D02TXF – NAG Fortran Library Routine Document

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

D02TXF allows a solution to a nonlinear two point boundary value problem computed by D02TKF to be used as an initial approximation in the solution of a related nonlinear two point boundary value problem in a continuation call to D02TKF.

2 Specification

```
SUBROUTINE D02TXF(MXMESH, NMESH, MESH, IPMESH, RWORK, IWORK, IFAIL)
INTEGER           MXMESH, NMESH, IPMESH(MXMESH), IWORK(*), IFAIL
real              MESH(MXMESH), RWORK(*)
```

3 Description

D02TXF and its associated routines (D02TKF, D02TVF, D02TYF and D02TZF) solve the two point boundary value problem for a nonlinear system of ordinary differential equations

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

over an interval $[a, b]$ subject to p (> 0) nonlinear boundary conditions at a and q (> 0) nonlinear boundary conditions at b, where $p + q = \sum_1^n m_i$. Note that $y_i^{(m)}(x)$ is the mth derivative of the ith solution component. Hence $y_i^{(0)}(x) = y_i(x)$. The left boundary conditions at a are defined as

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

and the right boundary conditions at b as

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

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

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

First, D02TVF must be called to specify the initial mesh, error requirements and other details. Then, D02TKF can be used to solve the boundary value problem. After successful computation, D02TZF can be used to ascertain details about the final mesh. D02TYF can be used to compute the approximate solution anywhere on the interval $[a, b]$ using interpolation.

If the boundary value problem being solved is one of a sequence of related problems, for example as part of some continuation process, then D02TXF should be used between calls to D02TKF. This avoids the overhead of a complete initialization when the setup routine D02TVF is used. D02TXF allows the solution values computed in the previous call to D02TKF to be used as an initial approximation for the solution in the next call to D02TKF.

The new initial mesh must be specified by the user. The previous mesh can be obtained by a call to D02TZF. It may be used unchanged as the new mesh, in which case any fixed points in the previous mesh remain as fixed points in the new mesh. Fixed and other points may be added or subtracted from the mesh by manipulation of the contents of the array argument IPMESH. Initial values for the solution components on the new mesh are computed by interpolation on the values for the solution components on the previous mesh.

The routines are based on modified versions of the codes COLSYS and COLNEW, [2] and [1]. A comprehensive treatment of the numerical solution of boundary value problems can be found in [3] and [4].

4 References

[1] 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

[2] 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

[3] Ascher U M, Mattheij R M M and Russell R D (1988) *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations* Prentice Hall, Englewood Cliffs, NJ

[4] Keller H B (1992) *Numerical Methods for Two-point Boundary-value Problems* Dover, New York

5 Parameters

1: MXMESH — INTEGER *Input*

On entry: the maximum number of points allowed in the mesh.

Constraint: this must be identical to the value supplied for the argument MXMESH in the prior call to D02TVF.

2: NMESH — INTEGER *Input*

On entry: the number of points to be used in the new initial mesh.

Suggested value: $(n^* + 1)/2$, where n^* is the number of mesh points used in the previous mesh as returned in the argument NMESH of D02TZF.

Constraint: $6 \leq$ NMESH $\leq$ (MXMESH+1)/2.

3: MESH(MXMESH) — ***real*** array *Input*

On entry: the NMESH points to be used in the new initial mesh as specified by IPMESH.

Suggested values: the argument MESH returned from a call to D02TZF.

Constraints:

MESH(i_j) < MESH(i_{j+1}), for $j = 1, 2, \ldots,$NMESH-1 ; the values of $i_1, i_2, \ldots, i_{\text{NMESH}}$ are defined in IPMESH below.

MESH(i_1) must contain the left boundary point, a, and MESH(i_{NMESH}) must contain the right boundary point, b, as specified in the previous call to D02TVF.

4: IPMESH(MXMESH) — INTEGER array *Input*

On entry: specifies the points in MESH to be used as the new initial mesh. Let $\{i_j : j = 1, 2, \ldots,\text{NMESH}\}$ be the set of array indices of IPMESH such that IPMESH(i_j) = 1 or 2 and $1 = i_1 < i_2 < \ldots < i_{\text{NMESH}}$. Then MESH($i_j$) will be included in the new initial mesh. If IPMESH(i_j) = 1, then MESH(i_j) will be a fixed point in the new initial mesh. If IPMESH(k) = 3 for any k, then MESH(k) will not be included in the new mesh.

Suggested values: the argument IPMESH returned in a call to D02TZF.

Constraints:

IPMESH(k) = 1, 2 or 3 for $k = 1, 2, \ldots, i_{\text{NMESH}}$

IPMESH(1) = IPMESH(i_{NMESH}) = 1.

5: RWORK($*$) — ***real*** array *Input/Output*

On entry: this must be the same array as supplied to D02TKF and **must** remain unchanged between calls.

On exit: contains information about the solution for use on subsequent calls to associated routines.

6: IWORK(∗) — INTEGER array *Input/Output*

On entry: this must be the same array as supplied to D02TKF and **must** remain unchanged between calls.

On exit: contains information about the solution for use on subsequent calls to associated routines.

7: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL $= 0$ unless the routine detects an error (see Section 6).

6 Errors 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 detected by the routine:

IFAIL $= 1$

An invalid call to D02TXF was made, for example without a previous successful call to the solver routine D02TKF, or, on entry, an invalid value for NMESH, MESH or IPMESH was detected. If on entry IFAIL $= 0$ or -1, the precise form of the error will be detailed on the current error message unit (as defined by X04AAF).

7 Accuracy

Not applicable.

8 Further Comments

For problems where sharp changes of behaviour are expected over short intervals it may be advisable to:

cluster the mesh points where sharp changes in behaviour are expected;

maintain fixed points in the mesh using the argument IPMESH to ensure that the remeshing process does not inadvertently remove mesh points from areas of known interest.

In the absence of any other information about the expected behaviour of the solution, using the values suggested in Section 5 for NMESH, IPMESH and MESH is strongly recommended.

9 Example

The following example is used to illustrate the use of continuation, solution on an infinite range, and solution of a system of two differential equations of orders 3 and 2. See also D02TKF, D02TVF, D02TYF and D02TZF, for the illustration of other facilities.

Consider the problem of swirling flow over an infinite stationary disk with a magnetic field along the axis of rotation. See [3] and the references therein. After transforming from a cylindrical coordinate system (r, θ, z), in which the θ component of the corresponding velocity field behaves like r^{-n}, the governing equations are

$$\begin{aligned} f''' + \frac{1}{2}(3-n)ff'' + n(f')^2 + g^2 - sf' &= \gamma^2 \\ g'' + \frac{1}{2}(3-n)fg' + (n-1)gf' - s(g-1) &= 0 \end{aligned}$$

with boundary conditions

$$f(0) = f'(0) = g(0) = 0, \quad f'(\infty) = 0, \quad g(\infty) = \gamma,$$

where s is the magnetic field strength, and γ is the Rossby number.

Some solutions of interest are for $\gamma = 1$, small n and $s \to 0$. An added complication is the infinite range, which we approximate by $[0, L]$. We choose $n = 0.2$ and first solve for $L = 60.0, s = 0.24$ using the initial approximations $f(x) = -x^2e^{-x}$ and $g(x) = 1.0 - e^{-x}$, which satisfy the boundary conditions, on a uniform mesh of 21 points. Simple continuation on the parameters L and s using the values $L = 120.0, s = 0.144$ and then $L = 240.0, s = 0.0864$ is used to compute further solutions. We use the suggested values for NMESH, IPMESH and MESH in the call to D02TXF prior to a continuation call, that is only every second point of the preceding mesh is used.

The equations are first mapped onto $[0, 1]$ to yield

$$\begin{aligned} f''' &= L^3(\gamma^2 - g^2) + L^2sg' - L(\frac{1}{2}(3-n)ff'' + n(g')^2) \\ g'' &= L^2s(g-1) - L(\frac{1}{2}(3-n)fg' + (n-1)f'g). \end{aligned}$$

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     D02TXF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NEQ, MMAX, NLBC, NRBC, NCOL, MXMESH
      PARAMETER        (NEQ=2,MMAX=3,NLBC=3,NRBC=2,NCOL=6,MXMESH=250)
      INTEGER          LRWORK, LIWORK
      PARAMETER        (LRWORK=MXMESH*(109*NEQ**2+78*NEQ+7),
     +                 LIWORK=MXMESH*(11*NEQ+6))
*     .. Scalars in Common ..
      real             EL, EN, S
*     .. Local Scalars ..
      real             ERMX, XX
      INTEGER          I, IERMX, IFAIL, IJERMX, J, NCONT, NMESH
      LOGICAL          FAILED
*     .. Local Arrays ..
      real             MESH(MXMESH), TOL(NEQ), WORK(LRWORK),
     +                 Y(NEQ,0:MMAX-1)
      INTEGER          IPMESH(MXMESH), IWORK(LIWORK), M(NEQ)
*     .. External Subroutines ..
      EXTERNAL         D02TKF, D02TVF, D02TXF, D02TYF, D02TZF, FFUN,
     +                 FJAC, GAFUN, GAJAC, GBFUN, GBJAC, GUESS
*     .. Intrinsic Functions ..
      INTRINSIC        real
*     .. Common blocks ..
      COMMON           /PROBS/EN, S, EL
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D02TXF Example Program Results'
      WRITE (NOUT,*)
      NMESH = 21
      MESH(1) = 0.0e0
      IPMESH(1) = 1
      DO 20 I = 2, NMESH - 1
         MESH(I) = real(I-1)/real(NMESH-1)
         IPMESH(I) = 2
   20 CONTINUE
      IPMESH(NMESH) = 1
      MESH(NMESH) = 1.0e0
      M(1) = 3
```

```
      M(2) = 2
      TOL(1) = 1.0e-5
      TOL(2) = TOL(1)
      IFAIL = 0
      CALL D02TVF(NEQ,M,NLBC,NRBC,NCOL,TOL,MXMESH,NMESH,MESH,IPMESH,
     +            WORK,LRWORK,IWORK,LIWORK,IFAIL)
*     Initialize number of continuation steps
      NCONT = 3
*     Initialize problem dependent parameters
      EL = 6.0e1
      S = 0.24e0
      EN = 0.2e0
      DO 80 J = 1, NCONT
         WRITE (NOUT,99997) TOL(1), EL, S
         IFAIL = -1
*     Solve
         CALL D02TKF(FFUN,FJAC,GAFUN,GBFUN,GAJAC,GBJAC,GUESS,WORK,IWORK,
     +               IFAIL)
         FAILED = IFAIL .NE. 0
         IFAIL = 0
*     Extract mesh
         CALL D02TZF(MXMESH,NMESH,MESH,IPMESH,ERMX,IERMX,IJERMX,WORK,
     +               IWORK,IFAIL)
         WRITE (NOUT,99996) NMESH, ERMX, IERMX, IJERMX
         IF (FAILED) GO TO 100
*     Print solution components on mesh
         WRITE (NOUT,99999)
         DO 40 I = 1, 16
            XX = real(I-1)*2.0e0/EL
            CALL D02TYF(XX,Y,NEQ,MMAX,WORK,IWORK,IFAIL)
            WRITE (NOUT,99998) XX*EL, Y(1,0), Y(2,0)
   40    CONTINUE
         DO 60 I = 1, 10
            XX = (3.0e1+(EL-3.0e1)*real(I)/10.0e0)/EL
            CALL D02TYF(XX,Y,NEQ,MMAX,WORK,IWORK,IFAIL)
            WRITE (NOUT,99998) XX*EL, Y(1,0), Y(2,0)
   60    CONTINUE
*     Select mesh for continuation
         IF (J.LT.NCONT) THEN
            EL = 2.0e0*EL
            S = 0.6e0*S
            NMESH = (NMESH+1)/2
            CALL D02TXF(MXMESH,NMESH,MESH,IPMESH,WORK,IWORK,IFAIL)
         END IF
   80 CONTINUE
  100 CONTINUE
      STOP
*
99999 FORMAT (/' Solution on original interval:',/'    ',' x        f',
     +       '          g')
99998 FORMAT (' ',F8.2,2F11.4)
99997 FORMAT (//' Tolerance = ',e8.1,'  L = ',F8.3,'  S = ',F6.4)
99996 FORMAT (/' Used a mesh of ',I4,' points',/' Maximum error = ',
     +       e10.2,'  in interval ',I4,' for component ',I4)
      END
```

```
      SUBROUTINE FFUN(X,Y,NEQ,M,F)
*     .. Scalar Arguments ..
      real              X
      INTEGER           NEQ
*     .. Array Arguments ..
      real              F(NEQ), Y(NEQ,0:*)
      INTEGER           M(NEQ)
*     .. Scalars in Common ..
      real              EL, EN, S
*     .. Common blocks ..
      COMMON            /PROBS/EN, S, EL
*     .. Executable Statements ..
      F(1) = EL**3*(1.0e0-Y(2,0)**2) + EL**2*S*Y(1,1) -
     +       EL*(0.5e0*(3.0e0-EN)*Y(1,0)*Y(1,2)+EN*Y(1,1)**2)
      F(2) = EL**2*S*(Y(2,0)-1.0e0) - EL*(0.5e0*(3.0e0-EN)*Y(1,0)*Y(2,1)
     +       +(EN-1.0e0)*Y(1,1)*Y(2,0))
      RETURN
      END
      SUBROUTINE FJAC(X,Y,NEQ,M,DF)
*     .. Scalar Arguments ..
      real              X
      INTEGER           NEQ
*     .. Array Arguments ..
      real              DF(NEQ,NEQ,0:*), Y(NEQ,0:*)
      INTEGER           M(NEQ)
*     .. Scalars in Common ..
      real              EL, EN, S
*     .. Common blocks ..
      COMMON            /PROBS/EN, S, EL
*     .. Executable Statements ..
      DF(1,2,0) = -2.0e0*EL**3*Y(2,0)
      DF(1,1,0) = -EL*0.5e0*(3.0e0-EN)*Y(1,2)
      DF(1,1,1) = EL**2*S - EL*2.0e0*EN*Y(1,1)
      DF(1,1,2) = -EL*0.5e0*(3.0e0-EN)*Y(1,0)
      DF(2,2,0) = EL**2*S - EL*(EN-1.0e0)*Y(1,1)
      DF(2,2,1) = -EL*0.5e0*(3.0e0-EN)*Y(1,0)
      DF(2,1,0) = -EL*0.5e0*(3.0e0-EN)*Y(2,1)
      DF(2,1,1) = -EL*(EN-1.0e0)*Y(2,0)
      RETURN
      END
      SUBROUTINE GAFUN(YA,NEQ,M,NLBC,GA)
*     .. Scalar Arguments ..
      INTEGER           NEQ, NLBC
*     .. Array Arguments ..
      real              GA(NLBC), YA(NEQ,0:*)
      INTEGER           M(NEQ)
*     .. Executable Statements ..
      GA(1) = YA(1,0)
      GA(2) = YA(1,1)
      GA(3) = YA(2,0)
      RETURN
      END
      SUBROUTINE GBFUN(YB,NEQ,M,NRBC,GB)
*     .. Scalar Arguments ..
      INTEGER           NEQ, NRBC
```

```
*     .. Array Arguments ..
      real              GB(NRBC), YB(NEQ,0:*)
      INTEGER           M(NEQ)
*     .. Executable Statements ..
      GB(1) = YB(1,1)
      GB(2) = YB(2,0) - 1.0e0
      RETURN
      END
      SUBROUTINE GAJAC(YA,NEQ,M,NLBC,DGA)
*     .. Scalar Arguments ..
      INTEGER           NEQ, NLBC
*     .. Array Arguments ..
      real              DGA(NLBC,NEQ,0:*), YA(NEQ,0:*)
      INTEGER           M(NEQ)
*     .. Executable Statements ..
      DGA(1,1,0) = 1.0e0
      DGA(2,1,1) = 1.0e0
      DGA(3,2,0) = 1.0e0
      RETURN
      END
      SUBROUTINE GBJAC(YB,NEQ,M,NRBC,DGB)
*     .. Scalar Arguments ..
      INTEGER           NEQ, NRBC
*     .. Array Arguments ..
      real              DGB(NRBC,NEQ,0:*), YB(NEQ,0:*)
      INTEGER           M(NEQ)
*     .. Executable Statements ..
      DGB(1,1,1) = 1.0e0
      DGB(2,2,0) = 1.0e0
      RETURN
      END
      SUBROUTINE GUESS(X,NEQ,M,Z,DMVAL)
*     .. Scalar Arguments ..
      real              X
      INTEGER           NEQ
*     .. Array Arguments ..
      real              DMVAL(NEQ), Z(NEQ,0:*)
      INTEGER           M(NEQ)
*     .. Scalars in Common ..
      real              EL, EN, S
*     .. Local Scalars ..
      real              EX, EXPMX
*     .. Intrinsic Functions ..
      INTRINSIC         EXP
*     .. Common blocks ..
      COMMON            /PROBS/EN, S, EL
*     .. Executable Statements ..
      EX = X*EL
      EXPMX = EXP(-EX)
      Z(1,0) = -EX**2*EXPMX
      Z(1,1) = (-2.0e0*EX+EX**2)*EXPMX
      Z(1,2) = (-2.0e0+4.0e0*EX-EX**2)*EXPMX
      Z(2,0) = 1.0e0 - EXPMX
      Z(2,1) = EXPMX
      DMVAL(1) = (6.0e0-6.0e0*EX+EX**2)*EXPMX
      DMVAL(2) = -EXPMX
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
D02TXF Example Program Results



Tolerance =  0.1E-04  L =   60.000  S = 0.2400

Used a mesh of   21 points
Maximum error =   0.27E-07  in interval    7 for component    1

Solution on original interval:
     x        f          g
    0.00     0.0000     0.0000
    2.00    -0.9769     0.8011
    4.00    -2.0900     1.1459
    6.00    -2.6093     1.2389
    8.00    -2.5498     1.1794
   10.00    -2.1397     1.0478
   12.00    -1.7176     0.9395
   14.00    -1.5465     0.9206
   16.00    -1.6127     0.9630
   18.00    -1.7466     1.0068
   20.00    -1.8286     1.0244
   22.00    -1.8338     1.0185
   24.00    -1.7956     1.0041
   26.00    -1.7582     0.9940
   28.00    -1.7445     0.9926
   30.00    -1.7515     0.9965
   33.00    -1.7695     1.0019
   36.00    -1.7730     1.0018
   39.00    -1.7673     0.9998
   42.00    -1.7645     0.9993
   45.00    -1.7659     0.9999
   48.00    -1.7672     1.0002
   51.00    -1.7671     1.0001
   54.00    -1.7666     0.9999
   57.00    -1.7665     0.9999
   60.00    -1.7666     1.0000


Tolerance =  0.1E-04  L =  120.000  S = 0.1440

Used a mesh of   21 points
Maximum error =   0.69E-05  in interval    7 for component    2

Solution on original interval:
     x        f          g
    0.00     0.0000     0.0000
    2.00    -1.1406     0.7317
    4.00    -2.6531     1.1315
    6.00    -3.6721     1.3250
    8.00    -4.0539     1.3707
   10.00    -3.8285     1.3003
   12.00    -3.1339     1.1407
```

```
    14.00    -2.2469     0.9424
    16.00    -1.6146     0.8201
    18.00    -1.5472     0.8549
    20.00    -1.8483     0.9623
    22.00    -2.1761     1.0471
    24.00    -2.3451     1.0778
    26.00    -2.3236     1.0600
    28.00    -2.1784     1.0165
    30.00    -2.0214     0.9775
    39.00    -2.1109     1.0155
    48.00    -2.0362     0.9931
    57.00    -2.0709     1.0023
    66.00    -2.0588     0.9995
    75.00    -2.0616     1.0000
    84.00    -2.0615     1.0001
    93.00    -2.0611     0.9999
   102.00    -2.0614     1.0000
   111.00    -2.0613     1.0000
   120.00    -2.0613     1.0000


 Tolerance =  0.1E-04  L =  240.000  S = 0.0864

 Used a mesh of   81 points
 Maximum error =   0.33E-06  in interval   19 for component    2

 Solution on original interval:
      x        f          g
     0.00     0.0000     0.0000
     2.00    -1.2756     0.6404
     4.00    -3.1604     1.0463
     6.00    -4.7459     1.3011
     8.00    -5.8265     1.4467
    10.00    -6.3412     1.5036
    12.00    -6.2862     1.4824
    14.00    -5.6976     1.3886
    16.00    -4.6568     1.2263
    18.00    -3.3226     1.0042
    20.00    -2.0328     0.7718
    22.00    -1.4035     0.6943
    24.00    -1.6603     0.8218
    26.00    -2.2975     0.9928
    28.00    -2.8661     1.1139
    30.00    -3.1641     1.1641
    51.00    -2.5307     1.0279
    72.00    -2.3520     0.9919
    93.00    -2.3674     0.9975
   114.00    -2.3799     1.0003
   135.00    -2.3800     1.0002
   156.00    -2.3792     1.0000
   177.00    -2.3791     1.0000
   198.00    -2.3792     1.0000
   219.00    -2.3792     1.0000
   240.00    -2.3792     1.0000
```