), LIW, IFAIL
      double precision   U(N,2), V(N,2), A, B, TOL, X(MNP), Y(N,MNP), W(LW)
      EXTERNAL           FCN
```

3 Description

D02GAF solves a two-point boundary-value problem for a system of n differential equations in the interval $[a, b]$. The system is written in the form

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

and the derivatives are evaluated by a (sub)program FCN supplied by you. Initially, n boundary-values of the variables y_i must be specified (assigned), some at a and some at b. You also supply estimates of the remaining n boundary-values and all the boundary-values are used in constructing an initial approximation to the solution. This approximate solution is corrected by a finite-difference technique with deferred correction allied with a Newton iteration to solve the finite-difference equations. The technique used is described fully in Pereyra (1979). The Newton iteration requires a Jacobian matrix $\frac{\partial f_i}{\partial y_j}$ and this is calculated by numerical differentiation using an algorithm described in Curtis *et al.* (1974).

You supply an absolute error tolerance and may also supply an initial mesh for the construction of the finite-difference equations (alternatively a default mesh is used). The algorithm constructs a solution on a mesh defined by adding points to the initial mesh. This solution is chosen so that the error is everywhere less than your tolerance and so that the error is approximately equidistributed on the final mesh. The solution is returned on this final mesh.

If the solution is required at a few specific points then these should be included in the initial mesh. If on the other hand the solution is required at several specific points then you should use the interpolation routines provided in Chapter E01 if these points do not themselves form a convenient mesh.

4 References

Curtis A R, Powell M J D and Reid J K (1974) On the estimation of sparse Jacobian matrices *J. Inst. Maths. Applics.* **13** 117–119

Pereyra V (1979) PASVA3: An adaptive finite-difference Fortran program for first order nonlinear, ordinary boundary problems *Codes for Boundary Value Problems in Ordinary Differential Equations. Lecture Notes in Computer Science* (ed B Childs, M Scott, J W Daniel, E Denman and P Nelson) **76** Springer–Verlag

5 Parameters

1: U(N,2) – ***double precision*** array *Input*

On entry: $\mathrm{U}(i,1)$ must be set to the known (assigned) or estimated values of y_i at a and $\mathrm{U}(i,2)$ must be set to the known or estimated values of y_i at b, for $i = 1, 2, \ldots, n$.

2: V(N,2) – ***double precision*** array *Input*

On entry: $\mathrm{V}(i,j)$ must be set to 0.0 if $\mathrm{U}(i,j)$ is a known (assigned) value and to 1.0 if $\mathrm{U}(i,j)$ is an estimated value, for $i = 1, 2, \ldots, n$; $j = 1, 2$.

Constraint: precisely N of the $\mathrm{V}(i,j)$ must be set to 0.0, i.e., precisely N of the $\mathrm{U}(i,j)$ must be known values, and these must not be all at a or all at b.

3: N – INTEGER *Input*

On entry: the number of equations.

Constraint: $\mathrm{N} \geq 2$.

4: A – ***double precision*** *Input*

On entry: a, the left-hand boundary point.

5: B – ***double precision*** *Input*

On entry: b, the right-hand boundary point.

Constraint: $\mathrm{B} > \mathrm{A}$.

6: TOL – ***double precision*** *Input*

On entry: a positive absolute error tolerance. If

$$a = x_1 < x_2 < \cdots < x_{\mathrm{NP}} = b$$

is the final mesh, $z_j(x_i)$ is the jth component of the approximate solution at x_i, and $y_j(x)$ is the jth component of the true solution of equation (1) (see Section 3) and the boundary conditions, then, except in extreme cases, it is expected that

$$\left|z_j(x_i) - y_j(x_i)\right| \leq \mathrm{TOL}, \quad i = 1, 2, \ldots, \mathrm{NP}; j = 1, 2, \ldots, n. \tag{2}$$

Constraint: $\mathrm{TOL} > 0.0$.

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

FCN must evaluate the functions f_i (i.e., the derivatives y_i') at the general point x.

Its specification is:

```
SUBROUTINE FCN (X, Y, F)
double precision    X, Y(n), F(n)
```

where n is the actual value of N in the call of D02GAF.

1: X – ***double precision*** *Input*

On entry: the value of the argument x.

2: Y(n) – ***double precision*** array *Input*

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

3: F(n) – ***double precision*** array *Output*

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

FCN must be declared as EXTERNAL in the (sub)program from which D02GAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

8: MNP – INTEGER *Input*

On entry: the maximum permitted number of mesh points.

Constraint: $\text{MNP} \geq 32$.

9: X(MNP) – ***double precision*** array *Input/Output*

On entry: if $\text{NP} \geq 4$ (see NP), the first NP elements must define an initial mesh. Otherwise the elements of X need not be set.

Constraint:

$$\text{A} = \text{X}(1) < \text{X}(2) < \cdots < \text{X}(\text{NP}) = \text{B}, \quad \text{NP} \geq 4. \tag{3}$$

On exit: $\text{X}(1), \text{X}(2), \ldots, \text{X}(\text{NP})$ define the final mesh (with the returned value of NP) satisfying the relation (3).

10: Y(N,MNP) – ***double precision*** array *Output*

On exit: the approximate solution $z_j(x_i)$ satisfying (2), on the final mesh, that is

$$\text{Y}(j,i) = z_j(x_i), \quad i = 1,2,\ldots,\text{NP}; j = 1,2,\ldots,n,$$

where NP is the number of points in the final mesh.

The remaining columns of Y are not used.

11: NP – INTEGER *Input/Output*

On entry: determines whether a default or user-supplied mesh is used.

$\text{NP} = 0$

A default value of 4 for NP and a corresponding equispaced mesh $\text{X}(1), \text{X}(2), \ldots, \text{X}(\text{NP})$ are used.

$\text{NP} \geq 4$

You must define an initial mesh using the array X as described.

Constraint: $\text{NP} = 0$ or $4 \leq \text{NP} \leq \text{MNP}$.

On exit: the number of points in the final (returned) mesh.

12: W(LW) – ***double precision*** array *Workspace*
13: LW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which D02GAF is called.

Constraint: $\text{LW} \geq \text{MNP} \times \left(3\text{N}^2 + 6\text{N} + 2\right) + 4\text{N}^2 + 4\text{N}$.

14: IW(LIW) – INTEGER array *Workspace*
15: LIW – INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which D02GAF is called.

Constraint: $\text{LIW} \geq \text{MNP} \times (2\text{N} + 1) + \text{N}^2 + 4\text{N} + 2$.

16: IFAIL – INTEGER *Input/Output*

For this routine, the normal use of IFAIL is extended to control the printing of error and warning messages as well as specifying hard or soft failure (see Chapter P01).

On entry: IFAIL must be set to a value with the decimal expansion cba, where each of the decimal digits c, b and a must have a value of 0 or 1.

$a = 0$ specifies hard failure, otherwise soft failure;

$b = 0$ suppresses error messages, otherwise error messages will be printed (see Section 6);

$c = 0$ suppresses warning messages, otherwise warning messages will be printed (see Section 6).

The recommended value for inexperienced users is 110 (i.e., hard failure with all messages printed).

On exit: IFAIL $= 0$ unless the routine detects an error (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 $= 1$

One or more of the parameters N, TOL, NP, MNP, LW or LIW has been incorrectly set, or B $\leq$ A, or the condition (3) on X is not satisfied, or the number of known boundary-values (specified by V) is not N.

IFAIL $= 2$

The Newton iteration has failed to converge. This could be due to there being too few points in the initial mesh or to the initial approximate solution being too inaccurate. If this latter reason is suspected you should use (sub)program D02RAF instead. If the warning 'Jacobian matrix is singular' is printed this could be due to specifying zero estimated boundary-values and these should be varied. This warning could also be printed in the unlikely event of the Jacobian matrix being calculated inaccurately. If you cannot make changes to prevent the warning then (sub)program D02RAF should be used.

IFAIL $= 3$

The Newton iteration has reached round-off level. It could be, however, that the answer returned is satisfactory. This error might occur if too much accuracy is requested.

IFAIL $= 4$

A finer mesh is required for the accuracy requested; that is MNP is not large enough.

IFAIL $= 5$

A serious error has occurred in a call to D02GAF. Check all array subscripts and (sub)program parameter lists in calls to D02GAF. Seek expert help.

7 Accuracy

The solution returned by the routine will be accurate to your tolerance as defined by the relation (2) except in extreme circumstances. If too many points are specified in the initial mesh, the solution may be more accurate than requested and the error may not be approximately equidistributed.

8 Further Comments

The time taken by D02GAF depends on the difficulty of the problem, the number of mesh points used (and the number of different meshes used), the number of Newton iterations and the number of deferred corrections.

You are strongly recommended to set IFAIL to obtain self-explanatory error messages, and also monitoring information about the course of the computation. You may select the channel numbers on which this output is to appear by calls of X04AAF (for error messages) or X04ABF (for monitoring information) – see Section 9 for an example. Otherwise the default channel numbers will be used, as specified in the Users' Note.

A common cause of convergence problems in the Newton iteration is that you have specified too few points in the initial mesh. Although the routine adds points to the mesh to improve accuracy it is unable to do so until the solution on the initial mesh has been calculated in the Newton iteration.

If you specify zero known **and** estimated boundary-values, the routine constructs a zero initial approximation and in many cases the Jacobian is singular when evaluated for this approximation, leading to the breakdown of the Newton iteration.

You may be unable to provide a sufficiently good choice of initial mesh and estimated boundary-values, and hence the Newton iteration may never converge. In this case the continuation facility provided in D02RAF is recommended.

In the case where you wish to solve a sequence of similar problems, the final mesh from solving one case is strongly recommended as the initial mesh for the next.

9 Example

We solve the differential equation

$$y''' = -yy'' - \beta\left(1 - y'^2\right)$$

with boundary conditions

$$y(0) = y'(0) = 0, \quad y'(10) = 1$$

for $\beta = 0.0$ and $\beta = 0.2$ to an accuracy specified by $\mathrm{TOL} = 1.0\mathrm{D} - 3$. We solve first the simpler problem with $\beta = 0.0$ using an equispaced mesh of 26 points and then we solve the problem with $\beta = 0.2$ using the final mesh from the first problem.

Note the call to X04ABF prior to the call to D02GAF.

9.1 Program Text

```
*     D02GAF Example Program Text
*     Mark 14 Revised. NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          N, MNP, LW, LIW
      PARAMETER        (N=3,MNP=40,LW=MNP*(3*N*N+6*N+2)+4*N*N+4*N,
     +                 LIW=MNP*(2*N+1)+N*N+4*N+2)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Scalars in Common ..
      DOUBLE PRECISION BETA
*     .. Local Scalars ..
      DOUBLE PRECISION A, B, TOL
      INTEGER          I, IFAIL, J, K, NP
*     .. Local Arrays ..
      DOUBLE PRECISION U(N,2), V(N,2), W(LW), X(MNP), Y(N,MNP)
      INTEGER          IW(LIW)
*     .. External Subroutines ..
      EXTERNAL         D02GAF, FCN, X04ABF
*     .. Intrinsic Functions ..
      INTRINSIC        DBLE
*     .. Common blocks ..
      COMMON           BETA
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D02GAF Example Program Results'
      TOL = 1.0D-3
      NP = 26
      A = 0.0D0
      B = 10.0D0
      CALL X04ABF(1,NOUT)
      BETA = 0.0D0
```

```
      DO 40 I = 1, N
         DO 20 J = 1, 2
            U(I,J) = 0.0D0
            V(I,J) = 0.0D0
   20    CONTINUE
   40 CONTINUE
      V(3,1) = 1.0D0
      V(1,2) = 1.0D0
      V(3,2) = 1.0D0
      U(2,2) = 1.0D0
      U(1,2) = B
      X(1) = A
      DO 60 I = 2, NP - 1
         X(I) = (DBLE(NP-I)*A+DBLE(I-1)*B)/DBLE(NP-1)
   60 CONTINUE
      X(NP) = B
      DO 80 K = 1, 2
         WRITE (NOUT,*)
         WRITE (NOUT,99999) 'Problem with BETA = ', BETA
*        * Set IFAIL to 111 to obtain monitoring information *
         IFAIL = 11
*
         CALL D02GAF(U,V,N,A,B,TOL,FCN,MNP,X,Y,NP,W,LW,IW,LIW,IFAIL)
*
         IF (IFAIL.EQ.0 .OR. IFAIL.EQ.3) THEN
            WRITE (NOUT,*)
            IF (IFAIL.EQ.3) WRITE (NOUT,99996) ' IFAIL = ', IFAIL
            WRITE (NOUT,99998) 'Solution on final mesh of ', NP,
     +        ' points'
            WRITE (NOUT,*)
     +        '      X(I)        Y1(I)        Y2(I)        Y3(I)'
            WRITE (NOUT,99997) (X(I),(Y(J,I),J=1,N),I=1,NP)
            BETA = BETA + 0.2D0
         ELSE
            STOP
         END IF
   80 CONTINUE
      STOP
*
99999 FORMAT (1X,A,F7.2)
99998 FORMAT (1X,A,I2,A)
99997 FORMAT (1X,F11.3,3F13.4)
99996 FORMAT (1X,A,I3)
      END
*
      SUBROUTINE FCN(X,Y,F)
*     .. Parameters ..
      INTEGER          N
      PARAMETER        (N=3)
*     .. Scalar Arguments ..
      DOUBLE PRECISION X
*     .. Array Arguments ..
      DOUBLE PRECISION F(N), Y(N)
*     .. Scalars in Common ..
      DOUBLE PRECISION BETA
*     .. Common blocks ..
      COMMON           BETA
*     .. Executable Statements ..
      F(1) = Y(2)
      F(2) = Y(3)
      F(3) = -Y(1)*Y(3) - BETA*(1.0D0-Y(2)*Y(2))
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 D02GAF Example Program Results

 Problem with BETA =    0.00

 Solution on final mesh of 26 points
        X(I)        Y1(I)        Y2(I)         Y3(I)
       0.000       0.0000       0.0000        0.4695
       0.400       0.0375       0.1876        0.4673
       0.800       0.1497       0.3719        0.4511
       1.200       0.3336       0.5450        0.4104
       1.600       0.5828       0.6963        0.3424
       2.000       0.8864       0.8163        0.2558
       2.400       1.2309       0.9009        0.1678
       2.800       1.6026       0.9529        0.0953
       3.200       1.9900       0.9805        0.0464
       3.600       2.3851       0.9930        0.0193
       4.000       2.7834       0.9978        0.0069
       4.400       3.1829       0.9994        0.0021
       4.800       3.5828       0.9999        0.0006
       5.200       3.9828       1.0000        0.0001
       5.600       4.3828       1.0000        0.0000
       6.000       4.7828       1.0000        0.0000
       6.400       5.1828       1.0000        0.0000
       6.800       5.5828       1.0000        0.0000
       7.200       5.9828       1.0000       -0.0000
       7.600       6.3828       1.0000        0.0000
       8.000       6.7828       1.0000       -0.0000
       8.400       7.1828       1.0000        0.0000
       8.800       7.5828       1.0000       -0.0000
       9.200       7.9828       1.0000        0.0000
       9.600       8.3828       1.0000       -0.0000
      10.000       8.7828       1.0000       -0.0000

 Problem with BETA =    0.20

 Solution on final mesh of 26 points
        X(I)        Y1(I)        Y2(I)         Y3(I)
       0.000       0.0000       0.0000        0.6865
       0.400       0.0528       0.2584        0.6040
       0.800       0.2020       0.4814        0.5091
       1.200       0.4324       0.6636        0.4001
       1.600       0.7268       0.8007        0.2860
       2.000       1.0670       0.8939        0.1821
       2.400       1.4368       0.9498        0.1017
       2.800       1.8233       0.9791        0.0492
       3.200       2.2180       0.9924        0.0206
       3.600       2.6162       0.9976        0.0074
       4.000       3.0157       0.9993        0.0023
       4.400       3.4156       0.9998        0.0006
       4.800       3.8155       1.0000        0.0001
       5.200       4.2155       1.0000        0.0000
       5.600       4.6155       1.0000        0.0000
       6.000       5.0155       1.0000        0.0000
       6.400       5.4155       1.0000       -0.0000
       6.800       5.8155       1.0000       -0.0000
       7.200       6.2155       1.0000       -0.0000
       7.600       6.6155       1.0000       -0.0000
       8.000       7.0155       1.0000       -0.0000
       8.400       7.4155       1.0000       -0.0000
       8.800       7.8155       1.0000       -0.0000
       9.200       8.2155       1.0000       -0.0000
       9.600       8.6155       1.0000       -0.0000
      10.000       9.0155       1.0000       -0.0000
```