NAG Toolbox

nag_ode_bvp_ps_lin_solve (d02ue)

1 Purpose

nag_ode_bvp_ps_lin_solve (d02ue) finds the solution of a linear constant coefficient boundary value problem by using the Chebyshev integration formulation on a Chebyshev Gauss–Lobatto grid.

2 Syntax

```
[bmat, f, uc, resid, ifail] = nag_ode_bvp_ps_lin_solve(n, a, b, c, bmat, y, bvec,
f, 'm', m)
```

```
[bmat, f, uc, resid, ifail] = d02ue(n, a, b, c, bmat, y, bvec, f, 'm', m)
```

3 Description

nag_ode_bvp_ps_lin_solve (d02ue) solves the constant linear coefficient ordinary differential problem

$$\sum_{j=0}^{m} f_{j+1} \frac{d^j u}{dx^j} = f(x), \quad x \in [a,b]$$

subject to a set of m linear constraints at points $y_i \in [a,b]$, for $i = 1,2,\dots,m$:

$$\sum_{j=0}^{m} B_{i,j+1} \left(\frac{d^j u}{dx^j} \right)_{(x=y_i)} = \beta_i,$$

where $1 \leq m \leq 4$, B is an $m \times (m+1)$ matrix of constant coefficients and β_i are constants. The points y_i are usually either a or b.

The function $f(x)$ is supplied as an array of Chebyshev coefficients c_j, $j = 0,1,\dots,n$ for the function discretized on $n+1$ Chebyshev Gauss–Lobatto points (as returned by nag_ode_bvp_ps_lin_cgl_grid (d02uc)); the coefficients are normally obtained by a previous call to nag_ode_bvp_ps_lin_coeffs (d02ua). The solution and its derivatives (up to order m) are returned, in the form of their Chebyshev series representation, as arrays of Chebyshev coefficients; subsequent calls to nag_ode_bvp_ps_lin_cgl_vals (d02ub) will return the corresponding function and derivative values at the Chebyshev Gauss–Lobatto discretization points on $[a,b]$. Function and derivative values can be obtained on any uniform grid over the same range $[a,b]$ by calling the interpolation function nag_ode_bvp_ps_lin_grid_vals (d02uw).

4 References

Clenshaw C W (1957) The numerical solution of linear differential equations in Chebyshev series *Proc. Camb. Phil. Soc.* **53** 134–149

Coutsias E A, Hagstrom T and Torres D (1996) An efficient spectral method for ordinary differential equations with rational function coefficients *Mathematics of Computation* **65(214)** 611–635

Greengard L (1991) Spectral integration and two-point boundary value problems *SIAM J. Numer. Anal.* **28(4)** 1071–80

Lundbladh A, Hennigson D S and Johannson A V (1992) An efficient spectral integration method for the solution of the Navier–Stokes equations *Technical report FFA–TN* 1992–28 Aeronautical Research Institute of Sweden

Muite B K (2010) A numerical comparison of Chebyshev methods for solving fourth-order semilinear initial boundary value problems *Journal of Computational and Applied Mathematics* **234(2)** 317–342

5 Parameters

5.1 Compulsory Input Parameters

1: **n** – INTEGER

n, where the number of grid points is $n+1$.

Constraint: $\mathbf{n} \geq 8$ and $\mathbf{n}$ is even.

2: **a** – REAL (KIND=nag_wp)

a, the lower bound of domain $[a, b]$.

Constraint: $\mathbf{a} < \mathbf{b}$.

3: **b** – REAL (KIND=nag_wp)

b, the upper bound of domain $[a, b]$.

Constraint: $\mathbf{b} > \mathbf{a}$.

4: $\mathbf{c}(\mathbf{n}+\mathbf{1})$ – REAL (KIND=nag_wp) array

The Chebyshev coefficients c_j, $j = 0, 1, \ldots, n$, for the right hand side of the boundary value problem. Usually these are obtained by a previous call of nag_ode_bvp_ps_lin_coeffs (d02ua).

5: $\mathbf{bmat}(\mathbf{m}, \mathbf{m}+\mathbf{1})$ – REAL (KIND=nag_wp) array

$\mathbf{bmat}(i, j+1)$ must contain the coefficients $B_{i,j+1}$, for $i = 1, 2, \ldots, m$ and $j = 0, 1, \ldots, m$, in the problem formulation of Section 3.

6: $\mathbf{y}(\mathbf{m})$ – REAL (KIND=nag_wp) array

The points, y_i, for $i = 1, 2, \ldots, m$, where the boundary conditions are discretized.

7: $\mathbf{bvec}(\mathbf{m})$ – REAL (KIND=nag_wp) array

The values, β_i, for $i = 1, 2, \ldots, m$, in the formulation of the boundary conditions given in Section 3.

8: $\mathbf{f}(\mathbf{m}+\mathbf{1})$ – REAL (KIND=nag_wp) array

The coefficients, f_j, for $j = 1, 2, \ldots, m+1$, in the formulation of the linear boundary value problem given in Section 3. The highest order term, $\mathbf{f}(\mathbf{m}+1)$, needs to be nonzero to have a well posed problem.

5.2 Optional Input Parameters

1: **m** – INTEGER

Default: the first dimension of the array **bmat** and the dimension of the arrays **y**, **bvec**. (An error is raised if these dimensions are not equal.)

The order, m, of the boundary value problem to be solved.

Constraint: $1 \leq \mathbf{m} \leq 4$.

5.3 Output Parameters

1: $\mathbf{bmat}(\mathbf{m}, \mathbf{m}+\mathbf{1})$ – REAL (KIND=nag_wp) array

The coefficients have been scaled to form an equivalent problem defined on the domain $[-1, 1]$.

2: **f**$(\mathbf{m}+\mathbf{1})$ – REAL (KIND=nag_wp) array

The coefficients have been scaled to form an equivalent problem defined on the domain $[-1,1]$.

3: **uc**$(\mathbf{n}+\mathbf{1},\mathbf{m}+\mathbf{1})$ – REAL (KIND=nag_wp) array

The Chebyshev coefficients in the Chebyshev series representations of the solution and derivatives of the solution to the boundary value problem. The $n+1$ elements $\mathbf{uc}(1:\mathbf{n}+1,1)$ contain the coefficients representing the solution $U(x_i)$, for $i=0,1,\ldots,n$. $\mathbf{uc}(1:\mathbf{n}+1,j+1)$ contains the coefficients representing the jth derivative of U, for $j=1,2,\ldots,m$.

4: **resid** – REAL (KIND=nag_wp)

The maximum residual resulting from substituting the solution vectors returned in **uc** into both linear equations of Section 3 representing the linear boundary value problem and associated boundary conditions. That is

$$\max\left\{\max_{i=1,m}\left(\left|\sum_{j=0}^{m}B_{i,j+1}\left(\frac{d^ju}{dx^j}\right)_{(x=y_i)}-\beta_i\right|\right),\max_{i=1,n+1}\left(\left|\sum_{j=0}^{m}f_{j+1}\left(\frac{d^ju}{dx^j}\right)_{(x=x_i)}-f(x)\right|\right)\right\}.$$

5: **ifail** – INTEGER

ifail $=0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $=1$

Constraint: **n** is even.

Constraint: **n** ≥ 8.

ifail $=2$

Constraint: **a** $<$ **b**.

ifail $=3$

On entry, **f**$(\mathbf{m}+1)=0.0$.

ifail $=6$

Constraint: $1\leq$ **m** ≤ 4.

ifail $=7$

Internal error while unpacking matrix during iterative refinement.
Please contact NAG.

ifail $=8$

Singular matrix encountered during iterative refinement.
Please check that your system is well posed.

ifail $=9$ (*warning*)

During iterative refinement, the maximum number of iterations was reached.

ifail $=10$ (*warning*)

During iterative refinement, convergence was achieved, but the residual is more than $100\times$ ***machine precision***.

ifail $= -99$

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

ifail $= -399$

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

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

The accuracy should be close to ***machine precision*** for well conditioned boundary value problems.

8 Further Comments

The number of operations is of the order $n\log(n)$ and the memory requirements are $O(n)$; thus the computation remains efficient and practical for very fine discretizations (very large values of n). Collocation methods will be faster for small problems, but the method of nag_ode_bvp_ps_lin_solve (d02ue) should be faster for larger discretizations.

9 Example

This example solves the third-order problem $4U_{xxx} + 3U_{xx} + 2U_x + U = 2\sin x - 2\cos x$ on $[-\pi/2, \pi/2]$ subject to the boundary conditions $U[-\pi/2] = 0$, $3U_{xx}[-\pi/2] + 2U_x[-\pi/2] + U[-\pi/2] = 2$, and $3U_{xx}[\pi/2] + 2U_x[\pi/2] + U[\pi/2] = -2$ using the Chebyshev integration formulation on a Chebyshev Gauss–Lobatto grid of order 16.

9.1 Program Text

```
function d02ue_example

fprintf('d02ue example results\n\n');

n = nag_int(16);
a = -pi/2;
b =  pi/2;

% Set up boundary condition on left side of domain
y = [a, b];
% Set up Dirichlet condition using exact solution at x=a.
bmat = zeros(2, 3);
bmat(1, 1:2) = [1, 1];
bmat(2, 1:2) = [1, 1];
bvec = [cos(a) - sin(a), cos(b) - sin(b)];

% Set up problem definition
f = [1, 2, 3];

% Set up solution grid
[x, ifail] = d02uc(n, a, b);

% Set up problem right hand sides for grid and transform
f0 = -2*sin(x) - 2*cos(x);
[c, ifail] = d02ua(n, f0);

% Solve in coefficient space
[bmat, f, uc, resid, ifail] = d02ue(n, a, b, c, bmat, y, bvec, f);

% Evaluate solution and derivatives on Chebyshev grid
u = zeros(n+1, 3);
for q=0:2
  [u(:, q+1),  ifail] = d02ub(n, a, b, nag_int(q), uc(:, q+1));
```

```
end

% Print Solution
fprintf('\nNumerical solution U and its first two derivatives\n');
fprintf('      x          U          Ux         Uxx\n');
fprintf('%10.4f %10.4f %10.4f %10.4f\n', [x u]');
```

9.2 Program Results

```
     d02ue example results

Numerical solution U and its first two derivatives
      x          U          Ux         Uxx
   -1.5708    -0.0000     1.0000     0.0000
   -1.5406     0.0302     0.9995    -0.0302
   -1.4512     0.1193     0.9929    -0.1193
   -1.3061     0.2616     0.9652    -0.2616
   -1.1107     0.4440     0.8960    -0.4440
   -0.8727     0.6428     0.7661    -0.6428
   -0.6011     0.8247     0.5656    -0.8247
   -0.3064     0.9534     0.3017    -0.9534
   -0.0000     1.0000     0.0000    -1.0000
    0.3064     0.9534    -0.3017    -0.9534
    0.6011     0.8247    -0.5656    -0.8247
    0.8727     0.6428    -0.7661    -0.6428
    1.1107     0.4440    -0.8960    -0.4440
    1.3061     0.2616    -0.9652    -0.2616
    1.4512     0.1193    -0.9929    -0.1193
    1.5406     0.0302    -0.9995    -0.0302
    1.5708    -0.0000    -1.0000    -0.0000
```