NAG Toolbox for MATLAB

d02tg

1 Purpose

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

3 Description

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 $y_k^{(j)}(x) = \frac{d^j y_k(x)}{dx^j}$. The user-supplied (sub)program **coeff** provided by you 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 the user-supplied (sub)program **bdyc**. For example the jth boundary condition out of those associated with the ith differential equation takes the form

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

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 $[-1, 1]$.

e02ak can be used to evaluate the components of the solution at any point on the range $[x_0, x_1]$ (see Section 9 for an example). e02ah and 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) – int32 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) – int32 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 – double scalar**

The left-hand boundary, x_0.

4: **x1 – double scalar**

The right-hand boundary, x_1.

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

5: **k1 – int32 scalar**

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: **k1** $\geq 1 + \max\limits_{1 \leq i \leq \mathbf{n}} \mathbf{m}(i)$.

6: **kp – int32 scalar**

The number of collocation points to be used, k_p.

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

7: **coeff – string containing name of m-file**

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 d02tg before calling **coeff**.

Its specification is:

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

Input Parameters

1: **x – double scalar**

The point x at which the functions must be evaluated.

2: **ii – int32 scalar**

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

3: **a(ia,ia1) – double array**

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

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

4: **ia – int32 scalar**

5: **ia1 – int32 scalar**

The first and second dimensions of **a**, respectively.

6: **rhs – double scalar**

Is set to zero.

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

Output Parameters

1: **a(ia,ia1) – double array**

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

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

2: **rhs – double scalar**

Is set to zero.

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

8: **bdyc – string containing name of m-file**

bdyc defines the boundary conditions (see Section 3). It must evaluate the coefficient functions f_{kj}^{ij} 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 d02tg before calling **bdyc**.

Its specification is:

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

Input Parameters

1: **ii – int32 scalar**

The differential equation with which the condition is associated.

2: **j – int32 scalar**

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

3: **a(ia,ia1) – double array**

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

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

4: **ia – int32 scalar**

5: **ia1 – int32 scalar**

The first and second dimensions of **a**, respectively.

6: **rhs – double scalar**

Is set to zero.

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

Output Parameters

1: **x – double scalar**

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

2: **a(ia,ia1) – double array**

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

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

3: **rhs – double scalar**

Is set to zero.

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

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the arrays **m**, **l**, **c**. (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 Input Parameters Omitted from the MATLAB Interface

ldc, w, lw, iw, liw

5.4 Output Parameters

1: **c(ldc,n) – double 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:

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

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 – 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 $= 1$

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 $\mathbf{ldc} < \mathbf{k1}$.

ifail = 2

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

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.

ifail = 4

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

7 Accuracy

Estimates of the accuracy of the solution may be obtained by using the checks described in Section 8. The Chebyshev coefficients are calculated by a stable numerical method.

8 Further Comments

The time taken by 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 9, 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 e02ad or e02af to obtain a Chebyshev-series for an approximate solution. The coefficients of the current iterate must be communicated to user-supplied (sub)program **coeff** and user-supplied (sub)program **bdyc**, e.g., via a global

variable. (See Section 9.) The convergence of the (Newton) iteration cannot be guaranteed in general, though it is usually satisfactory from a good starting approximation.

9 Example

d02tg_bdyc.m

```
function [x, a, rhs] = bdyc(ii, jj, a, ia, ia1, rhs)
      x = -1;
      a(ii,jj) = 1;
      if (ii == 2 && jj == 1)
        rhs = 3;
      end
```

d02tg_coeff.m

```
function [a, rhs] = coeff(x, ii, a, ia, ia1, rhs)
      b1 = zeros(9,1);
      b2 = zeros(9,1);
      b2(1) = 6;
      if (ii <= 1)
         % [result, ifail] = e02ak(n, xmin, xmax, a, ia1, x);
         [z1, ifail] = e02ak(int32(8), -1, 1, b1, int32(1), x);
         [z2, ifail] = e02ak(int32(8), -1, 1, b2, int32(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
```

```
m = [int32(1);
     int32(2)];
l = [int32(1);
     int32(2)];
x0 = -1;
x1 = 1;
k1 = int32(9);
kp = int32(15);
[c, ifail] = d02tg(m, l, x0, x1, k1, kp, 'd02tg_coeff', 'd02tg_bdyc')
```

```
c =
   -0.5659    5.7083
   -0.1162   -0.1642
    0.0906   -0.0087
   -0.0468    0.0059
    0.0196   -0.0025
   -0.0069    0.0009
    0.0021   -0.0003
   -0.0006    0.0001
    0.0001   -0.0000
ifail =
           0
```