# NAG Toolbox for MATLAB

# d02ty

## 1 Purpose

d02ty interpolates on the solution of a general two-point boundary-value problem computed by d02tk.

## 2 Syntax

```
[y, rwork, ifail] = d02ty(x, neq, mmax, rwork, iwork)
```

## 3 Description

d02ty and its associated functions (d02tv, d02tk, d02tx and d02tz) solve the two-point boundary-value problem for a nonlinear mixed order system of ordinary differential equations

$$\begin{aligned} y_1^{(m_1)}(x) &= f_1\left(x, y_1, y_1^{(1)}, \ldots, y_1^{(m_1-1)}, y_2, \ldots y_n^{(m_n-1)}\right) \\ y_2^{(m_2)}(x) &= f_2\left(x, y_1, y_1^{(1)}, \ldots, y_1^{(m_1-1)}, y_2, \ldots y_n^{(m_n-1)}\right) \\ &\vdots \\ y_n^{(m_n)}(x) &= f_n\left(x, y_1, y_1^{(1)}, \ldots, y_1^{(m_1-1)}, y_2, \ldots y_n^{(m_n-1)}\right) \end{aligned}$$

over an interval $[a,b]$ subject to $p\ (>0)$ nonlinear boundary conditions at $a$ and $q\ (>0)$ nonlinear boundary conditions at $b$, where $p+q=\sum_{i=1}^{n} m_i$. Note that $y_i^{(m)}(x)$ is the $m$th derivative of the $i$th solution component. Hence $y_i^{(0)}(x)=y_i(x)$. The left boundary conditions at $a$ are defined as

$$g_i(z(y(a)))=0, \qquad i=1,2,\ldots,p,$$

and the right boundary conditions at $b$ as

$$\bar{g}_j(z(y(b)))=0, \qquad j=1,2,\ldots,q,$$

where $y=(y_1,y_2,\ldots,y_n)$ and

$$z(y(x))=\left(y_1(x), y_1^{(1)}(x), \ldots, y_1^{(m_1-1)}(x), y_2(x), \ldots, y_n^{(m_n-1)}(x)\right).$$

First, d02tv must be called to specify the initial mesh, error requirements and other details. Then, d02tk can be used to solve the boundary-value problem. After successful computation, d02tz can be used to ascertain details about the final mesh and other details of the solution procedure, and d02ty can be used to compute the approximate solution anywhere on the interval $[a,b]$ using interpolation.

The functions are based on modified versions of the codes COLSYS and COLNEW (see Ascher *et al.* 1979 and Ascher and Bader 1987). A comprehensive treatment of the numerical solution of boundary-value problems can be found in Ascher *et al.* 1988 and Keller 1992.

## 4 References

Ascher U M and Bader G 1987 A new basis implementation for a mixed order boundary value ODE solver *SIAM J. Sci. Stat. Comput.* **8** 483–500

Ascher U M, Christiansen J and Russell R D 1979 A collocation solver for mixed order systems of boundary value problems *Math. Comput.* **33** 659–679

Ascher U M, Mattheij R M M and Russell R D 1988 *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations* Prentice–Hall

Grossman C 1992 Enclosures of the solution of the Thomas–Fermi equation by monotone discretization *J. Comput. Phys.* **98** 26–32

Keller H B 1992 *Numerical Methods for Two-point Boundary-value Problems* Dover, New York

# 5 Parameters

## 5.1 Compulsory Input Parameters

1: **x – double scalar**

$x$, the independent variable.

*Constraint*: $a \leq \mathbf{x} \leq b$.

2: **neq – int32 scalar**

the number of differential equations.

*Constraint*: **neq** must be the same value as supplied to d02tv.

3: **mmax – int32 scalar**

The maximal order of the differential equations, $\max(m_i)$, for $i = 1, 2, \ldots, \mathbf{neq}$.

*Constraint*: **mmax** must contain the maximum value of the components of the parameter**m** as supplied to d02tv.

4: **rwork**$(*)$ **– double array**

**Note**: the dimension of the array **rwork** must be at least **lrwork** (see d02tv).

This must be the same array as supplied to d02tk and **must** remain unchanged between calls.

5: **iwork**$(*)$ **– int32 array**

**Note**: the dimension of the array **iwork** must be at least **liwork** (see d02tv).

This must be the same array as supplied to d02tk and **must** remain unchanged between calls.

## 5.2 Optional Input Parameters

None.

## 5.3 Input Parameters Omitted from the MATLAB Interface

None.

## 5.4 Output Parameters

1: **y**$(\mathbf{neq}, \mathbf{0} : \mathbf{mmax} - \mathbf{1})$ **– double array**

$\mathbf{y}(i,j)$ contains an approximation to $y_i^{(j)}(x)$, for $i = 1, 2, \ldots, \mathbf{neq}, j = 0, 1, \ldots, m_i - 1$. The remaining elements of **y** (where $m_i < \mathbf{mmax}$) are initialized to 0.0.

2: **rwork**$(*)$ **– double array**

**Note**: the dimension of the array **rwork** must be at least **lrwork** (see d02tv).

Contains information about the solution for use on subsequent calls to associated functions.

3: **ifail – int32 scalar**

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

# 6 Error Indicators and Warnings

**Note**: d02ty may return useful information for one or more of the following detected errors or warnings.

**ifail** $= 1$

On entry, an invalid value for **neq**, **mmax** ( $\neq \max(m_i)$ for some $i$) or **x** (outside the range $[a, b]$) was detected, or an invalid call to d02ty was made, for example without a previous call to the solver function d02tk.

**ifail** $= 2$

The solver function d02tk did not converge to a solution or did not satisfy the error requirements. The last solution computed by d02tk, for which convergence was obtained, has been used for interpolation by d02ty. The results returned by d02ty should be treated with extreme caution as regarding either their quality or accuracy. See Section 8.

## 7 Accuracy

If d02ty returns the value **ifail** $= 0$, the computed values of the solution components $y_i$ should be of similar accuracy to that specified by the parameter **tols** of d02tv. Note that during the solution process the error in the derivatives $y_i^{(j)}$, for $j = 1, 2, \ldots, m_i - 1$, has not been controlled and that the derivative values returned by d02ty are computed via differentiation of the piecewise polynomial approximation to $y_i$. See also Section 8.

## 8 Further Comments

If d02ty returns the value **ifail** $= 2$, and the solver function d02tk returned **ifail** $= 5$, then the accuracy of the interpolated values may be proportional to the quantity **ermx** as returned by d02tz.

If d02tk returned any other nonzero value for **ifail**, then nothing can be said regarding either the quality or accuracy of the values computed by d02ty.

## 9 Example

d02tx_ffun.m

```
function [f] = ffun(x, y, neq, m)
  global el;
  global en;
  global s;
  f = zeros(neq, 1);

      f(1)   =   el^3*(1-y(2,1)^2)   +   el^2*s*y(1,2)   -   el*(0.5*(3-
en)*y(1,1)*y(1,3)+en*y(1,2)^2);
    f(2)  =  el^2*s*(y(2,1)-1)  -  el*(0.5*(3-en)*y(1,1)*y(2,2)  +(en-
1)*y(1,2)*y(2,1));
```

d02tx_fjac.m

```
function [dfdy] = fjac(x, y, neq, m)
  global el;
  global en;
  global s;
  dfdy = zeros(neq, neq, 3);

  dfdy(1,2,1) = -2.0*el^3*y(2,1);
  dfdy(1,1,1) = -el*0.5*(3.0-en)*y(1,3);
  dfdy(1,1,2) = el^2*s - el*2.0*en*y(1,2);
  dfdy(1,1,3) = -el*0.5*(3.0-en)*y(1,1);
  dfdy(2,2,1) = el^2*s - el*(en-1.0)*y(1,2);
  dfdy(2,2,2) = -el*0.5*(3.0-en)*y(1,1);
  dfdy(2,1,1) = -el*0.5*(3.0-en)*y(2,2);
  dfdy(2,1,2) = -el*(en-1.0)*y(2,1);
```

d02tx_gafun.m

```
function [ga] = gafun(ya, neq, m, nlbc)
  global el;
  global en;
  global s;
  ga = zeros(nlbc, 1);

  ga(1) = ya(1,1);
  ga(2) = ya(1,2);
  ga(3) = ya(2,1);
```

d02tx_gajac.m

```
function [dgady] = gajac(ya, neq, m, nlbc)
  global el;
  global en;
  global s;
  dgady = zeros(nlbc, neq, 3);

  dgady(1,1,1) = 1;
  dgady(2,1,2) = 1;
  dgady(3,2,1) = 1;
```

d02tx_gbfun.m

```
function [gb] = gbfun(yb, neq, m, nrbc)
  global el;
  global en;
  global s;
  gb = zeros(nrbc, 1);

  gb(1) = yb(1,2);
  gb(2) = yb(2,1) - 1;
```

d02tx_gbjac.m

```
function [dgbdy] = gbjac(yb, neq, m, nrbc)
  global el;
  global en;
  global s;
  dgbdy = zeros(nrbc, neq, 3);

  dgbdy(1,1,2) = 1;
  dgbdy(2,2,1) = 1;
```

d02tx_guess.m

```
function [y, dym] = guess(x, neq, m)
  global el;
  global en;
  global s;
  y = zeros(neq, 3);
  dym = zeros(neq, 1);

  ex = x*el;
  expmx = exp(-ex);
  y(1,1) = -ex^2*expmx;
  y(1,2) = (-2*ex+ex^2)*expmx;
  y(1,3) = (-2+4*ex-ex^2)*expmx;
  y(2,1) = 1 - expmx;
  y(2,2) = expmx;
  dym(1) = (6-6*ex+ex^2)*expmx;
```

```
  dym(2) = -expmx;
```

```
m = [int32(3); int32(2)];
nlbc = int32(3);
nrbc = int32(2);
ncol = int32(6);
neq = int32(2);
tols = [0.00001; 0.00001];
nmesh = int32(21);
mxmesh = int32(250);
mesh = zeros(250,1);
ipmesh = zeros(250,1, 'int32');
ipmesh(1) = int32(1);
for i=2:nmesh
  mesh(i) = double(i-1)/double(nmesh-1);
  ipmesh(i) = int32(2);
end
ipmesh(nmesh) = int32(1);

global el;
el = 60;
global s;
s = 0.24;
global en;
en = 0.2;

ncont = int32(3);
mmax  = int32(3);

% Initialise
[work, iwork, ifail] = d02tv(m, nlbc, nrbc, ncol, tols, nmesh, mesh,
ipmesh);
% Solve
for j = 1:ncont
  fprintf('\n Tolerance = %8.1e, L = %8.3f, S = %6.4f\n\n', tols(1), el,
s);
  [work, iwork, ifail] = ...
    d02tk('d02tx_ffun', 'd02tx_fjac', 'd02tx_gafun', 'd02tx_gbfun', ...
    'd02tx_gajac', 'd02tx_gbjac', 'd02tx_guess', work, iwork);
  % Extract Mesh
  [nmesh, mesh, ipmesh, ermx, iermx, ijermx, ifail] = ...
    d02tz(mxmesh, work, iwork);
  fprintf(' Used a mesh of %d points\n', nmesh);
  fprintf(' Maximum error = %10.2e in interval %d for component %d\n\n',
...
    ermx, iermx, ijermx);
  % Print solution components on mesh
  fprintf(' Solution on original interval:\n');
  fprintf('      x        f          g\n');
  for i=1:16
    xx = double(i-1)*2/el;
    [y, work, ifail] = d02ty(xx, neq, mmax, work, iwork);
    fprintf(' %2.8f%11.4f%11.4f\n', xx*el, y(1,1), y(2,1));
  end
  for i=1:10
    xx = (30+(el-30)*double(i)/10)/el;
    [y, work, ifail] = d02ty(xx, neq, mmax, work, iwork);
    fprintf(' %2.8f%11.4f%11.4f\n', xx*el, y(1,1), y(2,1));
  end
  % Select Mesh for continuation
  if j < ncont
    el = 2*el;
    s = 0.6*s;
    nmesh = (nmesh+1)/2;
    [work, iwork, ifail] = d02tx(nmesh, mesh, ipmesh, work, iwork);
  end
end
```

```
Tolerance =  1.0e-05, L =   60.000, S = 0.2400

 Used a mesh of 21 points
 Maximum error =   2.66e-08 in interval 7 for component 1

 Solution on original interval:
      x         f          g
 0.00000000     0.0000     0.0000
 2.00000000    -0.9769     0.8011
 4.00000000    -2.0900     1.1459
 6.00000000    -2.6093     1.2389
 8.00000000    -2.5498     1.1794
 10.00000000    -2.1397     1.0478
 12.00000000    -1.7176     0.9395
 14.00000000    -1.5465     0.9206
 16.00000000    -1.6127     0.9630
 18.00000000    -1.7466     1.0068
 20.00000000    -1.8286     1.0244
 22.00000000    -1.8338     1.0185
 24.00000000    -1.7956     1.0041
 26.00000000    -1.7582     0.9940
 28.00000000    -1.7445     0.9926
 30.00000000    -1.7515     0.9965
 33.00000000    -1.7695     1.0019
 36.00000000    -1.7730     1.0018
 39.00000000    -1.7673     0.9998
 42.00000000    -1.7645     0.9993
 45.00000000    -1.7659     0.9999
 48.00000000    -1.7672     1.0002
 51.00000000    -1.7671     1.0001
 54.00000000    -1.7666     0.9999
 57.00000000    -1.7665     0.9999
 60.00000000    -1.7666     1.0000

 Tolerance =  1.0e-05, L =  120.000, S = 0.1440

 Used a mesh of 21 points
 Maximum error =   6.88e-06 in interval 7 for component 2

 Solution on original interval:
      x         f          g
 0.00000000     0.0000     0.0000
 2.00000000    -1.1406     0.7317
 4.00000000    -2.6531     1.1315
 6.00000000    -3.6721     1.3250
 8.00000000    -4.0539     1.3707
 10.00000000    -3.8285     1.3003
 12.00000000    -3.1339     1.1407
 14.00000000    -2.2469     0.9424
 16.00000000    -1.6146     0.8201
 18.00000000    -1.5472     0.8549
 20.00000000    -1.8483     0.9623
 22.00000000    -2.1761     1.0471
 24.00000000    -2.3451     1.0778
 26.00000000    -2.3236     1.0600
 28.00000000    -2.1784     1.0165
 30.00000000    -2.0214     0.9775
 39.00000000    -2.1109     1.0155
 48.00000000    -2.0362     0.9931
 57.00000000    -2.0709     1.0023
 66.00000000    -2.0588     0.9995
 75.00000000    -2.0616     1.0000
 84.00000000    -2.0615     1.0001
 93.00000000    -2.0611     0.9999
 102.00000000    -2.0614     1.0000
 111.00000000    -2.0613     1.0000
 120.00000000    -2.0613     1.0000

 Tolerance =  1.0e-05, L =  240.000, S = 0.0864
```

```
Used a mesh of 81 points
Maximum error =   3.30e-07 in interval 19 for component 2

Solution on original interval:
     x         f          g
0.00000000     0.0000     0.0000
2.00000000    -1.2756     0.6404
4.00000000    -3.1604     1.0463
6.00000000    -4.7459     1.3011
8.00000000    -5.8265     1.4467
10.00000000    -6.3412     1.5036
12.00000000    -6.2862     1.4824
14.00000000    -5.6976     1.3886
16.00000000    -4.6568     1.2263
18.00000000    -3.3226     1.0042
20.00000000    -2.0328     0.7718
22.00000000    -1.4035     0.6943
24.00000000    -1.6603     0.8218
26.00000000    -2.2975     0.9928
28.00000000    -2.8661     1.1139
30.00000000    -3.1641     1.1641
51.00000000    -2.5307     1.0279
72.00000000    -2.3520     0.9919
93.00000000    -2.3674     0.9975
114.00000000    -2.3799     1.0003
135.00000000    -2.3800     1.0002
156.00000000    -2.3792     1.0000
177.00000000    -2.3791     1.0000
198.00000000    -2.3792     1.0000
219.00000000    -2.3792     1.0000
240.00000000    -2.3792     1.0000
```