NAG Toolbox for MATLAB

d02gb

1 Purpose

d02gb solves a general linear two-point boundary-value problem for a system of ordinary differential equations using a deferred correction technique.

2 Syntax

```
[c, d, gam, x, y, np, ifail] = d02gb(a, b, tol, fcnf, fcng, c, d, gam,
x, np, 'n', n, 'mnp', mnp)
```

3 Description

d02gb solves the 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

$$\mathrm{C}y(a) + Dy(b) = \gamma. \tag{2}$$

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 specialisation of that used in (sub)program d02ra 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 functions 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* (ed B Childs, M Scott, J W Daniel, E Denman and P Nelson) **76** Springer–Verlag

5 Parameters

5.1 Compulsory Input Parameters

1: **a – double scalar**

a, the left-hand boundary point.

2: **b – double scalar**

b, the right-hand boundary point.

Constraint: **b** $>$ **a**.

3: **tol – double scalar**

A positive absolute error tolerance. If

$$a = x_1 < x_2 < \cdots < x_{\mathbf{np}} = b$$

is the final mesh, $z(x)$ is the approximate solution from d02gb and $y(x)$ is the true solution of equations (1) and (2) then, except in extreme cases, it is expected that

$$\|z - y\| \leq \mathbf{tol} \tag{3}$$

where

$$\|u\| = \max_{1 \leq i \leq \mathbf{n}} \max_{1 \leq j \leq \mathbf{np}} \left|u_i(x_j)\right|.$$

Constraint: **tol** > 0.0.

4: **fcnf – string containing name of m-file**

fcnf must evaluate the matrix $\mathbf{f}(x)$ in (1) at a general point x.

Its specification is:

```
[f] = fcnf(x)
```

Input Parameters

1: **x – double scalar**

The value of the independent variable x.

Output Parameters

1: **f**(n,n) **– double array**

The (i,j)th element of the matrix $\mathbf{f}(x)$, for $i,j = 1,2,\ldots,n$. (See Section 9 for an example.)

5: **fcng – string containing name of m-file**

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

Its specification is:

```
[g] = fcng(x)
```

Input Parameters

1: **x – double scalar**

The value of the independent variable x.

Output Parameters

1: **g**(n) **– double array**

The ith element of the vector $g(x)$, for $i = 1,2,\ldots,n$. (See Section 9 for an example.)

6: **c**(**n**,**n**) **– double array**

7: **d**(**n**,**n**) **– double array**

8: **gam**(**n**) **– double array**

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).

9: **x(mnp) – double array**

If **np** ≥ 4 (see **np**), the first **np** elements must define an initial mesh. Otherwise the elements of x need not be set.

Constraint:

$$\mathbf{a} = \mathbf{x}(1) < \mathbf{x}(2) < \cdots < \mathbf{x}(\mathbf{np}) = \mathbf{b}, \qquad \mathbf{np} \geq 4. \tag{4}$$

10: **np – int32 scalar**

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 $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(\mathbf{np})$ are used.

np ≥ 4

You must define an initial mesh **x** as in (4).

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the arrays **c**, **d**, **gam**, **y**. (An error is raised if these dimensions are not equal.)

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

Constraint: **n** ≥ 2.

2: **mnp – int32 scalar**

Default: The dimension of the arrays **x**, **y**. (An error is raised if these dimensions are not equal.)

the maximum permitted number of mesh points.

Constraint: **mnp** ≥ 32.

5.3 Input Parameters Omitted from the MATLAB Interface

w, lw, iw, liw

5.4 Output Parameters

1: **c(n,n) – double array**
2: **d(n,n) – double array**
3: **gam(n) – double array**

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 function 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.

4: **x(mnp) – double array**

$\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(\mathbf{np})$ define the final mesh (with the returned value of **np**) satisfying the relation (4).

5: **y(n,mnp) – double array**

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

$$\mathbf{y}(j,i) = z_j(x_i), \qquad i = 1, 2, \ldots, \mathbf{np}; j = 1, 2, \ldots, n$$

where **np** is the number of points in the final mesh.

The remaining columns of **y** are not used.

6: **np – int32 scalar**

The number of points in the final (returned) mesh.

7: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

One or more of the parameters **n**, **tol**, **np**, **mnp**, **lw** or **liw** is incorrectly set, $\mathbf{b} \leq \mathbf{a}$ 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, $\mathbf{iw}(1) = 0$ 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 $\mathbf{iw}(1)$ 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 $\mathbf{iw}(1)$ contains minus the number of non-identically zero columns.

ifail $= 3$

The function 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) below;

(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 d02gb. 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 $\mathbf{f}(x)$ 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 d02gb. Check all array subscripts and (sub)program parameter lists in calls to d02gb. 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 $\mathbf{iw}(1)$ on exit; and

(ii) as 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 $\mathbf{iw}(1)$ on exit.

In case above 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 function.

For example, if

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

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

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

7 Accuracy

The solution returned by the function 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 Further Comments

The time taken by d02gb 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 channel numbers on which this output is to appear by calls of x04aa (for error messages) or x04ab (for monitoring information) – see Section 9 for an example. Otherwise the default channel numbers will be used.

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

9 Example

d02gb_fcnf.m

```
function f = fcn(x)
      f=zeros(2,2);
      f(1,1) = 0;
      f(1,2) = 1;
      f(2,1) = 0;
      f(2,2) = -10;
```

```
d02gb_fcng.m
```

```
function g = fcng(x)
  g=zeros(2,1);
```

```
a = 0;
b = 1;
tol = 0.001;
c = [1, 0;
     0, 0];
d = [0, 0;
     1, 0];
gam = [0;
     1];
x = [0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
```

```
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0];
np = int32(0);
[cOut, dOut, gamOut, xOut, y, npOut, ifail] = ...
    d02gb(a, b, tol, 'd02gb_fcnf', 'd02gb_fcng', c, d, gam, x, np)
```

```
cOut =
     1     0
     0     0
dOut =
     0     0
     1     0
gamOut =
     0
     1
xOut =
     array elided
y =
     array elided
npOut =
          15
ifail =
           0
```