NAG Library Routine Document

D02TGF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

D02TGF solves a system of linear ordinary differential equations by least squares fitting of a series of Chebyshev polynomials using collocation.

2 Specification

```
SUBROUTINE D02TGF (N, M, L, X0, X1, K1, KP, C, LDC, COEFF, BDYC, W, LW, IW,     &
                   LIW, IFAIL)

INTEGER             N, M(N), L(N), K1, KP, LDC, LW, IW(LIW), LIW, IFAIL
REAL (KIND=nag_wp) X0, X1, C(LDC,N), W(LW)
EXTERNAL            COEFF, BDYC
```

3 Description

D02TGF 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

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

where x^{ij} lies between x_0 and x_1. It is assumed in this routine 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 routine 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 $[-1, 1]$.

E02AKF 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). E02AHF and E02AJF 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

1: N – INTEGER *Input*

On entry: n, the number of differential equations in the system.

Constraint: N ≥ 1.

2: M(N) – INTEGER array *Input*

On entry: 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 $i = 1, 2, \ldots, n$.

3: L(N) – INTEGER array *Input*

On entry: 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 $i = 1, 2, \ldots, n$.

4: X0 – REAL (KIND=nag_wp) *Input*

On entry: the left-hand boundary, x_0.

5: X1 – REAL (KIND=nag_wp) *Input*

On entry: the right-hand boundary, x_1.

Constraint: X1 $>$ X0.

6: K1 – INTEGER *Input*

On entry: 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 $\geq 1 + \max\limits_{1 \leq i \leq \mathrm{N}} \mathrm{M}(i)$.

7: KP – INTEGER *Input*

On entry: the number of collocation points to be used, k_p.

Constraint: N $\times$ KP $\geq$ N $\times$ K1 $+ \sum\limits_{i=1}^{\mathrm{N}} \mathrm{L}(i)$.

8: C(LDC,N) – REAL (KIND=nag_wp) array *Output*

On exit: 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:

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

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

9: LDC – INTEGER *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which D02TGF is called.

Constraint: LDC $\geq$ K1.

10: COEFF – SUBROUTINE, supplied by the user. *External Procedure*

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 D02TGF before calling COEFF.

The specification of COEFF is:

```
SUBROUTINE COEFF (X, I, A, IA, IA1, RHS)

INTEGER            I, IA, IA1
REAL (KIND=nag_wp) X, A(IA,IA1), RHS
```

Important: the dimension declaration for A must contain the variable IA, not an integer constant.

1: X – REAL (KIND=nag_wp) *Input*

On entry: x, the point at which the functions must be evaluated.

2: I – INTEGER *Input*

On entry: the equation for which the coefficients and right-hand side are to be evaluated.

3: A(IA,IA1) – REAL (KIND=nag_wp) array *Input/Output*

On entry: all elements of A are set to zero.

On exit: 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 – INTEGER *Input*

5: IA1 – INTEGER *Input*

On entry: the first dimension of the array A and the second dimension of the array A as declared in the (sub)program from which D02TGF is called.

6: RHS – REAL (KIND=nag_wp) *Input/Output*

On entry: is set to zero.

On exit: it must contain the value $r^i(x)$.

COEFF must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D02TGF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

11: BDYC – SUBROUTINE, supplied by the user. *External Procedure*

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 D02TGF before calling BDYC.

The specification of BDYC is:

```
SUBROUTINE BDYC (X, I, J, A, IA, IA1, RHS)

INTEGER            I, J, IA, IA1
REAL (KIND=nag_wp) X, A(IA,IA1), RHS
```

Important: the dimension declaration for A must contain the variable IA, not an integer constant.

1: X – REAL (KIND=nag_wp) *Output*

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

2: I – INTEGER *Input*

On entry: the differential equation with which the condition is associated.

3: J – INTEGER *Input*

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

4: A(IA,IA1) – REAL (KIND=nag_wp) array *Input/Output*

On entry: all elements of A are set to zero.

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

5: IA – INTEGER *Input*

6: IA1 – INTEGER *Input*

On entry: the first dimension of the array A and the second dimension of the array A as declared in the (sub)program from which D02TGF is called.

7: RHS – REAL (KIND=nag_wp) *Input/Output*

On entry: is set to zero.

On exit: the value $r^{ij}\left(x^{ij}\right)$.

BDYC must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D02TGF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

12: W(LW) – REAL (KIND=nag_wp) array *Workspace*

13: LW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which D02TGF is called.

Constraint: LW $\geq 2 \times (\mathrm{N} \times \mathrm{KP} + NL) \times (\mathrm{N} \times \mathrm{K1} + 1) + 7 \times \mathrm{N} \times \mathrm{K1}$, where $NL = \sum_{i=1}^{n} \mathrm{L}(i)$.

14: IW(LIW) – INTEGER array *Workspace*

15: LIW – INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which D02TGF is called.

Constraint: LIW $\geq \mathrm{N} \times \mathrm{K1} + 1$.

16: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, N < 1,
or M$(i) < 1$ for some i,
or L$(i) < 0$ for some i,
or X0 $\geq$ X1,
or K1 $<$ 1 + M(i) for some i,
or $\mathrm{N} \times \mathrm{KP} < \mathrm{N} \times \mathrm{K1} + \sum_{i=1}^{n} \mathrm{L}(i)$,
or LDC $<$ K1.

IFAIL $= 2$

On entry, LW is too small (see Section 5),
or LIW $<$ N $\times$ 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 routine F04AMF has failed to correct the first approximate solution (see F04AMF). 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 D02TGF 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 routine. 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 E02ADF or E02AFF to obtain a Chebyshev series for an approximate solution. The coefficients of the current iterate must be communicated to COEFF and BDYC, e.g., in COMMON. (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

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. E02AKF is used to calculate the solution from its Chebyshev coefficients.

9.1 Program Text

```
!   D02TGF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module d02tgfe_mod

!     D02TGF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter        :: coeff_tol = 1.0E-5_nag_wp
```

```
      Real (Kind=nag_wp), Parameter        :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter        :: two = 2.0_nag_wp
      Integer, Parameter                   :: n = 2, nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: x0, x1
      Integer                              :: k1
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: b(:,:)
      Integer                              :: l(n) = (/1,2/)
      Integer                              :: m(n) = (/1,2/)
    Contains
      Subroutine coeff(x,i,a,ia,ia1,rhs)

!       .. Use Statements ..
        Use nag_library, Only: e02akf
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Inout)   :: rhs
        Real (Kind=nag_wp), Intent (In)      :: x
        Integer, Intent (In)                 :: i, ia, ia1
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout)   :: a(ia,ia1)
!       .. Local Scalars ..
        Real (Kind=nag_wp)                   :: z1, z2
        Integer                              :: ifail
!       .. Executable Statements ..
        If (i<=1) Then

!         Evaulate z1, z2 at x using previous coeffs b.
          ifail = 0
          Call e02akf(k1,x0,x1,b(1,1),1,k1,x,z1,ifail)
          Call e02akf(k1,x0,x1,b(1,2),1,k1,x,z2,ifail)

          a(1,1) = z2*z2 - one
          a(1,2) = two
          a(2,1) = two*z1*z2 + one
          rhs = two*z1*z2*z2
        Else
          a(1,2) = -one
          a(2,3) = two
        End If
        Return
      End Subroutine coeff
      Subroutine bdyc(x,i,j,a,ia,ia1,rhs)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Inout)   :: rhs
        Real (Kind=nag_wp), Intent (Out)     :: x
        Integer, Intent (In)                 :: i, ia, ia1, j
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout)   :: a(ia,ia1)
!       .. Executable Statements ..
        x = -one
        a(i,j) = one
        If (i==2 .And. j==1) rhs = 3.0_nag_wp
        Return
      End Subroutine bdyc
    End Module d02tgfe_mod
    Program d02tgfe

!     D02TGF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d02tgf, e02akf, nag_wp
      Use d02tgfe_mod, Only: b, bdyc, coeff, coeff_tol, k1, l, m, n, nin,      &
                             nout, x0, x1
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: emax, x, xinc
      Integer                              :: i, ia1, ifail, iter, j, k, kp,   &
                                              ldc, liw, lsum, lw, mimax
```

```
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: c(:,:), w(:), y(:)
      Integer, Allocatable                 :: iw(:)
!     .. Intrinsic Procedures ..
      Intrinsic                            :: abs, max, real, sum
!     .. Executable Statements ..
      Write (nout,*) 'D02TGF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) mimax, kp
      Read (nin,*) x0, x1

      lsum = sum(l(1:n))
      k1 = mimax + 1
      ldc = k1
      liw = n*k1
      lw = 2*(n*kp+lsum)*(n*k1+1) + 7*n*k1
      Allocate (b(k1,n),c(ldc,n),w(lw),y(n),iw(liw))

!     initialize coefficients b(:,:) such that z1 = 0 and z2 = 3.
      b(1:k1,1:n) = 0.0_nag_wp
      b(1,2) = 6.0_nag_wp

!     Iterate until coefficients of linearized systems converge.
      iter = 0
      emax = 1.0_nag_wp
iters: Do While (emax>=coeff_tol)
        iter = iter + 1
        Write (nout,*)
        Write (nout,99999) ' Iteration', iter, ' Chebyshev coefficients are'

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        ifail = 0
        Call d02tgf(n,m,l,x0,x1,k1,kp,c,ldc,coeff,bdyc,w,lw,iw,liw,ifail)

        Write (nout,99998)(c(1:k1,j),j=1,n)
        emax = 0.0_nag_wp
        Do j = 1, n
          Do i = 1, k1
            emax = max(emax,abs(c(i,j)-b(i,j)))
          End Do
        End Do
        b(1:k1,1:n) = c(1:k1,1:n)
      End Do iters

      Deallocate (b)

!     Print solution on uniform mesh.
      k = 9
      ia1 = 1
      Write (nout,*)
      Write (nout,99999) 'Solution evaluated at', k, '  equally spaced points'
      Write (nout,*)
      Write (nout,99997) '      X ', (j,j=1,n)
      xinc = (x1-x0)/real(k-1,kind=nag_wp)
      x = x0
      Do i = 1, k
        Do j = 1, n
          ifail = 0
          Call e02akf(k1,x0,x1,c(1,j),ia1,k1,x,y(j),ifail)
        End Do
        Write (nout,99996) x, (y(j),j=1,n)
        x = x + xinc
      End Do

99999 Format (1X,A,I3,A)
99998 Format (1X,9F8.4)
99997 Format (1X,A,2('      Y(',I1,')'))
99996 Format (1X,3F10.4)
    End Program d02tgfe
```

9.2 Program Data

```
D02TGF Example Program Data
   8  15                : mimax, kp
   -1.0  1.0            : x0, x1
```

9.3 Program Results

```
 D02TGF Example Program 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)      Y(2)
    -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
```