

## D02TYF – 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

D02TYF interpolates on the solution of a general two point boundary value problem computed by D02TKF.

### 2 Specification

```
SUBROUTINE D02TYF(X, Y, NEQ, MMAX, RWORK, IWORK, IFAIL)
INTEGER          NEQ, MMAX, IWORK(*), IFAIL
real             X, Y(NEQ,0:MMAX-1), RWORK(*)
```

### 3 Description

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

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

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 and other details of the solution procedure, and 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, [2] and [1]. A comprehensive treatment of the numerical solution of boundary value problems can be found in [3] and [5].

### 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] Grossman C (1992) Enclosures of the solution of the Thomas–Fermi equation by monotone discretization *J. Comput. Phys.* **98** 26–32
- [5] Keller H B (1992) *Numerical Methods for Two-point Boundary-value Problems* Dover, New York

## 5 Parameters

- 1: X — *real* *Input*  
*On entry:* the independent variable,  $x$ .  
*Constraint:*  $a \leq X \leq b$ .
- 2: Y(NEQ,0:MMAx-1) — *real* array *Output*  
*On exit:* Y( $i, j$ ) contains an approximation to  $y_i^{(j)}(x)$ , for  $i = 1, 2, \dots, \text{NEQ}$ ,  $j = 0, 1, \dots, m_i - 1$ . The remaining elements of Y (where  $m_i < \text{MMAx}$ ) are initialized to 0.0.
- 3: NEQ — INTEGER *Input*  
*On entry:* the number of differential equations.  
*Constraint:* NEQ must be the same value as supplied to D02TVF.
- 4: MMAx — INTEGER *Input*  
*On entry:* the maximal order of the differential equations,  $\max(m_i)$ , for  $i = 1, 2, \dots, \text{NEQ}$ .  
*Constraint:* MMAx must contain the maximum value of the components of the argument M as supplied to D02TVF.
- 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. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.  
*On exit:* IFAIL = 0 unless the routine detects an error or gives a warning (see Section 6).

**For this routine**, because the values of output parameters may be useful even if IFAIL  $\neq$  0 on exit, users are recommended to set IFAIL to -1 before entry. **It is then essential to test the value of IFAIL on exit.** To suppress the output of an error message when soft failure occurs, set IFAIL to 1.

## 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 or warnings specified by the routine:

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 D02TYF 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 solution computed by D02TKF, for which convergence was obtained, has been used for interpolation by D02TYF. The results returned by D02TYF should be treated with extreme caution as regarding either their quality or accuracy. See Section 8.

## 7 Accuracy

If D02TYF returns the value IFAIL = 0, the computed values of the solution components  $y_i$  should be of similar accuracy to that specified by the argument TOLS of D02TVF. Note that during the solution process the error in the derivatives  $y_i^{(j)}$ ,  $j = 1, 2, \dots, m_i - 1$  has not been controlled and that the derivative values returned by D02TYF are computed via differentiation of the piecewise polynomial approximation to  $y_i$ . See also Section 8.

## 8 Further Comments

If D02TYF returns the value IFAIL = 2, and the solver routine D02TKF returned IFAIL = 5, then the accuracy of the interpolated values may be proportional to the quantity ERMX as returned by D02TZF.

If D02TKF returned any other non-zero value for IFAIL, then nothing can be said regarding either the quality or accuracy of the values computed by D02TYF.

## 9 Example

The following example is used to illustrate that a system with singular coefficients can be treated without modification of the system definition. See also D02TKF, D02TVF, D02TXF and D02TZF, for the illustration of other facilities.

Consider the Thomas–Fermi equation used in the investigation of potentials and charge densities of ionized atoms. See [4], for example, and the references therein. The equation is

$$y'' = x^{-1/2}y^{3/2}$$

with boundary conditions

$$y(0) = 1, \quad y(a) = 0, \quad a > 0.$$

The coefficient  $x^{-1/2}$  implies a singularity at the left hand boundary  $x = 0$ .

We use the initial approximation  $y(x) = 1 - x/a$ , which satisfies the boundary conditions, on a uniform mesh of six points. For illustration we choose  $a = 1$ , as in [4]. Note that in the subroutines FFUN and FJAC we have taken the precaution of setting the function value and Jacobian value to 0.0 in case a value of  $y$  becomes negative, although starting from our initial solution profile this proves unnecessary during the solution phase. Of course the true solution  $y(x)$  is positive for all  $x < a$ .

## 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.

```

*   D02TYF 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=4,MXMESH=100)
INTEGER         LRWORK, LIWORK
PARAMETER       (LRWORK=MXMESH*(109*NEQ**2+78*NEQ+7),
+              LIWORK=MXMESH*(11*NEQ+6))
*   .. Scalars in Common ..
  real          A
*   .. Local Scalars ..
  real          AINC, ERMX, XX
INTEGER        I, IERMX, IFAIL, IJERMX, 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, D02TYF, D02TZF, FFUN, FJAC,
+              GAFUN, GAJAC, GBFUN, GBJAC, GUESS
*   .. Intrinsic Functions ..
INTRINSIC     real
*   .. Common blocks ..
COMMON        /PROBS/A
*   .. Executable Statements ..
WRITE (NOUT,*) 'D02TYF Example Program Results'
WRITE (NOUT,*)
A = 1.0e0
NMESH = 6
AINC = A/real(NMESH-1)
MESH(1) = 0.0e0
IPMESH(1) = 1
DO 20 I = 2, NMESH - 1
    MESH(I) = real(I-1)*AINC
    IPMESH(I) = 2
20 CONTINUE
MESH(NMESH) = A
IPMESH(NMESH) = 1
TOL(1) = 1.0e-5
M(1) = 2
IFAIL = 0
CALL D02TVF(NEQ,M,NLBC,NRBC,NCOL,TOL,MXMESH,NMESH,MESH,IPMESH,
+          WORK,LRWORK,IWORK,LIWORK,IFAIL)
WRITE (NOUT,99997) TOL(1), A
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,
+ (I,IPMESH(I),MESH(I),I=1,NMESH)
IF ( .NOT. FAILED) THEN
  AINC = 0.1e0*A
  WRITE (NOUT,99999)
  DO 40 I = 1, 11
    XX = real(I-1)*AINC
    CALL D02TYF(XX,Y,NEQ,MMAX,WORK,IWORK,IFAIL)
    WRITE (NOUT,99998) XX, Y(1,0), Y(1,1)
40  CONTINUE
  END IF
  STOP
*
99999 FORMAT (' Computed solution',/ ' x solution derivati',
+ 've')
99998 FORMAT (' ',F8.2,2F11.5)
99997 FORMAT (// ' Tolerance = ',e8.1, ' A = ',F8.2)
99996 FORMAT (' Used a mesh of ',I4,' points',/ ' Maximum error = ',
+ e10.2, ' in interval ',I4,' for component ',I4,/' Mesh p',
+ 'oints:',/4(I4,'(',I1,')',e11.4))
  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)
* .. Intrinsic Functions ..
  INTRINSIC SQRT
* .. Executable Statements ..
  IF (Y(1,0).LE.0.0e0) THEN
    F(1) = 0.0e0
    PRINT *, ' F'
  ELSE
    F(1) = (Y(1,0))**1.5e0/SQRT(X)
  END IF
  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)
* .. Intrinsic Functions ..
  INTRINSIC SQRT
* .. Executable Statements ..
  IF (Y(1,0).LE.0.0e0) THEN
    DF(1,1,0) = 0.0e0
    PRINT *, ' JAC'
  ELSE
    DF(1,1,0) = 1.5e0*SQRT(Y(1,0))/SQRT(X)
  END IF
  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) - 1.0e0
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(1,0)
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
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(1,1,0) = 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           A
*      .. Common blocks ..
COMMON          /PROBS/A
*      .. Executable Statements ..
Z(1,0) = 1.0e0 - X/A
Z(1,1) = -1.0e0/A
DMVAL(1) = 0.0e0
RETURN
END

```

## 9.2 Program Data

None.

## 9.3 Program Results

D02TYF Example Program Results

Tolerance = 0.1E-04 A = 1.00

Used a mesh of 11 points

Maximum error = 0.31E-05 in interval 1 for component 1

Mesh points:

1(1)	0.0000E+00	2(3)	0.1000E+00	3(2)	0.2000E+00	4(3)	0.3000E+00
5(2)	0.4000E+00	6(3)	0.5000E+00	7(2)	0.6000E+00	8(3)	0.7000E+00
9(2)	0.8000E+00	10(3)	0.9000E+00	11(1)	0.1000E+01		

Computed solution

x	solution	derivative
0.00	1.00000	-1.84496
0.10	0.84944	-1.32330
0.20	0.72721	-1.13911
0.30	0.61927	-1.02776
0.40	0.52040	-0.95468
0.50	0.42754	-0.90583
0.60	0.33867	-0.87372
0.70	0.25239	-0.85369
0.80	0.16764	-0.84248
0.90	0.08368	-0.83756
1.00	0.00000	-0.83655

---