D02TKF – 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

D02TKF solves a general two point boundary value problem for a nonlinear mixed order system of ordinary differential equations.

2 Specification

```
      SUBROUTINE D02TKF(FFUN, FJAC, GAFUN, GBFUN, GAJAC, GBJAC, GUESS,
     1                  WORK, IWORK, IFAIL)
      INTEGER           IWORK(*), IFAIL
      real              WORK(*)
      EXTERNAL          FFUN, FJAC, GAFUN, GBFUN, GAJAC, GBJAC, GUESS
```

3 Description

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

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

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. Note that the error requirements apply only to the solution components $y_1, y_2, \ldots, y_n$ and that no error control is applied to derivatives of solution components. (If error control is required on derivatives then the system must be reduced in order by introducing the derivatives whose error is to be controlled as new variables. See Section 8 of the document for D02TVF.) 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 $[a, b]$.

A description of the numerical technique used in D02TKF is given in Section 3 of the document for D02TVF.

D02TKF can also be used in the solution of a series of problems, for example in performing continuation, when the mesh used to compute the solution of one problem is to be used as the initial mesh for the solution of the next related problem. D02TXF should be used in between calls to D02TKF in this context.

See Section 8 of the document for D02TVF for details of how to solve boundary value problems of a more general nature.

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: FFUN — SUBROUTINE, supplied by the user. *External Procedure*

FFUN must evaluate the functions f_i for given values $x, z(y(x))$.

Its specification is:

```
SUBROUTINE FFUN(X, Y, NEQ, M, F)
INTEGER          NEQ, M(NEQ)
real             X, Y(NEQ,0:*), F(NEQ)
```

1: X — ***real*** *Input*

On entry: the independent variable, x.

2: Y(NEQ,0:∗) — ***real*** array *Input*

On entry: Y(i,j) contains $y_i^{(j)}(x)$, for $i = 1,2,\ldots,\mathrm{NEQ}$, $j = 0,1,\ldots,\mathrm{M}(i)-1$.
Note. $y_i^{(0)}(x) = y_i(x)$.

3: NEQ — INTEGER *Input*

On entry: the number of differential equations.

4: M(NEQ) — INTEGER array *Input*

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

5: F(NEQ) — ***real*** array *Output*

On exit: the values of f_i, for $i = 1,2,\ldots,\mathrm{NEQ}$.

FFUN must be declared as EXTERNAL in the (sub)program from which D02TKF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

2: FJAC — SUBROUTINE, supplied by the user. *External Procedure*

FJAC must evaluate the partial derivatives of f_i with respect to the elements of $z(y(x))$ $(= (y_1(x), y_1^1(x), \ldots, y_1^{(m_1-1)}(x), y_2(x), \ldots, y_n^{(m_n-1)}(x)))$.

Its specification is:

```
SUBROUTINE FJAC(X, Y, NEQ, M, DFDY)
INTEGER         NEQ, M(NEQ)
real            X, Y(NEQ,0:*), DFDY(NEQ,NEQ,0:*)
```

1: X — ***real*** *Input*

On entry: the independent variable, x.

2: Y(NEQ,0:∗) — ***real*** array *Input*

On entry: Y(i,j) contains $y_i^{(j)}(x)$, for $i = 1,2,\ldots,\mathrm{NEQ}$, $j = 0,1,\ldots,\mathrm{M}(i)-1$.
Note. $y_i^{(0)}(x) = y_i(x)$.

3: NEQ — INTEGER *Input*

On entry: the number of differential equations.

4: M(NEQ) — INTEGER array *Input*

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

5: DFDY(NEQ,NEQ,0:∗) — ***real*** array *Output*

On exit: DFDY(i,j,k) must contain the partial derivative of f_i with respect to $y_j^{(k)}$, for $i,j = 1,2,\ldots,\mathrm{NEQ}$, $k = 0,1,\ldots,\mathrm{M}(j)-1$. Only non-zero partial derivatives need be set.

FJAC must be declared as EXTERNAL in the (sub)program from which D02TKF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

3: GAFUN — SUBROUTINE, supplied by the user. *External Procedure*

GAFUN must evaluate the boundary conditions at the left hand end of the range, that is functions $g_i(z(y(a)))$ for given values of $z(y(a))$.

Its specification is:

```
SUBROUTINE GAFUN(YA, NEQ, M, NLBC, GA)
INTEGER          NEQ, M(NEQ), NLBC
real             YA(NEQ,0:*), GA(NLBC)
```

1: YA(NEQ,0:∗) — ***real*** array *Input*

On entry: YA(i,j) contains $y_i^{(j)}(a)$, for $i = 1,2,\ldots,\mathrm{NEQ}$, $j = 0,1,\ldots,\mathrm{M}(i)-1$.
Note. $y_i^{(0)}(a) = y_i(a)$.

2: NEQ — INTEGER *Input*

On entry: the number of differential equations.

3: M(NEQ) — INTEGER array *Input*

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

4: NLBC — INTEGER *Input*

On entry: the number of boundary conditions at a.

5: GA(NLBC) — ***real*** array *Output*

On exit: the values of $g_i(z(y(a)))$, for $i = 1,2,\ldots,\mathrm{NLBC}$.

GAFUN must be declared as EXTERNAL in the (sub)program from which D02TKF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

4: GBFUN — SUBROUTINE, supplied by the user. *External Procedure*

GBFUN must evaluate the boundary conditions at the right hand end of the range, that is functions $\bar{g}_i(z(y(b)))$ for given values of $z(y(b))$.

Its specification is:

```
SUBROUTINE GBFUN(YB, NEQ, M, NRBC, GB)
INTEGER          NEQ, M(NEQ), NRBC
real             YB(NEQ,0:*), GB(NRBC)
```

1: YB(NEQ,0:∗) — ***real*** array *Input*

On entry: YB(i,j) contains $y_i^{(j)}(b)$, for $i = 1, 2, \ldots,$ NEQ, $j = 0, 1, \ldots,$ M$(i) - 1$.
Note. $y_i^{(0)}(b) = y_i(b)$.

2: NEQ — INTEGER *Input*

On entry: the number of differential equations.

3: M(NEQ) — INTEGER array *Input*

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

4: NRBC — INTEGER *Input*

On entry: the number of boundary conditions at b.

5: GB(NRBC) — ***real*** array *Output*

On exit: the values of $\bar{g}_i(z(y(b)))$, for $i = 1, 2, \ldots,$ NRBC.

GBFUN must be declared as EXTERNAL in the (sub)program from which D02TKF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

5: GAJAC — SUBROUTINE, supplied by the user. *External Procedure*

GAJAC must evaluate the partial derivatives of $g_i(z(y(a)))$ with respect to the elements of $z(y(a))$ $(= (y_1(a), y_1^1(a), \ldots, y_1^{(m_1-1)}(a), y_2(a), \ldots, y_n^{(m_n-1)}(a)))$.

Its specification is:

```
SUBROUTINE GAJAC(YA, NEQ, M, NLBC, DGADY)
INTEGER          NEQ, M(NEQ), NLBC
real             YA(NEQ,0:*), DGADY(NLBC,NEQ,0:*)
```

1: YA(NEQ,0:∗) — ***real*** array *Input*

On entry: YA(i,j) contains $y_i^{(j)}(a)$, for $i = 1, 2, \ldots,$ NEQ, $j = 0, 1, \ldots,$ M$(i) - 1$.
Note. $y_i^{(0)}(a) = y_i(a)$.

2: NEQ — INTEGER *Input*

On entry: the number of differential equations.

3: M(NEQ) — INTEGER array *Input*

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

4: NLBC — INTEGER *Input*

On entry: the number of boundary conditions at a.

5: DGADY(NLBC,NEQ,0:∗) — ***real*** array *Output*

On exit: DGADY(i,j,k) must contain the partial derivative of $g_i(z(y(a)))$ with respect to $y_j^{(k)}(a)$, for $i = 1, 2, \ldots,$ NLBC, $j = 1, 2, \ldots,$ NEQ, $k = 0, 1, \ldots,$ M$(j) - 1$. Only non-zero partial derivatives need be set.

GAJAC must be declared as EXTERNAL in the (sub)program from which D02TKF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

6: GBJAC — SUBROUTINE, supplied by the user. *External Procedure*

GBJAC must evaluate the partial derivatives of $\bar{g}_i(z(y(b)))$ with respect to the elements of $z(y(b))$ $(= (y_1(b), y_1^1(b), \ldots, y_1^{(m_1-1)}(b), y_2(b), \ldots, y_n^{(m_n-1)}(b)))$.

Its specification is:

```
SUBROUTINE GBJAC(YB, NEQ, M, NRBC, DGBDY)
INTEGER          NEQ, M(NEQ), NRBC
real             YB(NEQ,0:*), DGBDY(NRBC,NEQ,0:*)
```

1: YB(NEQ,0:∗) — ***real*** array *Input*

On entry: YB(i,j) contains $y_i^{(j)}(b)$, for $i = 1, 2, \ldots,$ NEQ, $j = 0, 1, \ldots,$ M$(i) - 1$.

Note. $y_i^{(0)}(b) = y_i(b)$.

2: NEQ — INTEGER *Input*

On entry: the number of differential equations.

3: M(NEQ) — INTEGER array *Input*

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

4: NRBC — INTEGER *Input*

On entry: the number of boundary conditions at a.

5: DGBDY(NRBC,NEQ,0:∗) — ***real*** array *Output*

On exit: DGBDY(i,j,k) must contain the partial derivative of $\bar{g}_i(z(y(b)))$ with respect to $y_j^{(k)}(b)$, for $i = 1, 2, \ldots,$ NRBC, $j = 1, 2, \ldots,$ NEQ, $k = 0, 1, \ldots,$ M$(j) - 1$. Only non-zero partial derivatives need be set.

GBJAC must be declared as EXTERNAL in the (sub)program from which D02TKF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

7: GUESS — SUBROUTINE, supplied by the user. *External Procedure*

GUESS must return initial approximations for the solution components $y_i^{(j)}$ and the derivatives $y_i^{(m_i)}$, for $i = 1, 2, \ldots,$ NEQ, $j = 0, 1, \ldots,$ M$(i) - 1$. Try to compute each derivative $y_i^{(m_i)}$ such that it corresponds to your approximations to $y_i^{(j)}$ for $j = 0, 1, \ldots,$ M$(i) - 1$. You should **not** call FFUN to compute $y_i^{(m_i)}$.

If D02TKF is being used in conjunction with D02TXF as part of a continuation process, then GUESS is not called by D02TKF after the call to D02TXF.

Its specification is:

```
SUBROUTINE GUESS(X, NEQ, M, Y, DYM)
INTEGER          M(NEQ), NEQ
real             X, Y(NEQ,0:*), DYM(NEQ)
```

1: X — ***real*** *Input*

On entry: the independent variable, x; $x \in [a, b]$.

2: NEQ — INTEGER *Input*

On entry: the number of differential equations.

3: M(NEQ) — INTEGER array *Input*

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

4: Y(NEQ,0:∗) — ***real*** array *Output*

On exit: Y(i,j) must contain $y_i^{(j)}(x)$, for $i = 1,2,\ldots,\mathrm{NEQ},\ j = 0,1,\ldots,\mathrm{M}(i)-1$.

Note. $y_i^{(0)}(x) = y_i(x)$.

5: DYM(NEQ) — ***real*** array *Output*

On exit: DYM(i) must contain $y_i^{(m_i)}(x)$, for $i = 1,2,\ldots,\mathrm{NEQ}$.

GUESS must be declared as EXTERNAL in the (sub)program from which D02TKF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

8: WORK(∗) — ***real*** array *Input/Output*

On entry: this must be the same array as supplied to D02TVF 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 D02TVF 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 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**.

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, an invalid call was made to D02TKF, for example, without a previous call to the setup routine D02TVF. 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

Numerical singularity has been detected in the Jacobian used in the underlying Newton iteration. No meaningful results have been computed. You should check carefully how you have coded procedures FJAC, GAJAC and GBJAC. If the procedures have been coded correctly then supplying a different initial approximation to the solution in GUESS might be appropriate. See also Section 8.

IFAIL = 3

The nonlinear iteration has failed to converge. At no time during the computation was convergence obtained and no meaningful results have been computed. You should check carefully how you have coded procedures FJAC, GAJAC and GBJAC. If the procedures have been coded correctly then supplying a better initial approximation to the solution in GUESS might be appropriate. See also Section 8.

IFAIL = 4

The nonlinear iteration has failed to converge. At some earlier time during the computation convergence was obtained and the corresponding results have been returned for diagnostic purposes and may be inspected by a call to D02TZF. Nothing can be said regarding the suitability of these results for use in any subsequent computation for the same problem. You should try to provide a better mesh and initial approximation to the solution in GUESS. See also Section 8.

IFAIL = 5

The expected number of sub-intervals required exceeds the maximum number specified by the argument MXMESH in the setup routine D02TVF. Results for the last mesh on which convergence was obtained have been returned. Nothing can be said regarding the suitability of these results for use in any subsequent computation for the same problem. An indication of the error in the solution on the last mesh where convergence was obtained can be obtained by calling D02TZF. The error requirements may need to be relaxed and/or the maximum number of mesh points may need to be increased. See also Section 8.

7 Accuracy

The accuracy of the solution is determined by the parameter TOLS in the prior call to D02TVF (see Section 3 of the document for D02TVF and Section 8 of the document for D02TVF for details and advice). Note that error control is applied only to solution components (variables) and not to any derivatives of the solution. An estimate of the maximum error in the computed solution is available by calling D02TZF.

8 Further Comments

If D02TKF returns with IFAIL = 2, 3, 4 or 5 and the call to D02TKF was a part of some continuation procedure for which successful calls to D02TKF have already been made, then it is possible that the adjustment(s) to the continuation parameter(s) between calls to D02TKF is (are) too large for the problem under consideration. More conservative adjustment(s) to the continuation parameter(s) might be appropriate.

9 Example

The following example is used to illustrate the treatment of a high order system, control of the error in a derivative of a component of the original system, and the use of continuation. See also D02TVF, D02TXF, D02TYF and D02TZF, for the illustration of other facilities.

Consider the steady flow of an incompressible viscous fluid between two infinite coaxial rotating discs. See [2] and the references therein. The governing equations are

$$\begin{aligned} \frac{1}{\sqrt{R}} f'''' + f f''' + g g' &= 0 \\ \frac{1}{\sqrt{R}} g'' + f g' - f' g &= 0 \end{aligned}$$

subject to the boundary conditions

$$f(0) = f'(0) = 0, \;\; g(0) = \Omega_0, \quad f(1) = f'(1) = 0, \;\; g(1) = \Omega_1,$$

where R is the Reynolds number and Ω_0, Ω_1 are the angular velocities of the disks.

We consider the case of counter-rotation and a symmetric solution, that is $\Omega_0 = 1, \Omega_1 = -1$. This problem is more difficult to solve, the larger the value of R. For illustration, we use simple continuation to compute the solution for three different values of R $(= 10^6, 10^8, 10^{10})$. However, this problem can be addressed directly for the largest value of R considered here. Instead of the values suggested in Section 5 of the document for D02TXF for NMESH, IPMESH and MESH in the call to D02TXF prior to a continuation call, we use every point of the final mesh for the solution of the first value of R, that is we must modify the contents of IPMESH. For illustrative purposes we wish to control the computed error in f' and so recast the equations as

$$\begin{aligned} y_1' &= y_2 \\ y_2''' &= -\sqrt{R}(y_1y_2'' + y_3y_3') \\ y_3'' &= \sqrt{R}(y_2y_3 - y_1y_3') \end{aligned}$$

subject to the boundary conditions

$$y_1(0) = y_2(0) = 0, \;\; y_3(0) = \Omega, \quad y_1(1) = y_2(1) = 0, \;\; y_3(1) = -\Omega, \quad \Omega = 1.$$

For the symmetric boundary conditions considered, there exists an odd solution about $x = 0.5$. Hence, to satisfy the boundary conditions, we use the following initial approximations to the solution in GUESS:

$$\begin{aligned} y_1(x) &= -x^2(x-\frac{1}{2})(x-1)^2 \\ y_2(x) &= -x(x-1)(5x^2-5x+1) \\ y_3(x) &= -8\Omega(x-\frac{1}{2})^3. \end{aligned}$$

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.

```
*     D02TKF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NEQ, MMAX, NLBC, NRBC, NCOL, MXMESH
      PARAMETER        (NEQ=3,MMAX=3,NLBC=3,NRBC=3,NCOL=7,MXMESH=51)
      INTEGER          LRWORK, LIWORK
      PARAMETER        (LRWORK=MXMESH*(109*NEQ**2+78*NEQ+7),
     +                 LIWORK=MXMESH*(11*NEQ+6))
*     .. Scalars in Common ..
      real             OMEGA, SQROFR
*     .. Local Scalars ..
      real             ERMX, R
      INTEGER          I, IERMX, IFAIL, IJERMX, J, NCONT, NMESH
*     .. 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, SQRT
*     .. Common blocks ..
      COMMON           /PROBS/SQROFR, OMEGA
```

```
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D02TKF Example Program Results'
      WRITE (NOUT,*)
      NMESH = 11
      MESH(1) = 0.0e0
      IPMESH(1) = 1
      DO 20 I = 2, NMESH - 1
         MESH(I) = (I-1)/real(NMESH-1)
         IPMESH(I) = 2
   20 CONTINUE
      MESH(NMESH) = 1.0e0
      IPMESH(NMESH) = 1
      M(1) = 1
      M(2) = 3
      M(3) = 2
      TOL(1) = 1.0e-4
      TOL(2) = TOL(1)
      TOL(3) = 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
      OMEGA = 1.0e0
      R = 1.0e+6
      DO 80 J = 1, NCONT
         SQROFR = SQRT(R)
         WRITE (NOUT,99999) TOL(1), R
*     Solve
         CALL D02TKF(FFUN,FJAC,GAFUN,GBFUN,GAJAC,GBJAC,GUESS,WORK,IWORK,
     +               IFAIL)
*     Extract mesh
         CALL D02TZF(MXMESH,NMESH,MESH,IPMESH,ERMX,IERMX,IJERMX,WORK,
     +               IWORK,IFAIL)
         WRITE (NOUT,99998) NMESH, ERMX, IERMX, IJERMX,
     +     (I,IPMESH(I),MESH(I),I=1,NMESH)
*     Print solution components on mesh
         WRITE (NOUT,99997)
         DO 40 I = 1, NMESH
            CALL D02TYF(MESH(I),Y,NEQ,MMAX,WORK,IWORK,IFAIL)
            WRITE (NOUT,99996) MESH(I), Y(1,0), Y(2,0), Y(3,0)
   40    CONTINUE
*     Select mesh for continuation and modify problem dependent
*     parameters
         IF (J.LT.NCONT) THEN
            R = 1.0e+02*R
            DO 60 I = 2, NMESH - 1
               IPMESH(I) = 2
   60       CONTINUE
            CALL D02TXF(MXMESH,NMESH,MESH,IPMESH,WORK,IWORK,IFAIL)
         END IF
   80 CONTINUE
      STOP
*
99999 FORMAT (/' Tolerance = ',1P,e8.1,'  R = ',e10.3)
99998 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))
99997 FORMAT (/'      x        f        f''       g')
99996 FORMAT (' ',F8.3,1X,3F9.4)
      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)
*     .. Scalars in Common ..
      real             OMEGA, SQROFR
*     .. Common blocks ..
      COMMON           /PROBS/SQROFR, OMEGA
*     .. Executable Statements ..
      F(1) = Y(2,0)
      F(2) = -(Y(1,0)*Y(2,2)+Y(3,0)*Y(3,1))*SQROFR
      F(3) = (Y(2,0)*Y(3,0)-Y(1,0)*Y(3,1))*SQROFR
      RETURN
      END
      SUBROUTINE FJAC(X,Y,NEQ,M,DFDY)
*     .. Scalar Arguments ..
      real             X
      INTEGER          NEQ
*     .. Array Arguments ..
      real             DFDY(NEQ,NEQ,0:*), Y(NEQ,0:*)
      INTEGER          M(NEQ)
*     .. Scalars in Common ..
      real             OMEGA, SQROFR
*     .. Common blocks ..
      COMMON           /PROBS/SQROFR, OMEGA
*     .. Executable Statements ..
      DFDY(1,2,0) = 1.0e0
      DFDY(2,1,0) = -Y(2,2)*SQROFR
      DFDY(2,2,2) = -Y(1,0)*SQROFR
      DFDY(2,3,0) = -Y(3,1)*SQROFR
      DFDY(2,3,1) = -Y(3,0)*SQROFR
      DFDY(3,1,0) = -Y(3,1)*SQROFR
      DFDY(3,2,0) = Y(3,0)*SQROFR
      DFDY(3,3,0) = Y(2,0)*SQROFR
      DFDY(3,3,1) = -Y(1,0)*SQROFR
      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              OMEGA, SQROFR
*     .. Common blocks ..
      COMMON            /PROBS/SQROFR, OMEGA
*     .. Executable Statements ..
      GA(1) = YA(1,0)
      GA(2) = YA(2,0)
      GA(3) = YA(3,0) - OMEGA
```

```
      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             OMEGA, SQROFR
*     .. Common blocks ..
      COMMON           /PROBS/SQROFR, OMEGA
*     .. Executable Statements ..
      GB(1) = YB(1,0)
      GB(2) = YB(2,0)
      GB(3) = YB(3,0) + OMEGA
      RETURN
      END
      SUBROUTINE GAJAC(YA,NEQ,M,NLBC,DGADY)
*     .. Scalar Arguments ..
      INTEGER          NEQ, NLBC
*     .. Array Arguments ..
      real             DGADY(NLBC,NEQ,0:*), YA(NEQ,0:*)
      INTEGER          M(NEQ)
*     .. Executable Statements ..
      DGADY(1,1,0) = 1.0e0
      DGADY(2,2,0) = 1.0e0
      DGADY(3,3,0) = 1.0e0
      RETURN
      END
      SUBROUTINE GBJAC(YB,NEQ,M,NRBC,DGBDY)
*     .. Scalar Arguments ..
      INTEGER          NEQ, NRBC
*     .. Array Arguments ..
      real             DGBDY(NRBC,NEQ,0:*), YB(NEQ,0:*)
      INTEGER          M(NEQ)
*     .. Executable Statements ..
      DGBDY(1,1,0) = 1.0e0
      DGBDY(2,2,0) = 1.0e0
      DGBDY(3,3,0) = 1.0e0
      RETURN
      END
      SUBROUTINE GUESS(X,NEQ,M,Y,DYM)
*     .. Scalar Arguments ..
      real             X
      INTEGER          NEQ
*     .. Array Arguments ..
      real             DYM(NEQ), Y(NEQ,0:*)
      INTEGER          M(NEQ)
*     .. Scalars in Common ..
      real             OMEGA, SQROFR
*     .. Common blocks ..
      COMMON           /PROBS/SQROFR, OMEGA
*     .. Executable Statements ..
      Y(1,0) = -X**2*(X-0.5e0)*(X-1.0e0)**2
      Y(2,0) = -X*(X-1.0e0)*(5.0e0*X**2-5.0e0*X+1.0e0)
      Y(2,1) = -(20.0e0*X**3-30.0e0*X**2+12.0e0*X-1.0e0)
      Y(2,2) = -(60.0e0*X**2-60.0e0*X+12.0e0*X)
```

```
      Y(3,0) = -8.0e0*OMEGA*(X-0.5e0)**3
      Y(3,1) = -24.0e0*OMEGA*(X-0.5e0)**2
      DYM(1) = Y(2,0)
      DYM(2) = -(120.0e0*X-60.0e0)
      DYM(3) = -56.0e0*OMEGA*(X-0.5e0)
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
 D02TKF Example Program Results


 Tolerance =  1.0E-04  R =  1.000E+06

 Used a mesh of   21 points
 Maximum error =   0.62E-09  in interval   20 for component    3

 Mesh points:
   1(1) 0.0000E+00   2(3) 0.5000E-01   3(2) 0.1000E+00   4(3) 0.1500E+00
   5(2) 0.2000E+00   6(3) 0.2500E+00   7(2) 0.3000E+00   8(3) 0.3500E+00
   9(2) 0.4000E+00  10(3) 0.4500E+00  11(2) 0.5000E+00  12(3) 0.5500E+00
  13(2) 0.6000E+00  14(3) 0.6500E+00  15(2) 0.7000E+00  16(3) 0.7500E+00
  17(2) 0.8000E+00  18(3) 0.8500E+00  19(2) 0.9000E+00  20(3) 0.9500E+00
  21(1) 0.1000E+01

      x        f         f'        g
    0.000    0.0000    0.0000    1.0000
    0.050    0.0070    0.1805    0.4416
    0.100    0.0141    0.0977    0.1886
    0.150    0.0171    0.0252    0.0952
    0.200    0.0172   -0.0165    0.0595
    0.250    0.0157   -0.0400    0.0427
    0.300    0.0133   -0.0540    0.0322
    0.350    0.0104   -0.0628    0.0236
    0.400    0.0071   -0.0683    0.0156
    0.450    0.0036   -0.0714    0.0078
    0.500    0.0000   -0.0724    0.0000
    0.550   -0.0036   -0.0714   -0.0078
    0.600   -0.0071   -0.0683   -0.0156
    0.650   -0.0104   -0.0628   -0.0236
    0.700   -0.0133   -0.0540   -0.0322
    0.750   -0.0157   -0.0400   -0.0427
    0.800   -0.0172   -0.0165   -0.0595
    0.850   -0.0171    0.0252   -0.0952
    0.900   -0.0141    0.0977   -0.1886
    0.950   -0.0070    0.1805   -0.4416
    1.000    0.0000    0.0000   -1.0000

 Tolerance =  1.0E-04  R =  1.000E+08

 Used a mesh of   21 points
 Maximum error =   0.45E-08  in interval    6 for component    3
```

```
Mesh points:
  1(1) 0.0000E+00   2(3) 0.1757E-01   3(2) 0.3515E-01   4(3) 0.5203E-01
  5(2) 0.6891E-01   6(3) 0.8593E-01   7(2) 0.1030E+00   8(3) 0.1351E+00
  9(2) 0.1672E+00  10(3) 0.2306E+00  11(2) 0.2939E+00  12(3) 0.4713E+00
 13(2) 0.6486E+00  14(3) 0.7455E+00  15(2) 0.8423E+00  16(3) 0.8824E+00
 17(2) 0.9225E+00  18(3) 0.9449E+00  19(2) 0.9673E+00  20(3) 0.9836E+00
 21(1) 0.1000E+01

     x        f         f'        g
   0.000    0.0000    0.0000    1.0000
   0.018    0.0025    0.1713    0.3923
   0.035    0.0047    0.0824    0.1381
   0.052    0.0056    0.0267    0.0521
   0.069    0.0058    0.0025    0.0213
   0.086    0.0057   -0.0073    0.0097
   0.103    0.0056   -0.0113    0.0053
   0.135    0.0052   -0.0135    0.0027
   0.167    0.0047   -0.0140    0.0020
   0.231    0.0038   -0.0142    0.0015
   0.294    0.0029   -0.0142    0.0012
   0.471    0.0004   -0.0143    0.0002
   0.649   -0.0021   -0.0143   -0.0008
   0.745   -0.0035   -0.0142   -0.0014
   0.842   -0.0049   -0.0139   -0.0022
   0.882   -0.0054   -0.0127   -0.0036
   0.922   -0.0058   -0.0036   -0.0141
   0.945   -0.0057    0.0205   -0.0439
   0.967   -0.0045    0.0937   -0.1592
   0.984   -0.0023    0.1753   -0.4208
   1.000    0.0000    0.0000   -1.0000

Tolerance =  1.0E-04  R =  1.000E+10

Used a mesh of   21 points
Maximum error =   0.31E-05  in interval    7 for component    3

Mesh points:
  1(1) 0.0000E+00   2(3) 0.6256E-02   3(2) 0.1251E-01   4(3) 0.1851E-01
  5(2) 0.2450E-01   6(3) 0.3076E-01   7(2) 0.3702E-01   8(3) 0.4997E-01
  9(2) 0.6292E-01  10(3) 0.9424E-01  11(2) 0.1256E+00  12(3) 0.4190E+00
 13(2) 0.7125E+00  14(3) 0.8246E+00  15(2) 0.9368E+00  16(3) 0.9544E+00
 17(2) 0.9719E+00  18(3) 0.9803E+00  19(2) 0.9886E+00  20(3) 0.9943E+00
 21(1) 0.1000E+01

     x        f         f'        g
   0.000    0.0000    0.0000    1.0000
   0.006    0.0009    0.1623    0.3422
   0.013    0.0016    0.0665    0.1021
   0.019    0.0018    0.0204    0.0318
   0.025    0.0019    0.0041    0.0099
   0.031    0.0019   -0.0014    0.0028
   0.037    0.0019   -0.0031    0.0007
   0.050    0.0019   -0.0038   -0.0002
   0.063    0.0018   -0.0038   -0.0003
   0.094    0.0017   -0.0039   -0.0003
   0.126    0.0016   -0.0039   -0.0002
   0.419    0.0004   -0.0041   -0.0001
   0.712   -0.0008   -0.0042    0.0001
```

```
 0.825   -0.0013  -0.0043   0.0002
 0.937   -0.0018  -0.0043   0.0003
 0.954   -0.0019  -0.0042   0.0001
 0.972   -0.0019  -0.0003  -0.0049
 0.980   -0.0019   0.0152  -0.0252
 0.989   -0.0015   0.0809  -0.1279
 0.994   -0.0008   0.1699  -0.3814
 1.000    0.0000   0.0000  -1.0000
```