NAG Toolbox

nag_ode_bvp_ps_lin_cgl_vals (d02ub)

1 Purpose

nag_ode_bvp_ps_lin_cgl_vals (d02ub) evaluates a function, or one of its lower order derivatives, from its Chebyshev series representation at Chebyshev Gauss–Lobatto points on $[a, b]$. The coefficients of the Chebyshev series representation required are usually derived from those returned by nag_ode_bvp_ps_lin_coeffs (d02ua) or nag_ode_bvp_ps_lin_solve (d02ue).

2 Syntax

```
[f, ifail] = nag_ode_bvp_ps_lin_cgl_vals(n, a, b, q, c)
[f, ifail] = d02ub(n, a, b, q, c)
```

3 Description

nag_ode_bvp_ps_lin_cgl_vals (d02ub) evaluates the Chebyshev series

$$S(\bar{x}) = \tfrac{1}{2}c_1T_0(\bar{x}) + c_2T_1(\bar{x}) + c_3T_2(\bar{x}) + \cdots + c_{n+1}T_n(\bar{x}),$$

or its derivative (up to fourth order) at the Chebyshev Gauss–Lobatto points on $[a, b]$. Here $T_j(\bar{x})$ denotes the Chebyshev polynomial of the first kind of degree j with argument $\bar{x}$ defined on $[-1, 1]$. In terms of your original variable, x say, the input values at which the function values are to be provided are

$$x_r = -\tfrac{1}{2}(b-a)\cos(\pi(r-1)/n) + \tfrac{1}{2}(b+a), \quad r = 1, 2, \ldots, n+1,$$

where b and a are respectively the upper and lower ends of the range of x over which the function is required.

The calculation is implemented by a forward one-dimensional discrete Fast Fourier Transform (DFT).

4 References

Canuto C (1988) *Spectral Methods in Fluid Dynamics* 502 Springer

Canuto C, Hussaini M Y, Quarteroni A and Zang T A (2006) *Spectral Methods: Fundamentals in Single Domains* Springer

Trefethen L N (2000) *Spectral Methods in MATLAB* SIAM

5 Parameters

5.1 Compulsory Input Parameters

1: **n** – INTEGER

n, where the number of grid points is $n + 1$. This is also the largest order of Chebyshev polynomial in the Chebyshev series to be computed.

Constraint: **n** > 0 and **n** is even.

2: **a** – REAL (KIND=nag_wp)

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

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

3: **b** – REAL (KIND=nag_wp)

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

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

4: **q** – INTEGER

The order, q, of the derivative to evaluate.

Constraint: $0 \leq$ **q** ≤ 4.

5: **c**(**n** + **1**) – REAL (KIND=nag_wp) array

The Chebyshev coefficients, c_i, for $i = 1, 2, \ldots, n + 1$.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **f**(**n** + **1**) – REAL (KIND=nag_wp) array

The derivatives $S^{(q)}x_i$, for $i = 1, 2, \ldots, n + 1$, of the Chebyshev series, S.

2: **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** > 0.

Constraint: **n** is even.

ifail $= 2$

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

ifail $= 3$

Constraint: $0 \leq$ **q** ≤ 4.

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

Evaluations of DFT to obtain function or derivative values should be an order n multiple of ***machine precision*** assuming full accuracy to ***machine precision*** in the given Chebyshev series representation.

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).

9 Example

See Section 10 in nag_ode_bvp_ps_lin_solve (d02ue).

9.1 Program Text

```
     function d02ub_example

fprintf('d02ub 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, a, b];
% Set up Dirichlet condition using exact solution at x=a.
bmat = zeros(3, 4);
bmat(1, 1) = 1;
bmat(2, 1:3) = [1, 2, 3];
bmat(3, 1:3) = [1, 2, 3];
bvec = [0, 2, -2];

% Set up problem definition
f = [1, 2, 3, 4];

% 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);

% Transform solution and derivative back to real space.
u = zeros(17, 4);
for q=0:3
  [u(:, q+1),  ifail] = d02ub(n, a, b, nag_int(q), uc(:, q+1));
end

% Print Solution
fprintf('\nNumerical solution U and its first three derivatives\n');
fprintf('      x          U          Ux         Uxx       Uxxx\n');
for i=1:17
  fprintf('%10.4f %10.4f %10.4f %10.4f %10.4f\n', x(i), u(i, :));
end
```

9.2 Program Results

```
     d02ub example results

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

```
   -0.0000    1.0000    0.0000   -1.0000   -0.0000
    0.3064    0.9534   -0.3017   -0.9534    0.3017
    0.6011    0.8247   -0.5656   -0.8247    0.5656
    0.8727    0.6428   -0.7661   -0.6428    0.7661
    1.1107    0.4440   -0.8960   -0.4440    0.8960
    1.3061    0.2616   -0.9652   -0.2616    0.9652
    1.4512    0.1193   -0.9929   -0.1193    0.9929
    1.5406    0.0302   -0.9995   -0.0302    0.9995
    1.5708   -0.0000   -1.0000    0.0000    1.0000
```