

NAG Fortran Library Routine Document

D02TZF

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

D02TZF returns information about the solution of a general two-point boundary value problem computed by D02TKF.

2 Specification

```

SUBROUTINE D02TZF(MXMESH, NMESH, MESH, IPMESH, ERMX, IERMX, IJERMX,
1              RWORK, IWORK, IFAIL)
INTEGER          MXMESH, NMESH, IPMESH(MXMESH), IERMX, IJERMX,
1              IWORK(*), IFAIL
real           MESH(MXMESH), ERMX, RWORK(*)

```

3 Description

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

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

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 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

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

and the right boundary conditions at b as

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

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

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

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.

The routines are based on modified versions of the codes COLSYS and COLNEW (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, Englewood Cliffs, NJ

Cole J D (1968) *Perturbation Methods in Applied Mathematics* Blaisdell, Waltham, Mass.

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 *Output*

On exit: the number of points in the mesh last used by D02TKF.

3: MESH(MXMESH) – *real* array *Output*

On exit: MESH(*i*) contains the *i*th point of the mesh last used by D02TKF, for $i = 1, 2, \dots, NMESH$. MESH(1) will contain *a* and MESH(NMESH) will contain *b*. The remaining elements of MESH are not initialised.

4: IPMESH(MXMESH) – INTEGER array *Output*

On exit: IPMESH(*i*) specifies the nature of the point MESH(*i*), $i = 1, 2, \dots, NMESH$, in the final mesh computed by D02TKF.

IPMESH(*i*) = 1

Indicates that the *i*th point is a fixed point and was used by the solver prior to an extrapolation-like error test.

IPMESH(*i*) = 2

Indicates that the *i*th point was used by the solver prior to an extrapolation-like error test.

IPMESH(*i*) = 3

Indicates that the *i*th point was used by the solver only as part of an extrapolation-like error test.

The remaining elements of IPMESH are initialised to -1 .

See Section 8 for advice on how these values may be used in conjunction with a continuation process.

5: ERMX – *real* *Output*

On exit: an estimate of the maximum error in the solution computed by D02TKF, that is

$$ERMX = \max \frac{\|y_i - v_i\|}{(1.0 + \|v_i\|)}$$

where v_i is the approximate solution for the *i*th solution component. If D02TKF returned successfully with IFAIL=0, then ERMX will be less than TOLS(IJERMX) where TOLS contains the error requirements as specified in Section 3 of the document for D02TVF and Section 5 of the document for D02TVF.

If D02TKF returned with IFAIL=5, then ERMX will be greater than TOLS(IJERMX).

If D02TKF returned any other value for IFAIL then an error estimate is not available and ERMX is initialised to 0.0.

- 6: IERMx – INTEGER *Output*
On exit: indicates the mesh sub-interval where the value of ERMx has been computed, that is [MESH(IERMx), MESH(IERMx + 1)].
 If an estimate of the error is not available then IERMx is initialised to 0.
- 7: IJERMx – INTEGER *Output*
On exit: indicates the component i (= IJERMx) of the solution for which ERMx has been computed, that is the approximation of y_i on [MESH(IERMx), MESH(IERMx + 1)] is estimated to have the largest error of all components y_i over mesh sub-intervals defined by MESH.
 If an estimate of the error is not available then IJERMx is initialised to 0.
- 8: 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.
- 9: 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.
- 10: IFAIL – INTEGER *Input/Output*
On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.
On exit: IFAIL = 0 unless the routine detects an error (see Section 6).
 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, because for this routine the values of the output parameters may be useful even if IFAIL \neq 0 on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

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, an illegal value for MXMESH was specified, or an invalid call to D02TZF was made, for example without a previous call to the solver routine D02TKF. 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).

IFAIL = 2

The solver routine D02TKF did not converge to a solution or did not satisfy the error requirements. The last mesh computed by D02TKF has been returned by D02TZF. This mesh should be treated with extreme caution as nothing can be said regarding its quality or suitability for any subsequent computation.

7 Accuracy

Not applicable.

8 Further Comments

Note that

if D02TKF returned $IFAIL = 0, 4$ or 5 then it will always be the case that $IPMESH(1) = IPMESH(NMESH) = 1$;

if D02TKF returned $IFAIL = 0$ or 5 then it will always be the case that $IPMESH(i) = 3$, for $i = 2, 4, \dots, NMESH - 1$ and $IPMESH(i) = 1$ or 2 , for $i = 3, 5, \dots, NMESH - 2$.

if D02TKF returned $IFAIL = 4$ then it will always be the case that $IPMESH(i) = 1$ or 2 , for $i = 2, 3, \dots, NMESH - 1$.

If D02TZF returns the value $IFAIL=0$, then examination of the mesh may provide assistance in determining a suitable starting mesh for D02TVF in any subsequent attempts to solve similar problems.

If the problem being treated by D02TKF is one of a series of related problems (for example, as part of a continuation process), then the values of $IPMESH$ and $MESH$ may be suitable as input parameters to D02TXF. Using the mesh points not involved in the extrapolation error test is usually appropriate. $IPMESH$ and $MESH$ should be passed unchanged to D02TXF but $NMESH$ should be replaced by $(NMESH + 1)/2$.

If D02TZF returns the value $IFAIL=2$, nothing can be said regarding the quality of the mesh returned. However, it may be a useful starting mesh for D02TVF in any subsequent attempts to solve the same problem.

If D02TKF returns the value $IFAIL=5$, this corresponds to the solver requiring more than $MXMESH$ mesh points to satisfy the error requirements. If $MXMESH$ can be increased and the preceding call to D02TKF was not part, or was the first part, of a continuation process then the values in $MESH$ may provide a suitable mesh with which to initialise a subsequent attempt to solve the same problem. If it is not possible to provide more mesh points then relaxing the error requirements by setting $TOL(IJERMX)$ to $ERMx$ might lead to a successful solution. It may be necessary to reset the other components of TOL . Note that resetting the tolerances can lead to a different sequence of meshes being computed and hence to a different solution being computed.

9 Example

The following example is used to illustrate the use of fixed mesh points, simple continuation and numerical approximation of a Jacobian. See also D02TKF, D02TVF, D02TXF and D02TYF, for the illustration of other facilities.

Consider the Lagerstrom–Cole equation

$$y'' = (y - yy')/\epsilon$$

with the boundary conditions

$$y(0) = \alpha \quad y(1) = \beta, \tag{1}$$

where ϵ is small and positive. The nature of the solution depends markedly on the values of α, β . See Cole (1968).

We choose $\alpha = -\frac{1}{3}, \beta = \frac{1}{3}$ for which the solution is known to have corner layers at $x = \frac{1}{3}, \frac{2}{3}$. We choose an initial mesh of seven points $[0.0, 0.15, 0.3, 0.5, 0.7, 0.85, 1.0]$ and ensure that the points $x = 0.3, 0.7$ near the corner layers are fixed, that is the corresponding elements of the array $IPMESH$ are set to 1. First we compute the solution for $\epsilon = 1.0E-4$ using in GUESS the initial approximation $y(x) = \alpha + (\beta - \alpha)x$ which satisfies the boundary conditions. Then we use simple continuation to compute the solution for $\epsilon = 1.0E-5$. We use the suggested values for $NMESH, IPMESH$ and $MESH$ in the call to D02TXF prior to the continuation call, that is only every second point of the preceding mesh is used and the fixed mesh points are retained.

Although the analytic Jacobian for this system is easy to evaluate, for illustration the procedure FJAC uses central differences and calls to FFUN to compute a numerical approximation to the Jacobian.

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.

```

*      D02TZF Example Program Text
*      Mark 17 Release. NAG Copyright 1995.
*      .. Parameters ..
INTEGER      NOUT
PARAMETER    (NOUT=6)
INTEGER      NEQ, MMAX, NLBC, NRBC, NCOL, MXMESH
PARAMETER    (NEQ=1,MMAX=2,NLBC=1,NRBC=1,NCOL=5,MXMESH=50)
INTEGER      LRWORK, LIWORK
PARAMETER    (LRWORK=MXMESH*(109*NEQ**2+78*NEQ+7),
+            LIWORK=MXMESH*(11*NEQ+6))
*      .. Scalars in Common ..
real       ALPHA, BETA, EPS
*      .. Local Scalars ..
real       ERMX
INTEGER      I, IERMX, IFAIL, IJERMX, J, 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
*      .. Common blocks ..
COMMON      /PROBS/ EPS, ALPHA, BETA
*      .. Executable Statements ..
WRITE (NOUT,*) 'D02TZF Example Program Results'
WRITE (NOUT,*)
NMESH = 7
MESH(1) = 0.0e0
MESH(2) = 0.15e0
MESH(3) = 0.3e0
MESH(4) = 0.5e0
MESH(5) = 0.7e0
MESH(6) = 0.85e0
MESH(NMESH) = 1.0e0
IPMESH(1) = 1
IPMESH(2) = 2
IPMESH(3) = 1
IPMESH(4) = 2
IPMESH(5) = 1
IPMESH(6) = 2
IPMESH(NMESH) = 1
ALPHA = -1.0e0/3.0e0
BETA = 1.0e0/3.0e0
TOL(1) = 1.0e-5
EPS = 1.0e-3
M(1) = 2
IFAIL = 0
CALL D02TVF(NEQ,M,NLBC,NRBC,NCOL,TOL,MXMESH,NMESH,MESH,IPMESH,
+          WORK,LRWORK,IWORK,LIWORK,IFAIL)
IFAIL = -1
DO 40 J = 1, 2
  EPS = 0.1e0*EPS
  WRITE (NOUT,99997) TOL(1), EPS
  IFAIL = -1
  CALL D02TKF(FFUN,FJAC,GAFUN,GBFUN,GAJAC,GBJAC,GUESS,WORK,IWORK,
+           IFAIL)
  FAILED = IFAIL .NE. 0
  IFAIL = 0
  CALL D02TZF(MXMESH,NMESH,MESH,IPMESH,ERMX,IERMX,IJERMX,WORK,
+           IWORK,IFAIL)
  WRITE (NOUT,99996) NMESH, ERMX, IERMX, IJERMX
  IF (FAILED) GO TO 60
  WRITE (NOUT,99999)

```

```

      DO 20 I = 1, NMESH, 2
        CALL D02TYF(MESH(I), Y, NEQ, MMAX, WORK, IWORK, IFAIL)
        WRITE (NOUT, 99998) MESH(I), Y(1,0), Y(1,1)
20     CONTINUE
      IF (J.LT.2) THEN
        NMESH = (NMESH+1)/2
        CALL D02TXF(MXMESS, NMESH, MESH, IPMESS, WORK, IWORK, IFAIL)
      END IF
40     CONTINUE
60     CONTINUE
      STOP

*
99999  FORMAT ('/' Solution and derivative at every second point:',
+           '/' ' ' x ' ' u ' ')
99998  FORMAT (' ', F8.3, 2F11.5)
99997  FORMAT ('/' Tolerance = ', e8.1,' EPS = ', e10.3)
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 ALPHA, BETA, EPS
*      .. Common blocks ..
      COMMON /PROBS/ EPS, ALPHA, BETA
*      .. Executable Statements ..
      F(1) = (Y(1,0) - Y(1,0)*Y(1,1))/EPS
      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 ALPHA, BETA, EPS
*      .. Local Scalars ..
      real FAC, MACHEP, PTRB
      INTEGER I, J, K
*      .. Local Arrays ..
      real F1(1), F2(1), YP(1, 0:3)
*      .. External Functions ..
      real X02AJF
      EXTERNAL X02AJF
*      .. External Subroutines ..
      EXTERNAL FFUN
*      .. Intrinsic Functions ..
      INTRINSIC ABS, MAX, SQRT
*      .. Common blocks ..
      COMMON /PROBS/ EPS, ALPHA, BETA
*      .. Executable Statements ..
      MACHEP = X02AJF()
      FAC = SQRT(MACHEP)
      DO 40 I = 1, NEQ
        DO 20 J = 0, M(I) - 1
          YP(I, J) = Y(I, J)
20     CONTINUE
40     CONTINUE
      DO 100 I = 1, NEQ
        DO 80 J = 0, M(I) - 1
          PTRB = MAX(1.0e2*MACHEP, FAC*ABS(Y(I, J)))
          YP(I, J) = Y(I, J) + PTRB
          CALL FFUN(X, YP, NEQ, M, F1)
          YP(I, J) = Y(I, J) - PTRB

```

```

        CALL FFUN(X,YP,NEQ,M,F2)
        DO 60 K = 1, NEQ
            DF(K,I,J) = 0.5e0*(F1(K)-F2(K))/PTRB
60      CONTINUE
        YP(I,J) = Y(I,J)
80      CONTINUE
100     CONTINUE
        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)
*      .. Scalars in Common ..
real              ALPHA, BETA, EPS
*      .. Common blocks ..
        COMMON      /PROBS/EPS, ALPHA, BETA
*      .. Executable Statements ..
        GA(1) = YA(1,0) - ALPHA
        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)
*      .. Scalars in Common ..
real              ALPHA, BETA, EPS
*      .. Common blocks ..
        COMMON      /PROBS/EPS, ALPHA, BETA
*      .. Executable Statements ..
        GB(1) = YB(1,0) - BETA
        RETURN
        END
        SUBROUTINE GAJAC(YA,NEQ,M,NLBC,DGA)
*      .. Parameters ..
real              ONE
        PARAMETER    (ONE=1.0e+0)
*      .. Scalar Arguments ..
        INTEGER      NEQ, NLBC
*      .. Array Arguments ..
real              DGA(NLBC,NEQ,0:*), YA(NEQ,0:*)
        INTEGER      M(NEQ)
*      .. Executable Statements ..
        DGA(1,1,0) = ONE
        RETURN
        END
        SUBROUTINE GBJAC(YB,NEQ,M,NRBC,DGB)
*      .. Parameters ..
real              ONE
        PARAMETER    (ONE=1.0e+0)
*      .. Scalar Arguments ..
        INTEGER      NEQ, NRBC
*      .. Array Arguments ..
real              DGB(NRBC,NEQ,0:*), YB(NEQ,0:*)
        INTEGER      M(NEQ)
*      .. Executable Statements ..
        DGB(1,1,0) = ONE
        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                ALPHA, BETA, EPS
*      .. Common blocks ..
COMMON                /PROBS/EPS, ALPHA, BETA
*      .. Executable Statements ..
      Z(1,0) = ALPHA + (BETA-ALPHA)*X
      Z(1,1) = (BETA-ALPHA)
      DMVAL(1) = 0.0e0
      RETURN
      END

```

9.2 Program Data

None.

9.3 Program Results

D02TZF Example Program Results

Tolerance = 0.1E-04 EPS = 0.100E-03

Used a mesh of 25 points
 Maximum error = 0.21E-05 in interval 16 for component 1

Solution and derivative at every second point:

x	u	u'
0.000	-0.33333	1.00000
0.075	-0.25833	1.00000
0.150	-0.18333	1.00000
0.225	-0.10833	1.00002
0.300	-0.03332	1.00372
0.400	-0.00001	0.00084
0.500	-0.00000	0.00000
0.600	0.00001	0.00084
0.700	0.03332	1.00372
0.775	0.10833	1.00002
0.850	0.18333	1.00000
0.925	0.25833	1.00000
1.000	0.33333	1.00000

Tolerance = 0.1E-04 EPS = 0.100E-04

Used a mesh of 49 points
 Maximum error = 0.21E-05 in interval 32 for component 1

Solution and derivative at every second point:

x	u	u'
0.000	-0.33333	1.00014
0.037	-0.29583	1.00018
0.075	-0.25833	1.00022
0.112	-0.22083	1.00029
0.150	-0.18333	1.00040
0.188	-0.14583	1.00059
0.225	-0.10833	1.00098
0.262	-0.07083	1.00202
0.300	-0.03333	1.00745
0.350	-0.00001	0.00354
0.400	-0.00000	0.00000
0.450	-0.00000	0.00000
0.500	-0.00000	0.00000
0.550	0.00000	0.00000
0.600	0.00000	0.00000
0.650	0.00001	0.00354
0.700	0.03333	1.00745
0.737	0.07083	1.00202
0.775	0.10833	1.00098
0.812	0.14583	1.00059
0.850	0.18333	1.00040

0.887	0.22083	1.00029
0.925	0.25833	1.00022
0.963	0.29583	1.00018
1.000	0.33333	1.00014
