NAG Toolbox

nag_ode_bvp_coll_nth_comp (d02tg)

1 Purpose

nag_ode_bvp_coll_nth_comp (d02tg) solves a system of linear ordinary differential equations by least squares fitting of a series of Chebyshev polynomials using collocation.

2 Syntax

```
[c, ifail] = nag_ode_bvp_coll_nth_comp(m, l, x0, x1, k1, kp, coeff, bdyc, 'n', n)
[c, ifail] = d02tg(m, l, x0, x1, k1, kp, coeff, bdyc, 'n', n)
```

3 Description

nag_ode_bvp_coll_nth_comp (d02tg) calculates an approximate solution of a linear or linearized system of ordinary differential equations as a Chebyshev series. Suppose there are n differential equations for n variables $y_1, y_2, \ldots, y_n$, over the range (x_0, x_1). Let the ith equation be

$$\sum_{j=1}^{m_i+1} \sum_{k=1}^{n} f_{kj}^i(x) y_k^{(j-1)}(x) = r^i(x)$$

where $y_k^{(j)}(x) = \frac{d^j y_k(x)}{dx^j}$. **coeff** evaluates the coefficients $f_{kj}^i(x)$ and the right-hand side $r^i(x)$ for each i, $1 \leq i \leq n$, at any point x. The boundary conditions may be applied either at the end points or at intermediate points; they are written in the same form as the differential equations, and specified by **bdyc**. For example the jth boundary condition out of those associated with the ith differential equation takes the form

$$\sum_{j=1}^{l_i+1} \sum_{k=1}^{n} f_{kj}^{ij}(x^{ij}) y_k^{(j-1)}(x^{ij}) = r^{ij}(x^{ij}),$$

where x^{ij} lies between x_0 and x_1. It is assumed in this function that certain of the boundary conditions are associated with each differential equation. This is for your convenience; the grouping does not affect the results.

The degree of the polynomial solution must be the same for all variables. You specify the degree required, $k_1 - 1$, and the number of collocation points, k_p, in the range. The function sets up a system of linear equations for the Chebyshev coefficients, with n equations for each collocation point and one for each boundary condition. The collocation points are chosen at the extrema of a shifted Chebyshev polynomial of degree $k_p - 1$. The boundary conditions are satisfied exactly, and the remaining equations are solved by a least squares method. The result produced is a set of Chebyshev coefficients for the n functions $y_1, y_2, \ldots, y_n$, with the range normalized to $[-1, 1]$.

nag_fit_1dcheb_eval2 (e02ak) can be used to evaluate the components of the solution at any point on the range $[x_0, x_1]$ (see Section 10 for an example). nag_fit_1dcheb_deriv (e02ah) and nag_fit_1dcheb_ integ (e02aj) may be used to obtain Chebyshev series representations of derivatives and integrals (respectively) of the components of the solution.

4 References

Picken S M (1970) Algorithms for the solution of differential equations in Chebyshev-series by the selected points method *Report Math. 94* National Physical Laboratory

5 Parameters

5.1 Compulsory Input Parameters

1: **m**(**n**) – INTEGER array

m(i) must be set to the highest order derivative occurring in the ith equation, for $i = 1, 2, \ldots, n$.

Constraint: **m**$(i) \geq 1$, for $i = 1, 2, \ldots, n$.

2: **l**(**n**) – INTEGER array

l(i) must be set to the number of boundary conditions associated with the ith equation, for $i = 1, 2, \ldots, n$.

Constraint: **l**$(i) \geq 0$, for $i = 1, 2, \ldots, n$.

3: **x0** – REAL (KIND=nag_wp)

The left-hand boundary, x_0.

4: **x1** – REAL (KIND=nag_wp)

The right-hand boundary, x_1.

Constraint: **x1** > **x0**.

5: **k1** – INTEGER

The number of coefficients, k_1, to be returned in the Chebyshev series representation of the solution (hence, the degree of the polynomial approximation is **k1** $- 1$).

Constraint: $\mathbf{k1} \geq 1 + \max_{1 \leq i \leq \mathbf{n}} \mathbf{m}(i)$.

6: **kp** – INTEGER

The number of collocation points to be used, k_p.

Constraint: $\mathbf{n} \times \mathbf{kp} \geq \mathbf{n} \times \mathbf{k1} + \sum_{i=1}^{\mathbf{n}} \mathbf{l}(i)$.

7: **coeff** – SUBROUTINE, supplied by the user.

coeff defines the system of differential equations (see Section 3). It must evaluate the coefficient functions $f_{kj}^i(x)$ and the right-hand side function $r^i(x)$ of the ith equation at a given point. Only nonzero entries of the array **a** and **rhs** need be specifically assigned, since all elements are set to zero by nag_ode_bvp_coll_nth_comp (d02tg) before calling **coeff**.

```
[a, rhs] = coeff(x, ii, a, ia, ia1, rhs)
```

Input Parameters

1: **x** – REAL (KIND=nag_wp)

x, the point at which the functions must be evaluated.

2: **ii** – INTEGER

The equation for which the coefficients and right-hand side are to be evaluated.

3: **a**(**ia**, **ia1**) – REAL (KIND=nag_wp) array

All elements of **a** are set to zero.

4: **ia** – INTEGER
5: **ia1** – INTEGER

The first dimension of the array **a** and the second dimension of the array **a**.

6: **rhs** – REAL (KIND=nag_wp)

Is set to zero.

Output Parameters

1: **a**(**ia**, **ia1**) – REAL (KIND=nag_wp) array

a(k,j) must contain the value $f^i_{kj}(x)$, for $1 \leq k \leq n$, $1 \leq j \leq m_i + 1$.

2: **rhs** – REAL (KIND=nag_wp)

It must contain the value $r^i(x)$.

8: **bdyc** – SUBROUTINE, supplied by the user.

bdyc defines the boundary conditions (see Section 3). It must evaluate the coefficient functions f^{ij}_{kj} and right-hand side function r^{ij} in the jth boundary condition associated with the ith equation, at the point x^{ij} at which the boundary condition is applied. Only nonzero entries of the array **a** and **rhs** need be specifically assigned, since all elements are set to zero by nag_ode_bvp_coll_nth_comp (d02tg) before calling **bdyc**.

```
[x, a, rhs] = bdyc(ii, j, a, ia, ia1, rhs)
```

Input Parameters

1: **ii** – INTEGER

The differential equation with which the condition is associated.

2: **j** – INTEGER

The boundary condition for which the coefficients and right-hand side are to be evaluated.

3: **a**(**ia**, **ia1**) – REAL (KIND=nag_wp) array

All elements of **a** are set to zero.

4: **ia** – INTEGER
5: **ia1** – INTEGER

The first dimension of the array **a** and the second dimension of the array **a**.

6: **rhs** – REAL (KIND=nag_wp)

Is set to zero.

Output Parameters

1: **x** – REAL (KIND=nag_wp)

x^{ij}, the value at which the boundary condition is applied.

2: **a**(**ia**, **ia1**) – REAL (KIND=nag_wp) array

The value $f^{ij}_{kj}(x^{ij})$, for $1 \leq k \leq n$, $1 \leq j \leq m_i + 1$.

3: **rhs** – REAL (KIND=nag_wp)

The value $r^{ij}(x^{ij})$.

5.2 Optional Input Parameters

1: **n** – INTEGER

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

n, the number of differential equations in the system.

Constraint: $\mathbf{n} \geq 1$.

5.3 Output Parameters

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

The kth column of **c** contains the computed Chebyshev coefficients of the kth component of the solution, y_k; that is, the computed solution is:

$$y_k = {\sum_{i=1}^{k_1}}' \mathbf{c}(i,k) T_{i-1}(x), \quad 1 \leq k \leq n,$$

where $T_i(x)$ is the Chebyshev polynomial of the first kind and $\sum'$ denotes that the first coefficient, $\mathbf{c}(1,k)$, is halved.

2: **ifail** – INTEGER

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

6 Error Indicators and Warnings

Errors or warnings detected by the function:

$\mathbf{ifail} = 1$

On entry, $\mathbf{n} < 1$,
or $\mathbf{m}(i) < 1$ for some i,
or $\mathbf{l}(i) < 0$ for some i,
or $\mathbf{x0} \geq \mathbf{x1}$,
or $\mathbf{k1} < 1 + \mathbf{m}(i)$ for some i,
or $\mathbf{n} \times \mathbf{kp} < \mathbf{n} \times \mathbf{k1} + \sum_{i=1}^{n} \mathbf{l}(i)$,
or $ldc < \mathbf{k1}$.

$\mathbf{ifail} = 2$

On entry, lw is too small (see Section 5),
or $liw < \mathbf{n} \times \mathbf{k1}$.

$\mathbf{ifail} = 3$

Either the boundary conditions are not linearly independent, or the rank of the matrix of equations for the coefficients is less than the number of unknowns. Increasing **kp** may overcome this latter problem.

$\mathbf{ifail} = 4$

The least squares function nag_linsys_real_gen_lsqsol (f04am) has failed to correct the first approximate solution (see nag_linsys_real_gen_lsqsol (f04am)). Increasing **kp** may remove this difficulty.

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

Estimates of the accuracy of the solution may be obtained by using the checks described in Section 9. The Chebyshev coefficients are calculated by a stable numerical method.

8 Further Comments

The time taken by nag_ode_bvp_coll_nth_comp (d02tg) depends on the complexity of the system of differential equations, the degree of the polynomial solution and the number of matching points.

If the number of matching points k_p is equal to the number of coefficients k_1 minus the average number of boundary conditions $\frac{1}{n}\sum_{i=1}^{n} l_i$, then the least squares solution reduces to simple solution of linear equations and true collocation results. The accuracy of the solution may be checked by repeating the calculation with different values of k_1. If the Chebyshev coefficients decrease rapidly, the size of the last two or three gives an indication of the error. If they do not decrease rapidly, it may be desirable to use a different method. Note that the Chebyshev coefficients are calculated for the range normalized to $[-1, 1]$.

Generally the number of boundary conditions required is equal to the sum of the orders of the n differential equations. However, in some cases fewer boundary conditions are needed, because the assumption of a polynomial solution is equivalent to one or more boundary conditions (since it excludes singular solutions).

A system of **nonlinear** differential equations must be linearized before using the function. The calculation is repeated iteratively. On each iteration the linearized equation is used. In the example in Section 10, the y variables are to be determined at the current iteration whilst the z variables correspond to the solution determined at the previous iteration, (or the initial approximation on the first iteration). For a starting approximation, we may take, say, a linear function, and set up the appropriate Chebyshev coefficients before starting the iteration. For example, if $y_1 = ax + b$ in the range (x_0, x_1), we set B, the array of coefficients,

$\mathrm{B}(1,1) = a \times (x_0 + x_1) + 2 \times b$,

$\mathrm{B}(1,2) = a \times (x_1 - x_0)/2$,

and the remainder of the entries to zero.

In some cases a better initial approximation may be needed and can be obtained by using nag_fit_1dcheb_arb (e02ad) or nag_fit_1dcheb_glp (e02af) to obtain a Chebyshev series for an approximate solution. The coefficients of the current iterate must be communicated to **coeff** and **bdyc**, e.g., via a global variable. (See Section 10.) The convergence of the (Newton) iteration cannot be guaranteed in general, though it is usually satisfactory from a good starting approximation.

9 Example

This example solves the nonlinear system

$$2y_1' + \left(y_2^2 - 1\right)y_1 + y_2 = 0,$$

$$2y_2'' - y_1' = 0,$$

in the range $(-1, 1)$, with $y_1 = 0$, $y_2 = 3$, $y_2' = 0$ at $x = -1$.

Suppose an approximate solution is z_1, z_2 such that $y_1 \sim z_1$, $y_2 \sim z_2$: then the first equation gives, on linearizing,

$$2y_1' + \left(z_2^2 - 1\right)y_1 + (2z_1z_2 + 1)y_2 = 2z_1z_2^2.$$

The starting approximation is taken to be $z_1 = 0$, $z_2 = 3$. In the program below, the array B is used to hold the coefficients of the previous iterate (or of the starting approximation). We iterate until the Chebyshev coefficients converge to five figures. nag_fit_1dcheb_eval2 (e02ak) is used to calculate the solution from its Chebyshev coefficients.

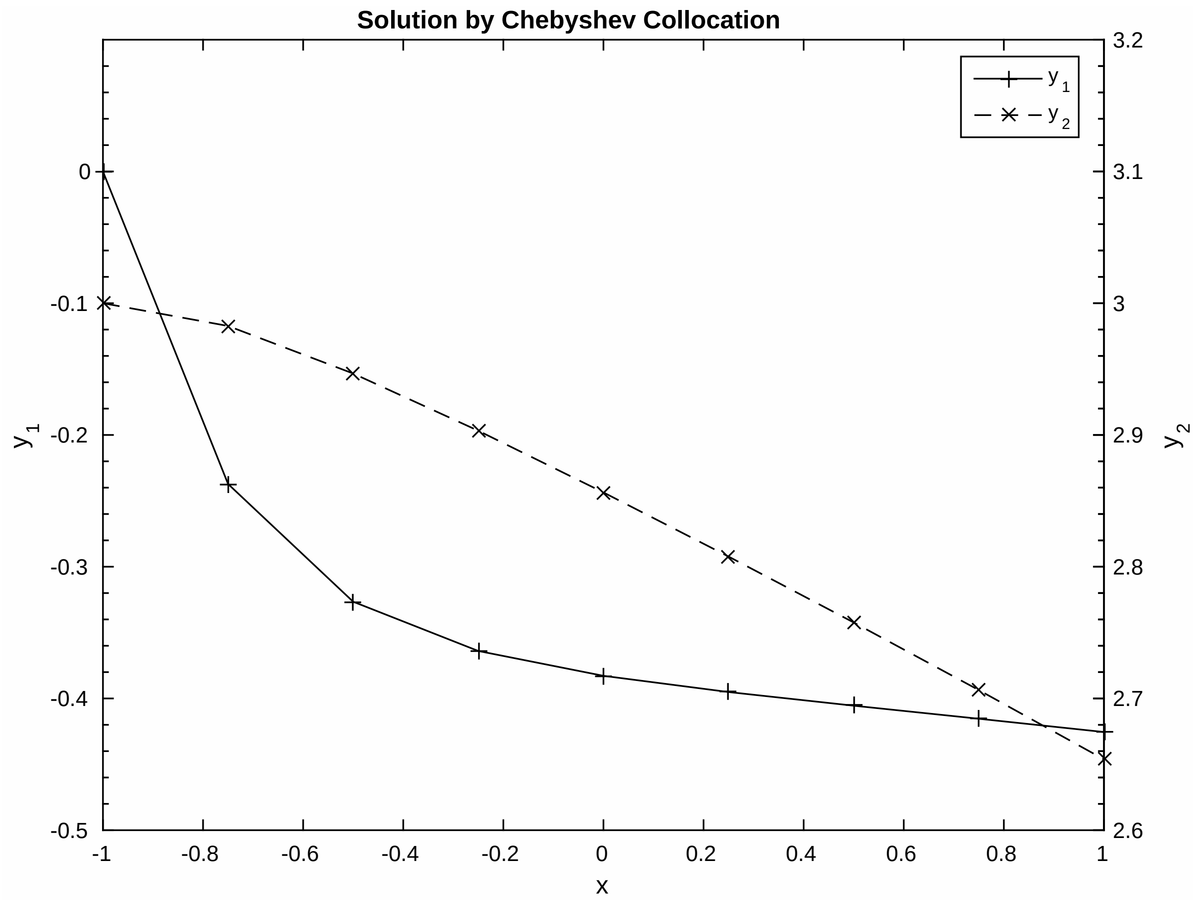
9.1 Program Text

```
     function d02tg_example

fprintf('d02tg example results\n\n');

global b x0 x1 ndegree

% Initialize variables and arrays.
n       = 2;
nderiv  = nag_int([1 2]);
nbc     = nderiv;
x0      = -1;
x1      = 1;
ncoeff  = nag_int(9);
ncolloc = nag_int(15);
ndegree = ncoeff-1;

% Iterate to covergence within tolerance of 1.0e0-5.
iter = 0;
emax = 1.0;
b = zeros(ncoeff, n);
b(1,2) = 6.0;
while emax >= 1.0e-05
  iter = iter + 1;

  [c, ifail] = d02tg(...
                   nderiv, nbc, x0, x1, ncoeff, ncolloc, ...
                   @coeff, @bdyc);
  % Output coefficients.
  fprintf('\n Iteration %2d. Chebyshev coefficients are:\n',iter);
  for j = 1:n
    fprintf('%8.4f',c(1:ncoeff,j));
    fprintf('\n');
  end

  emax = max(abs(c - b));
  b = c;
end

% Use these coefficients to evaluate the solution.
npt = 9;
fprintf('\n Solution evaluated at %d equally spaced points.\n\n', npt);
fprintf('%7s%16s\n','x','y');

% Prepare to store results for plotting.
dx = (x1-x0)/(double(npt-1));
xarray = [x0:dx:x1];
yarray = zeros(npt, n);
```

```
for ipt = 1:npt
  xx = xarray(ipt);
  fprintf('%10.4f', xx);
  for jeqn = 1:n
    [yarray(ipt, jeqn), ifail] = ...
    e02ak(...
           ndegree, x0, x1, c(:,jeqn), nag_int(1), xx);
    fprintf('%10.4f', yarray(ipt, jeqn));
  end
  fprintf('\n');
end

% Plot results.
fig1 = figure;
display_plot(xarray, yarray);

function [x, a, rhs] = bdyc(ii, jj, a, ia, ia1, rhs)
  % Evaluate coefficient functions and rhs of boundary conditions.

  x = -1;
  a(ii,jj) = 1;
  if (ii == 2 && jj == 1)
    rhs = 3;
  end

function [a, rhs] = coeff(x, ii, a, ia, ia1, rhs)
  % Evaluate coefficient functions and rhs of differential equations.

  global b x0 x1 ndegree

  if (ii <= 1)
    [z1, ifail] = e02ak(ndegree, x0, x1, b(:,1), nag_int(1), x);
    [z2, ifail] = e02ak(ndegree, x0, x1, b(:,2), nag_int(1), x);
    a(1,1) = z2*z2 - 1;
    a(1,2) = 2;
    a(2,1) = 2*z1*z2 + 1;
    rhs = 2*z1*z2*z2;
  else
    a(1,2) = -1;
    a(2,3) =  2;
  end

function display_plot(x, y)
  % Plot both curves.
  [haxes, hline1, hline2] = plotyy(x, y(:,1), x, y(:,2));
  % Set the axis limits and the tick specifications to beautify the plot.
  set(haxes(1), 'YLim', [-0.5 0.1]);
  set(haxes(1), 'YMinorTick', 'on');
  set(haxes(1), 'YTick', [-0.5:0.1:0]);
  set(haxes(2), 'YLim', [2.6 3.2]);
  set(haxes(2), 'YMinorTick', 'on');
  set(haxes(2), 'YTick', [2.6:0.1:3.2]);
  for iaxis = 1:2
    % These properties must be the same for both sets of axes.
    set(haxes(iaxis), 'XLim', [-1 1]);
  end
  % Add title.
  title('Solution by Chebyshev Collocation');
  % Label the axes.
  xlabel('x');
  ylabel(haxes(1), 'y_1');
  ylabel(haxes(2), 'y_2');
  % Add a legend.
  legend('y_1','y_2','Location','Best')
  % Set some features of the three lines.
  set(hline1, 'Linewidth', 0.25, 'Marker', '+', 'LineStyle', '-');
  set(hline2, 'Linewidth', 0.25, 'Marker', 'x', 'LineStyle', '--');
```

9.2 Program Results

```
     d02tg example results

 Iteration  1. Chebyshev coefficients are:
-0.5659 -0.1162  0.0906 -0.0468  0.0196 -0.0069  0.0021 -0.0006  0.0001
 5.7083 -0.1642 -0.0087  0.0059 -0.0025  0.0009 -0.0003  0.0001 -0.0000

 Iteration  2. Chebyshev coefficients are:
-0.6338 -0.1599  0.0831 -0.0445  0.0193 -0.0071  0.0023 -0.0006  0.0001
 5.6881 -0.1792 -0.0144  0.0053 -0.0023  0.0008 -0.0003  0.0001 -0.0000

 Iteration  3. Chebyshev coefficients are:
-0.6344 -0.1604  0.0828 -0.0446  0.0193 -0.0071  0.0023 -0.0006  0.0001
 5.6880 -0.1793 -0.0145  0.0053 -0.0023  0.0008 -0.0003  0.0001 -0.0000

 Iteration  4. Chebyshev coefficients are:
-0.6344 -0.1604  0.0828 -0.0446  0.0193 -0.0071  0.0023 -0.0006  0.0001
 5.6880 -0.1793 -0.0145  0.0053 -0.0023  0.0008 -0.0003  0.0001 -0.0000

 Solution evaluated at 9 equally spaced points.

      x              y
   -1.0000    0.0000    3.0000
   -0.7500   -0.2372    2.9827
   -0.5000   -0.3266    2.9466
   -0.2500   -0.3640    2.9032
    0.0000   -0.3828    2.8564
    0.2500   -0.3951    2.8077
    0.5000   -0.4055    2.7577
    0.7500   -0.4154    2.7064
    1.0000   -0.4255    2.6538
```


Solution by Chebyshev Collocation
y_1
y_2
x