D02TVF – 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

D02TVF is a setup routine which must be called prior to the first call of the nonlinear two point boundary value solver D02TKF.

2 Specification

```
 SUBROUTINE D02TVF(NEQ, M, NLBC, NRBC, NCOL, TOLS, MXMESH, NMESH,
1                  MESH, IPMESH, RWORK, LRWORK, IWORK, LIWORK, IFAIL)
 INTEGER           NEQ, M(NEQ), NLBC, NRBC, NCOL, MXMESH, NMESH,
1                  IPMESH(MXMESH), LRWORK, IWORK(LIWORK), LIWORK,
2                  IFAIL
 real              TOLS(NEQ), MESH(MXMESH), RWORK(LRWORK)
```

3 Description

D02TVF and its associated routines (D02TKF, D02TXF, 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 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

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

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

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

See Section 8 for information on how boundary value problems of a more general nature can be treated.

D02TVF is used to specify an initial mesh, error requirements and other details. D02TKF is then used to solve the boundary value problem.

The solution routine D02TKF proceeds as follows. A modified Newton method is applied to the equations

$$y_i^{(m_i)}(x) - f_i(x, z(y(x))) = 0, \quad i = 1, \ldots, n$$

and the boundary conditions. To solve these equations numerically the components y_i are approximated by piecewise polynomials v_{ij} using a monomial basis on the jth mesh sub-interval. The coefficients of the polynomials v_{ij} form the unknowns to be computed. Collocation is applied at Gaussian points

$$v_{ij}^{(m_i)}(x_{jk}) - f_i(x_{jk}, z(v(x_{jk}))) = 0, \quad i = 1, \ldots, n,$$

where x_{jk} is the kth collocation point in the jth mesh sub-interval. Continuity at the mesh points is imposed, that is

$$v_{ij}(x_{j+1}) - v_{i,j+1}(x_{j+1}) = 0, \quad i = 1, 2, \ldots, n,$$

where x_{j+1} is the right hand end of the jth mesh sub-interval. The linearized collocation equations and boundary conditions, together with the continuity conditions form a system of linear algebraic equations which are solved using F01LHF and F04LHF. For use in the modified Newton method, an approximation to the solution on the initial mesh must be supplied via the procedure argument GUESS of D02TKF.

The solver attempts to satisfy the conditions

$$\frac{\|y_i - v_i\|}{(1.0 + \|v_i\|)} \leq \text{TOLS}(i), \quad i = 1, 2, \ldots, n, \tag{1}$$

where v_i is the approximate solution for the ith solution component and TOLS is supplied by the user. The mesh is refined by trying to equidistribute the estimated error in the computed solution over all mesh sub-intervals, and an extrapolation-like test (doubling the number of mesh sub-intervals) is used to check for (1).

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] Gill P E, Murray W and Wright M H (1981) *Practical Optimization* Academic Press

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

[6] Schwartz I B (1983) Estimating regions of existence of unstable periodic orbits using computer-based techniques *SIAM J. Sci. Statist. Comput.* **20(1)** 106–120

5 Parameters

1: NEQ — INTEGER *Input*

On entry: the number of ordinary differential equations to be solved, n.

Constraint: NEQ ≥ 1.

2: M(NEQ) — INTEGER array *Input*

On entry: the order, m_i, of the ith differential equation, for $i = 1, 2, \ldots, n$.

Constraint: $1 \leq \text{M}(i) \leq 4$, $i = 1, 2, \ldots, n$.

3: NLBC — INTEGER *Input*

On entry: the number of left boundary conditions, p, defined at the left hand end, a (= MESH(1)).

Constraint: NLBC ≥ 1.

4: NRBC — INTEGER *Input*

On entry: the number of right boundary conditions, q, defined at the right hand end, b (= MESH(NMESH)).

Constraints:

$$\text{NRBC} \geq 1,$$
$$\text{NLBC} + \text{NRBC} = \sum_{1}^{n} \text{M}(i).$$

5: NCOL — INTEGER *Input*

On entry: the number of collocation points to be used in each mesh sub-interval.

Constraint: $m_{\max} \leq \text{NCOL} \leq 7$, where $m_{\max} = \max(\text{M}(i))$.

6: TOLS(NEQ) — ***real*** array *Input*

On entry: the error requirement for the ith solution component.

Constraint: $100 \times$ ***machine precision*** $< \text{TOLS}(i) < 1.0$, for $i = 1, 2, \ldots, n$.

7: MXMESH — INTEGER *Input*

On entry: the maximum number of mesh points to be used during the solution process.

Constraint: $\text{MXMESH} \geq 2 \times \text{NMESH} - 1$.

8: NMESH — INTEGER *Input*

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

Constraint: $\text{NMESH} \geq 6$.

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

On entry: the positions of the initial NMESH mesh points. The remaining elements of MESH need not be set. You should try to place the mesh points in areas where you expect the solution to vary most rapidly. In the absence of any other information the points should be equally distributed on $[a, b]$.

MESH(1) must contain the left boundary point, a, and MESH(NMESH) must contain the right boundary point, b.

Constraint: $\text{MESH}(i) < \text{MESH}(i+1)$, for $i = 1, 2, \ldots,$NMESH-1.

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

On entry: IPMESH(i) specifies whether or not the initial mesh point defined in MESH(i), $i = 1, \ldots,$NMESH, should be a fixed point in all meshes computed during the solution process. The remaining elements of IPMESH need not be set.

IPMESH(i) = 1 indicates that MESH(i) should be a fixed point in all meshes.

IPMESH(i) = 2 indicates that MESH(i) is not a fixed point.

Constraints:

IPMESH(1) = 1 and IPMESH(NMESH) = 1, (that is the left and right boundary points, a and b, must be fixed points, in all meshes)
IPMESH(i) = 1 or 2, $i = 2, 3, \ldots,$NMESH-1.

11: RWORK(LRWORK) — ***real*** array *Output*

On exit: contains information for use by D02TKF. This **must** be the same array as will be supplied to D02TKF. The contents of this array **must** remain unchanged between calls.

12: LRWORK — INTEGER *Input*

On entry: the dimension of the array RWORK as declared in the (sub)program from which D02TVF is called.

Suggested value: LRWORK = MXMESH × (109 × N^2 + 78 × N + 7), which will permit MXMESH mesh points for a system of N differential equations regardless of their order or the number of collocation points used.

Constraint: LRWORK ≥ 50 + NEQ × ($m_{\max}$ × (1+NEQ+max(NLBC,NRBC)) + 6) $-k_n \times (k_n +$ 6) $- m^* \times (k_n + m^* - 2)+$ MXMESH $\times((m^*+3)(2m^*+3) - 3 + k_n(k_n + m^* + 6))+$MXMESH/2, where $m^* = \sum_1^n \mathrm{M}(i)$ and k_n = NCOL × NEQ.

13: IWORK(LIWORK) — INTEGER array *Output*

On exit: contains information for use by D02TKF. This **must** be the same array as will be supplied to D02TKF. The contents of this array **must** remain unchanged between calls.

14: LIWORK — INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which D02TVF is called.

Suggested value: LIWORK = MXMESH × (11×N+6), which will permit MXMESH mesh points for a system of N differential equations regardless of their order or the number of collocation points used.

Constraint: LIWORK ≥ 23 + 3 ×NEQ $- k_n +$ MXMESH× $(m^* + k_n + 4)$.

15: 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

On entry, NEQ < 1,
or M(i) < 1, for some i,
or M(i) > 4, for some i,
or NMESH < 6,
or the values of MESH are not strictly increasing,
or IPMESH(i) is invalid for some i,
or MXMESH < 2 × NMESH − 1,
or NCOL < $m_{\max}$, where $m_{\max}$ = max(M(i)),
or NCOL > 7,
or NLBC < 1,
or NRBC < 1,
or a value of TOLS is invalid,
or NLBC + NRBC ≠ $\sum_1^n \mathrm{M}(i)$,
or LRWORK or LIWORK is too small.

7 Accuracy

Not applicable.

8 Further Comments

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

use a large value for NCOL;

cluster the initial 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 before they are detected automatically by the algorithm.

8.1 Nonseparated boundary conditions

A boundary value problem with nonseparated boundary conditions can be treated by transformation to an equivalent problem with separated conditions. As a simple example consider the system

$$\begin{aligned} y_1' &= f_1(x, y_1, y_2) \\ y_2' &= f_2(x, y_1, y_2) \end{aligned}$$

on $[a, b]$ subject to the boundary conditions

$$\begin{aligned} g_1(y_1(a)) &= 0 \\ g_2(y_2(a), y_2(b)) &= 0. \end{aligned}$$

By adjoining the trivial ordinary differential equation

$$r' = 0,$$

which implies $r(a) = r(b)$, and letting $r(b) = y_2(b)$, say, we have a new system

$$\begin{aligned} y_1' &= f_1(x, y_1, y_2) \\ y_2' &= f_2(x, y_1, y_2) \\ r' &= 0, \end{aligned}$$

subject to the separated boundary conditions

$$\begin{aligned} g_1(y_1(a)) &= 0 \\ g_2(y_2(a), r(a)) &= 0 \\ y_2(b) - r(b) &= 0. \end{aligned}$$

There is an obvious overhead in adjoining an extra differential equation: the system to be solved is increased in size.

8.2 Multipoint boundary value problems

Multipoint boundary value problems, that is problems where conditions are specified at more than two points, can also be transformed to an equivalent problem with two boundary points. Each sub-interval defined by the multipoint conditions can be transformed onto the interval $[0, 1]$, say, leading to a larger set of differential equations. The boundary conditions of the transformed system consist of the original boundary conditions and the conditions imposed by the requirement that the solution components be continuous at the interior break points. For example, consider the equation

$$y^{(3)} = f(t, y, y^{(1)}, y^{(2)}) \quad \text{on} \quad [a, c]$$

subject to the conditions

$$\begin{aligned} y(a) &= A \\ y(b) &= B \\ y^{(1)}(c) &= C \end{aligned}$$

where $a < b < c$. This can be transformed to the system

$$\left.\begin{array}{rcl} y_1^{(3)} & = & f(t, y_1, y_1^{(1)}, y_1^{(2)}) \\ y_2^{(3)} & = & f(t, y_2, y_2^{(1)}, y_2^{(2)}) \end{array}\right\} \quad \text{on } [0,1]$$

where

$$\begin{array}{rcl} y_1 & \equiv & y \quad \text{on} \quad [a,b] \\ y_2 & \equiv & y \quad \text{on} \quad [b,c], \end{array}$$

subject to the boundary conditions

$$\begin{array}{rcll} y_1(0) & = & A & \\ y_1(1) & = & B & \\ y_2^{(1)}(1) & = & C & \\ y_2(0) & = & B & (\text{from } y_1(1) = y_2(0)) \\ y_1^{(1)}(1) & = & y_2^{(1)}(0) & \\ y_1^{(2)}(1) & = & y_2^{(2)}(0). & \end{array}$$

In this instance two of the resulting boundary conditions are nonseparated but they may next be treated as described above.

8.3 High order systems

Systems of ordinary differential equations containing derivatives of order greater than four can always be reduced to systems of order suitable for treatment by D02TVF and its related routines. For example suppose we have the sixth order equation

$$y^{(6)} = -y.$$

Writing the variables $y_1 = y$ and $y_2 = y^{(4)}$ we obtain the system

$$\begin{array}{rcl} y_1^{(4)} & = & y_2 \\ y_2^{(2)} & = & -y_1 \end{array}$$

which has maximal order four, or writing the variables $y_1 = y$ and $y_2 = y^{(3)}$ we obtain the system

$$\begin{array}{rcl} y_1^{(3)} & = & y_2 \\ y_2^{(3)} & = & -y_1 \end{array}$$

which has maximal order three. The best choice of reduction by choosing new variables will depend on the structure and physical meaning of the system. Note that you will control the error in each of the variables y_1 and y_2. Indeed, if you wish to control the error in certain derivatives of the solution of an equation of order greater than one, then you should make those derivatives new variables.

8.4 Fixed points and singularities

The solver routine D02TKF employs collocation at Gaussian points in each sub-interval of the mesh. Hence the coefficients of the differential equations are not evaluated at the mesh points. Thus, fixed points should be specified in the mesh where either the coefficients are singular, or the solution has less smoothness, or where the differential equations should not be evaluated. Singular coefficients at boundary points often arise when physical symmetry is used to reduce partial differential equations to ordinary differential equations. These do not pose a direct numerical problem for using this code but they can severely impact its convergence.

8.5 Numerical Jacobians

The solver routine D02TKF requires an external procedure FJAC to evaluate the partial derivatives of f_i with respect to the elements of $z(y)$ $(= (y_1, y_1^1, \ldots, y_1^{(m_1-1)}, y_2, \ldots, y_n^{(m_n-1)}))$. In cases where the partial derivatives are difficult to evaluate, numerical approximations can be used. However, this approach might have a negative impact on the convergence of the modified Newton method. You could consider the use of symbolic mathematic packages and/or automatic differentiation packages if available to you.

See Section 9 of the document for D02TZF for an example using numerical approximations to the Jacobian. There central differences are used and each f_i is assumed to depend on all the components of z. This requires two evaluations of the system of differential equations for each component of z. The perturbation used depends on the size of each component of z and a minimum quantity dependent on the machine precision. The cost of this approach could be reduced by employing an alternative difference scheme and/or by only perturbing the components of z which appear in the definitions of the f_i. A discussion on the choice of perturbation factors for use in finite difference approximations to partial derivatives can be found in [4].

9 Example

The following example is used to illustrate the treatment of nonseparated boundary conditions. See also D02TKF, D02TXF, D02TYF and D02TZF, for the illustration of other facilities.

The following equations model of the spread of measles. See [6]. Under certain assumptions the dynamics of the model can be expressed as

$$\begin{aligned} y_1' &= \mu - \beta(x) y_1 y_3 \\ y_2' &= \beta(x) y_1 y_3 - y_2/\lambda \\ y_3' &= y_2/\lambda - y_3/\eta \end{aligned}$$

subject to the periodic boundary conditions

$$y_i(0) = y_i(1), \quad i = 1, 2, 3.$$

Here y_1, y_2 and y_3 are respectively the proportions of susceptibles, infectives and latents to the whole population. λ (= 0.0279 years) is the latent period, η (= 0.01 years) is the infectious period and μ (= 0.02) is the population birth rate. $\beta(x) = \beta_0(1.0 + \cos 2\pi x)$ is the contact rate where $\beta_0 = 1575.0$.

The nonseparated boundary conditions are treated as described in Section 8 by adjoining the trivial differential equations

$$\begin{aligned} y_4' &= 0 \\ y_5' &= 0 \\ y_6' &= 0 \end{aligned}$$

that is y_4, y_5 and y_6 are constants. The boundary conditions of the augmented system can then be posed in the separated form

$$\begin{aligned} y_1(0) - y_4(0) &= 0 \\ y_2(0) - y_5(0) &= 0 \\ y_3(0) - y_6(0) &= 0 \\ y_1(1) - y_4(1) &= 0 \\ y_2(1) - y_5(1) &= 0 \\ y_3(1) - y_6(1) &= 0. \end{aligned}$$

This is a relatively easy problem and an (arbitrary) initial guess of 1 for each component suffices, even though two components of the solution are much smaller than 1.

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.

```
*     D02TVF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NEQ, MMAX, NLBC, NRBC, NCOL, MXMESH
      PARAMETER        (NEQ=6,MMAX=1,NLBC=3,NRBC=3,NCOL=5,MXMESH=100)
      INTEGER          LRWORK, LIWORK
      PARAMETER        (LRWORK=MXMESH*(109*NEQ**2+78*NEQ+7),
     +                 LIWORK=MXMESH*(11*NEQ+6))
```

```
*     .. Scalars in Common ..
      real             BETA0, ETA, LAMBDA, MU, PI
*     .. Local Scalars ..
      real             ERMX
      INTEGER          I, IERMX, IFAIL, IJERMX, NMESH
*     .. Local Arrays ..
      real             MESH(MXMESH), RWORK(LRWORK), TOL(NEQ),
     +                 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        ATAN
*     .. Common blocks ..
      COMMON           /PROB/ETA, MU, LAMBDA, BETA0, PI
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D02TVF Example Program Results'
      WRITE (NOUT,*)
      NMESH = 11
      MESH(1) = 0.0e0
      IPMESH(1) = 1
      DO 20 I = 2, NMESH - 1
         MESH(I) = 0.1e0*(I-1)
         IPMESH(I) = 2
   20 CONTINUE
      IPMESH(NMESH) = 1
      MESH(NMESH) = 1.0e0
      DO 40 I = 1, NEQ
         TOL(I) = 1.0e-5
         M(I) = 1
   40 CONTINUE
      ETA = 0.01e0
      MU = 0.02e0
      LAMBDA = 0.0279e0
      BETA0 = 1575.0e0
      PI = 4.0e0*ATAN(1.0e0)
      IFAIL = 0
      CALL D02TVF(NEQ,M,NLBC,NRBC,NCOL,TOL,MXMESH,NMESH,MESH,IPMESH,
     +            RWORK,LRWORK,IWORK,LIWORK,IFAIL)
      IFAIL = -1
      CALL D02TKF(FFUN,FJAC,GAFUN,GBFUN,GAJAC,GBJAC,GUESS,RWORK,IWORK,
     +            IFAIL)
      CALL D02TZF(MXMESH,NMESH,MESH,IPMESH,ERMX,IERMX,IJERMX,RWORK,
     +            IWORK,IFAIL)
      WRITE (NOUT,99999) NMESH, ERMX, IERMX, IJERMX,
     +  (I,IPMESH(I),MESH(I),I=1,NMESH)
      WRITE (NOUT,99998)
      DO 60 I = 1, NMESH
         IFAIL = 1
         CALL D02TYF(MESH(I),Y,NEQ,MMAX,RWORK,IWORK,IFAIL)
         WRITE (NOUT,99997) MESH(I), Y(1,0), Y(2,0), Y(3,0)
   60 CONTINUE
      STOP
*
99999 FORMAT (/' Used a mesh of ',I4,' points',/' Maximum error = ',
     +       e10.2,'  in interval ',I4,' for component ',I4,//' Mesh p',
     +       'oints:',/4(I4,'(',I1,')',F7.4))
```

```
99998 FORMAT (/' Computed solution at mesh points',/'    x       y1    ',
     +       '      y2         y3')
99997 FORMAT (' ',F6.3,1X,3e11.3)
      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              BETA0, ETA, LAMBDA, MU, PI
*     .. Local Scalars ..
      real              BETA
*     .. Intrinsic Functions ..
      INTRINSIC         COS
*     .. Common blocks ..
      COMMON            /PROB/ETA, MU, LAMBDA, BETA0, PI
*     .. Executable Statements ..
      BETA = BETA0*(1.0e0+COS(2.0e0*PI*X))
      F(1) = MU - BETA*Y(1,0)*Y(3,0)
      F(2) = BETA*Y(1,0)*Y(3,0) - Y(2,0)/LAMBDA
      F(3) = Y(2,0)/LAMBDA - Y(3,0)/ETA
      F(4) = 0.0e0
      F(5) = 0.0e0
      F(6) = 0.0e0
      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              BETA0, ETA, LAMBDA, MU, PI
*     .. Local Scalars ..
      real              BETA
*     .. Intrinsic Functions ..
      INTRINSIC         COS
*     .. Common blocks ..
      COMMON            /PROB/ETA, MU, LAMBDA, BETA0, PI
*     .. Executable Statements ..
      BETA = BETA0*(1.0e0+COS(2.0e0*PI*X))
      DF(1,1,0) = -BETA*Y(3,0)
      DF(1,3,0) = -BETA*Y(1,0)
      DF(2,1,0) = BETA*Y(3,0)
      DF(2,2,0) = -1.0e0/LAMBDA
      DF(2,3,0) = BETA*Y(1,0)
      DF(3,2,0) = 1.0e0/LAMBDA
      DF(3,3,0) = -1.0e0/ETA
      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) - YA(4,0)
      GA(2) = YA(2,0) - YA(5,0)
      GA(3) = YA(3,0) - YA(6,0)
      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(1,0) - YB(4,0)
      GB(2) = YB(2,0) - YB(5,0)
      GB(3) = YB(3,0) - YB(6,0)
      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
      DGA(1,4,0) = -1.0e0
      DGA(2,2,0) = 1.0e0
      DGA(2,5,0) = -1.0e0
      DGA(3,3,0) = 1.0e0
      DGA(3,6,0) = -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
      DGB(1,4,0) = -1.0e0
      DGB(2,2,0) = 1.0e0
      DGB(2,5,0) = -1.0e0
      DGB(3,3,0) = 1.0e0
      DGB(3,6,0) = -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)
```

```
*     .. Local Scalars ..
      INTEGER          I
*     .. Executable Statements ..
      Z(1,0) = 1.0e0
      Z(2,0) = 1.0e0
      Z(3,0) = 1.0e0
      Z(4,0) = Z(1,0)
      Z(5,0) = Z(2,0)
      Z(6,0) = Z(3,0)
      DO 20 I = 1, NEQ
         DMVAL(I) = 0.0e0
   20 CONTINUE
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
D02TVF Example Program Results

Used a mesh of   21 points
Maximum error =   0.14E-07  in interval    5 for component    1

Mesh points:
  1(1) 0.0000   2(3) 0.0500   3(2) 0.1000   4(3) 0.1500
  5(2) 0.2000   6(3) 0.2500   7(2) 0.3000   8(3) 0.3500
  9(2) 0.4000  10(3) 0.4500  11(2) 0.5000  12(3) 0.5500
 13(2) 0.6000  14(3) 0.6500  15(2) 0.7000  16(3) 0.7500
 17(2) 0.8000  18(3) 0.8500  19(2) 0.9000  20(3) 0.9500
 21(1) 1.0000

Computed solution at mesh points
   x       y1          y2          y3
 0.000   0.752E-01  0.180E-04  0.498E-05
 0.050   0.761E-01  0.789E-04  0.219E-04
 0.100   0.766E-01  0.315E-03  0.892E-04
 0.150   0.758E-01  0.101E-02  0.298E-03
 0.200   0.726E-01  0.225E-02  0.713E-03
 0.250   0.678E-01  0.311E-02  0.108E-02
 0.300   0.641E-01  0.256E-02  0.984E-03
 0.350   0.629E-01  0.129E-02  0.550E-03
 0.400   0.633E-01  0.414E-03  0.197E-03
 0.450   0.643E-01  0.912E-04  0.478E-04
 0.500   0.653E-01  0.159E-04  0.881E-05
 0.550   0.663E-01  0.277E-05  0.151E-05
 0.600   0.673E-01  0.628E-06  0.313E-06
 0.650   0.683E-01  0.219E-06  0.964E-07
 0.700   0.693E-01  0.124E-06  0.487E-07
 0.750   0.703E-01  0.116E-06  0.409E-07
 0.800   0.713E-01  0.170E-06  0.551E-07
 0.850   0.723E-01  0.370E-06  0.113E-06
 0.900   0.733E-01  0.111E-05  0.322E-06
 0.950   0.743E-01  0.420E-05  0.118E-05
 1.000   0.752E-01  0.180E-04  0.498E-05
```