NAG Library Routine Document

D02GBF

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

D02GBF solves a general linear two-point boundary value problem for a system of ordinary differential equations, using a deferred correction technique.

2   Specification

```
SUBROUTINE D02GBF (A, B, N, TOL, FCNF, FCNG, C, D, GAM, MNP, X, Y, NP, W,    &
                   LW, IW, LIW, IFAIL)

INTEGER            N, MNP, NP, LW, IW(LIW), LIW, IFAIL
REAL (KIND=nag_wp) A, B, TOL, C(N,N), D(N,N), GAM(N), X(MNP), Y(N,MNP),    &
                   W(LW)
EXTERNAL           FCNF, FCNG
```

3   Description

D02GBF solves a linear two-point boundary value problem for a system of n ordinary differential equations in the interval $[a, b]$. The system is written in the form

$$y' = F(x)y + g(x) \tag{1}$$

and the boundary conditions are written in the form

$$Cy(a) + Dy(b) = \gamma. \tag{2}$$

Here $F(x)$, C and D are n by n matrices, and $g(x)$ and γ are n-component vectors. The approximate solution to (1) and (2) is found using a finite difference method with deferred correction. The algorithm is a specialization of that used in subroutine D02RAF which solves a nonlinear version of (1) and (2). The nonlinear version of the algorithm is described fully in Pereyra (1979).

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

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* (eds B Childs, M Scott, J W Daniel, E Denman and P Nelson) **76** Springer−Verlag

5   Arguments

1:    A – REAL (KIND=nag_wp)   *Input*

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

2:　　B – REAL (KIND=nag_wp)　　　　　　　　　　　　　　　　　　　　　　　*Input*

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

Constraint: $B > A$.

3:　　N – INTEGER　　　　　　　　　　　　　　　　　　　　　　　　　　　*Input*

On entry: the number of equations; that is n is the order of system (1).

Constraint: $N \geq 2$.

4:　　TOL – REAL (KIND=nag_wp)　　　　　　　　　　　　　　　　　　　*Input*

On entry: a positive absolute error tolerance. If

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

is the final mesh, $z(x)$ is the approximate solution from D02GBF and $y(x)$ is the true solution of equations (1) and (2) then, except in extreme cases, it is expected that

$$\|z - y\| \leq \mathrm{TOL} \tag{3}$$

where

$$\|u\| = \max_{1 \leq i \leq N} \max_{1 \leq j \leq \mathrm{NP}} \left| u_i\left(x_j\right) \right|.$$

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

5:　　FCNF – SUBROUTINE, supplied by the user.　　　　　　　*External Procedure*

FCNF must evaluate the matrix $F(x)$ in (1) at a general point x.

The specification of FCNF is:

```
SUBROUTINE FCNF (X, F)
REAL (KIND=nag_wp) X, F(*)
```

In the description of the arguments of D02GBF below, n denotes the actual value of N in the call of D02GBF.

1:　　X – REAL (KIND=nag_wp)　　　　　　　　　　　　　　　　*Input*

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

2:　　F($*$) – REAL (KIND=nag_wp) array　　　　　　　　　*Output*

On exit: $\mathrm{F}(n \times (i-1) + j)$ must contain the (i,j)th element of the matrix $F(x)$, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$. (See Section 10 for an example.)

FCNF must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which D02GBF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

6:　　FCNG – SUBROUTINE, supplied by the user.　　　　　　　*External Procedure*

FCNG must evaluate the vector $g(x)$ in (1) at a general point x.

The specification of FCNG is:

```
SUBROUTINE FCNG (X, G)
REAL (KIND=nag_wp) X, G(*)
```

In the description of the arguments of D02GBF below, n denotes the actual value of N in the call of D02GBF.

1:	X – REAL (KIND=nag_wp)	*Input*

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

2:	G($*$) – REAL (KIND=nag_wp) array	*Output*

On exit: the ith element of the vector $g(x)$, for $i = 1, 2, \ldots, n$. (See Section 10 for an example.)

FCNG must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which D02GBF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

7:	C(N, N) – REAL (KIND=nag_wp) array	*Input/Output*
8:	D(N, N) – REAL (KIND=nag_wp) array	*Input/Output*
9:	GAM(N) – REAL (KIND=nag_wp) array	*Input/Output*

On entry: the arrays C and D must be set to the matrices C and D in (2)). GAM must be set to the vector γ in (2).

On exit: the rows of C and D and the components of GAM are reordered so that the boundary conditions are in the order:

(i) conditions on $y(a)$ only;

(ii) condition involving $y(a)$ and $y(b)$; and

(iii) conditions on $y(b)$ only.

The routine will be slightly more efficient if the arrays C, D and GAM are ordered in this way before entry, and in this event they will be unchanged on exit.

Note that the problems (1) and (2) must be of boundary value type, that is neither C nor D may be identically zero. Note also that the rank of the matrix $[C, D]$ must be n for the problem to be properly posed. Any violation of these conditions will lead to an error exit.

10:	MNP – INTEGER	*Input*

On entry: the maximum permitted number of mesh points.

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

11:	X(MNP) – REAL (KIND=nag_wp) 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:

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

On exit: $X(1), X(2), \ldots, X(NP)$ define the final mesh (with the returned value of NP) satisfying the relation (4).

12:	Y(N, MNP) – REAL (KIND=nag_wp) array	*Output*

On exit: the approximate solution $z(x)$ satisfying (3), on the final mesh, that is

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

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

The remaining columns of Y are not used.

13:    NP – INTEGER   *Input/Output*

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

   NP = 0
         A default value of 4 for NP and a corresponding equispaced mesh $X(1), X(2), \ldots, X(NP)$
         are used.

   NP ≥ 4
         You must define an initial mesh X as in (4).

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

14:    W(LW) – REAL (KIND=nag_wp) array   *Workspace*
15:    LW – INTEGER   *Input*

   On entry: the dimension of the array W as declared in the (sub)program from which D02GBF is
   called.

   Constraint: $\text{LW} \geq \text{MNP} \times \left(3N^2 + 5N + 2\right) + 3N^2 + 5N$.

16:    IW(LIW) – INTEGER array   *Workspace*
17:    LIW – INTEGER   *Input*

   On entry: the dimension of the array IW as declared in the (sub)program from which D02GBF is
   called.

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

18:    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 Section 3.4 in How to Use the NAG
   Library and its Documentation).

   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 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 = 1

   One or more of the arguments N, TOL, NP, MNP, LW or LIW is incorrectly set, B ≤ A or the
   condition (4) on X is not satisfied.

IFAIL = 2

   There are three possible reasons for this error exit to be taken:

(i)   one of the matrices C or D is identically zero (that is, the problem is of initial value and not boundary value type). In this case, $\text{IW}(1) = 0$ on exit;

(ii) a row of C and the corresponding row of D are identically zero (that is, the boundary conditions are rank deficient). In this case, on exit $\text{IW}(1)$ contains the index of the first such row encountered; and

(iii) more than n of the columns of the n by $2n$ matrix $[C, D]$ are identically zero (that is, the boundary conditions are rank deficient). In this case, on exit $\text{IW}(1)$ contains minus the number of non-identically zero columns.

IFAIL $= 3$

The routine has failed to find a solution to the specified accuracy. There are a variety of possible reasons including:

(i)   the boundary conditions are rank deficient, which may be indicated by the message that the Jacobian is singular. However this is an unlikely explanation for the error exit as all rank deficient boundary conditions should lead instead to error exits with either IFAIL $= 2$ or 5; see also (iv);

(ii) not enough mesh points are permitted in order to attain the required accuracy. This is indicated by NP $=$ MNP on return from a call to D02GBF. This difficulty may be aggravated by a poor initial choice of mesh points;

(iii) the accuracy requested cannot be attained on the computer being used; and

(iv) an unlikely combination of values of Fx has led to a singular Jacobian. The error should not persist if more mesh points are allowed.

IFAIL $= 4$

A serious error has occurred in a call to D02GBF. Check all array subscripts and subroutine argument lists in calls to D02GBF. Seek expert help.

IFAIL $= 5$

There are two possible reasons for this error exit which occurs when checking the rank of the boundary conditions by reduction to a row echelon form:

(i)   at least one row of the n by $2n$ matrix $[C, D]$ is a linear combination of the other rows and hence the boundary conditions are rank deficient. The index of the first such row encountered is given by $\text{IW}(1)$ on exit; and

(ii) as (i) but the rank deficiency implied by this error exit has only been determined up to a numerical tolerance. Minus the index of the first such row encountered is given by $\text{IW}(1)$ on exit.

In case (ii) there is some doubt as to the rank deficiency of the boundary conditions. However even if the boundary conditions are not rank deficient they are not posed in a suitable form for use with this routine.

For example, if

$$C = \begin{pmatrix} 1 & 0 \\ 1 & \epsilon \end{pmatrix}, \quad D = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \quad \gamma = \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix}$$

and ϵ is small enough, this error exit is likely to be taken. A better form for the boundary conditions in this case would be

$$C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad D = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad \gamma = \begin{pmatrix} \gamma_1 \\ \epsilon^{-1}(\gamma_2 - \gamma_1) \end{pmatrix}.$$

$\text{IFAIL} = -99$

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

> See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

$\text{IFAIL} = -399$

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

> See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

$\text{IFAIL} = -999$

> Dynamic memory allocation failed.

> See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7   Accuracy

The solution returned by the routine will be accurate to your tolerance as defined by the relation (3) 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   Parallelism and Performance

D02GBF is not thread safe and should not be called from a multithreaded user program. Please see Section 3.12.1 in How to Use the NAG Library and its Documentation for more information on thread safety.

D02GBF 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

The time taken by D02GBF depends on the difficulty of the problem, the number of mesh points (and meshes) used 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 unit numbers on which this output is to appear by calls of X04AAF (for error messages) or X04ABF (for monitoring information) – see Section 10 for an example. Otherwise the default unit numbers will be used, as specified in the Users' Note.

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.

10   Example

This example solves the problem (written as a first-order system)

$$\epsilon y'' + y' = 0$$

with boundary conditions

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

for the cases $\epsilon = 10^{-1}$ and $\epsilon = 10^{-2}$ using the default initial mesh in the first case, and the final mesh of

the first case as initial mesh for the second (more difficult) case. We give the solution and the error at each mesh point to illustrate the accuracy of the method given the accuracy request TOL = 1.0E−3.

Note the call to X04ABF prior to the call to D02GBF.

10.1 Program Text

```
!   D02GBF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module d02gbfe_mod

!     Data for D02GBF example program

!       .. Use Statements ..
      Use nag_library, Only: nag_wp
!       .. Implicit None Statement ..
      Implicit None
!       .. Accessibility Statements ..
      Private
      Public                                :: fcnf, fcng
!       .. Parameters ..
      Real (Kind=nag_wp), Parameter, Public :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter, Public :: zero = 0.0_nag_wp
      Integer, Parameter, Public            :: iset = 1, n = 2, nin = 5, nout = 6
!       .. Local Scalars ..
      Real (Kind=nag_wp), Public, Save :: eps
    Contains
      Subroutine fcnf(x,f)

!         .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: x
!         .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: f(*)
!         .. Executable Statements ..
        f(1:2) = 0.0E0_nag_wp
        f(3) = 1.0E0_nag_wp
        f(4) = -1.0E0_nag_wp/eps
        Return
      End Subroutine fcnf
      Subroutine fcng(x,g)

!         .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: x
!         .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: g(*)
!         .. Executable Statements ..
        g(1:2) = 0.0E0_nag_wp
        Return
      End Subroutine fcng
    End Module d02gbfe_mod
    Program d02gbfe

!     D02GBF Example Main Program

!       .. Use Statements ..
      Use nag_library, Only: d02gbf, nag_wp, x04abf
      Use d02gbfe_mod, Only: eps, fcnf, fcng, iset, n, nin, nout, one, zero
!       .. Implicit None Statement ..
      Implicit None
!       .. Local Scalars ..
      Real (Kind=nag_wp)                    :: a, b, tol
      Integer                               :: i, ifail, j, liw, lw, mnp, np,       &
                                               outchn
!       .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: c(:,:), d(:,:), gam(:), w(:), x(:),  &
                                          y(:,:)
      Integer, Allocatable             :: iw(:)
!       .. Executable Statements ..
      Write (nout,*) 'D02GBF Example Program Results'
```

```
!     Skip heading in data file
      Read (nin,*)
!     mnp: maximum permitted number of mesh points.
      Read (nin,*) mnp
      liw = mnp*(2*n+1) + n
      lw = mnp*(3*n*n+5*n+2) + 3*n*n + 5*n
      Allocate (iw(liw),c(n,n),d(n,n),gam(n),w(lw),x(mnp),y(n,mnp))
!     tol: positive absolute error tolerance
!     np : determines whether a default or user-supplied mesh is used.
!     a  : left-hand boundary point, b: right-hand boundary point.
      Read (nin,*) tol
      Read (nin,*) np
      Read (nin,*) a, b
      outchn = nout
      Call x04abf(iset,outchn)
      gam(1:n) = zero
      c(1:n,1:n) = zero
      d(1:n,1:n) = zero
      c(1,1) = one
      d(2,1) = one
      gam(2) = one
loop: Do i = 1, 2
        eps = 10.0E0_nag_wp**(-i)
        Write (nout,*)
!       ifail: behaviour on error exit
!             =1 for quiet-soft exit
!       * Set ifail to 111 to obtain monitoring information *
        ifail = 1
        Call d02gbf(a,b,n,tol,fcnf,fcng,c,d,gam,mnp,x,y,np,w,lw,iw,liw,ifail)

        If (ifail>=0) Then
          Write (nout,99999) 'Problem with epsilon = ', eps
        End If
        If (ifail==0) Then
          Write (nout,99998) np
          Write (nout,*) '        X(I)      Y(1,I)'
          Write (nout,99997)(x(j),y(1,j),j=1,np)
        Else
          Write (nout,99996) ifail
          Exit loop
        End If
      End Do loop

99999 Format (1X,A,E10.2)
99998 Format (/,1X,'Approximate solution on final mesh of ',I2,' points')
99997 Format (1X,2F11.4)
99996 Format (1X,/,1X,' ** D02GBF returned with IFAIL = ',I5)
    End Program d02gbfe
```

10.2 Program Data

```
D02GBF Example Program Data
  70                      : mnp
  1.0E-3                  : tol
  0                       : np
  0.0  1.0                : a, b
```
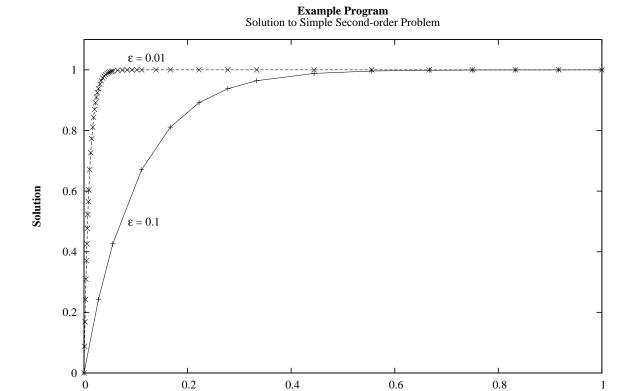
10.3 Program Results

```
 D02GBF Example Program Results

 Problem with epsilon =   0.10E+00

 Approximate solution on final mesh of 15 points
       X(I)      Y(1,I)
      0.0000      0.0000
      0.0278      0.2425
      0.0556      0.4263
      0.1111      0.6708
      0.1667      0.8112
```

```
        0.2222      0.8917
        0.2778      0.9379
        0.3333      0.9644
        0.4444      0.9883
        0.5556      0.9962
        0.6667      0.9988
        0.7500      0.9995
        0.8333      0.9998
        0.9167      0.9999
        1.0000      1.0000

 Problem with epsilon =   0.10E-01

 Approximate solution on final mesh of 49 points
        X(I)       Y(1,I)
        0.0000      0.0000
        0.0009      0.0884
        0.0019      0.1690
        0.0028      0.2425
        0.0037      0.3095
        0.0046      0.3706
        0.0056      0.4262
        0.0065      0.4770
        0.0074      0.5232
        0.0083      0.5654
        0.0093      0.6038
        0.0111      0.6708
        0.0130      0.7265
        0.0148      0.7727
        0.0167      0.8111
        0.0185      0.8431
        0.0204      0.8696
        0.0222      0.8916
        0.0241      0.9100
        0.0259      0.9252
        0.0278      0.9378
        0.0306      0.9529
        0.0333      0.9643
        0.0361      0.9730
        0.0389      0.9795
        0.0417      0.9845
        0.0444      0.9883
        0.0472      0.9911
        0.0500      0.9933
        0.0528      0.9949
        0.0556      0.9961
        0.0648      0.9985
        0.0741      0.9994
        0.0833      0.9998
        0.0926      0.9999
        0.1019      1.0000
        0.1111      1.0000
        0.1389      1.0000
        0.1667      1.0000
        0.2222      1.0000
        0.2778      1.0000
        0.3333      1.0000
        0.4444      1.0000
        0.5556      1.0000
        0.6667      1.0000
        0.7500      1.0000
        0.8333      1.0000
        0.9167      1.0000
        1.0000      1.0000
```

Example Program
Solution to Simple Second-order Problem