Collocation Software for Boundary-Value ODEs

U. ASCHER
University of British Columbia, Canada
and
J. CHRISTIANSEN and R. D. RUSSELL
Simon Fraser University, Canada

The use of a general-purpose code, COLSYS, is described. The code is capable of solving mixed-order systems of boundary-value problems in ordinary differential equations. The method of spline collocation at Gaussian points is implemented using a B-spline basis. Approximate solutions are computed on a sequence of automatically selected meshes until a user-specified set of tolerances is satisfied. A damped Newton's method is used for the nonlinear iteration. The code has been found to be particularly effective for difficult problems.

It is intended that a user be able to use COLSYS easily after reading its algorithm description. The use of the code is then illustrated by examples demonstrating its effectiveness and capabilities.

Key Words and Phrases: ordinary differential equations, boundary-value problems, collocation, B-spline, mesh selection, error estimates, damped Newton's method, general-purpose code
CR Categories 5.17

The Algorithm: COLSYS· Collocation Software for Boundary-Value ODEs, *ACM Trans. Math. Softw.* 7, 2 (June 1981), 223–229.

1. INTRODUCTION

While research in numerical methods for solving boundary-value problems (BVPs) for ordinary differential equations (ODEs) has been very active for some time, robust software for solving these problems has only recently been developed and partially tested. A few codes based on shooting [11], multiple shooting [5, 9], other initial-value techniques [15, 16], and finite differences with deferred corrections [13, 14] have been reported. See [5a] for further details.

Despite these advances, the accessibility of BVP software is at this time considerably behind that for initial-value problems (IVP), as evidenced by the fact that the large numerical libraries, for example, NAG and IMSL, contain either a simple shooting code or nothing at all for BVPs while having several

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.
Authors' addresses: U. Ascher, Department of Computer Science, University of British Columbia, Vancouver, B.C. V6T 1W5, Canada; J. Christiansen and R. D. Russell, Mathematics Department, Simon Fraser University, Burnaby, B.C. V5A 1S6, Canada.
© 1981 ACM 0098-3500/81/0600-0209 $00.75

advanced packages for IVPs. This is partially due to the fact that most of the above-mentioned codes are undergoing fairly regular modification.

We consider here a *col*location package for solving mixed-order *sys*tems of multipoint BVPs (COLSYS) [2, 3]. The (FORTRAN-written) code COLSYS has proved to be competitive with the other robust software for solving BVPs and to be particularly effective for difficult problems. The aim of this paper is to demonstrate the usefulness of COLSYS while describing how to use it through examples.

After a brief description of the class of problems admitted by COLSYS (Section 2) and the numerical techniques used to solve these problems (Section 3), the use of the code is demonstrated through three examples, ranging from simple to more sophisticated (Section 4). We conclude with a number of remarks concerning the use and usefulness of COLSYS.

2. PROBLEM DEFINITION

Consider a mixed-order system of d nonlinear differential equations of orders $1 \le m_1 \le m_2 \le \cdots \le m_d \le 4$,

$$u_n^{(m_n)}(x) = F_n(x; \mathbf{z}(\mathbf{u})), \qquad a < x < b \quad n = 1, \ldots, d, \tag{1}$$

where the sought solution $\mathbf{u}(x) = (u_1(x), \ldots, u_d(x))$ is an isolated solution vector and $\mathbf{z}(\mathbf{u}) = (u_1, u_1', \ldots, u^{(m_1-1)}, u_2, \ldots, u_d, \ldots, u_d^{(m_d-1)})$ is the vector of unknowns that would result from converting (1) to a first-order system. The system is subject to $m^* = \sum_{n=1}^{d} m_n$ nonlinear multipoint separated boundary conditions

$$g_j(\zeta_j; \mathbf{z}(\mathbf{u})) = 0, \qquad j = 1, \ldots, m^*, \tag{2}$$

where ζ_j is the location of the jth boundary (or side) condition, $a \le \zeta_1 \le \zeta_2 \le \cdots \le \zeta_{m^*} \le b$. For the simple example $u'' + e^u = 0$, $u(0) = u(1) = 0$, we have $d = 1$, $m_1 = 2$, $a = 0$, $b = 1$, $F_1 = -e^{z_1}$, $\mathbf{z}(u) = (u, u')$, $m^* = 2$, $\zeta_1 = 0$, $\zeta_2 = 1$, $g_1 = g_2 = z_1$.

Unlike the previously mentioned general-purpose codes, COLSYS does not explicitly convert (1) to a first-order system. Also, while (2) does not explicitly allow for nonseparated boundary conditions, such problems can be converted to the form (1), (2) at the expense of increasing the order of the system [2].

3. THE METHOD

The method of spline collocation at Gaussian points, as described in detail in [2] and references therein, is implemented in COLSYS to solve (1), (2). The problem is solved on a sequence of meshes until a user-specified set of tolerances is satisfied. For a particular mesh $\pi: a = x_1 < x_2 < \cdots < x_{N+1} = b$ with $h_i = x_{i+1} - x_i$, $h = \max_{1 \le i \le N} h_i$, and an integer $k > m_d$, a collocation solution $\mathbf{v}(x) = (v_1, \ldots, v_d)$ is determined with each $v_n(x) \in C^{m_n - 1}[a, b]$ being a polynomial of degree $< k + m_n$ on each subinterval (x_i, x_{i+1}), $i = 1, \ldots, N$. Specifically, $\mathbf{v}(x)$ is required to satisfy the differential equations (1) at the images of the k Gauss–Legendre points in each subinterval and the conditions (2). (This gives $Nkd + m^*$ equations.) Provided the problem (1), (2) is smooth enough, the local behavior of the error for $x \in [x_i, x_{i+1})$ is

$$\| u_n^{(l)}(x) - v_n^{(l)}(x) \|_{(i)} = c_{n,l} | u_n^{(k+m_n)}(x_i) | h_i^{k+m_n-l} + O(h^{k+m_n-l+1}),$$
$$l = 0, \ldots, m_n - 1, \quad n = 1, \ldots, d, \tag{3}$$

where $c_{n,l}$ are known constants and for any appropriate function ψ,

$$\|\psi\|_{(i)} := \max_{x\in[x_i,x_{i+1})} |\psi(x)|. \tag{4}$$

The expression (3) is used both for estimating the error (using mesh halving) to check against user-prescribed tolerances and for mesh refinement. By approximating $u_n^{(k+m_n)}(x_i)$ using $v_n(x)$, a rough approximation for the error in each subinterval is obtained. If deemed worthwhile, a redistribution of the mesh points is performed to roughly equidistribute the error (i.e., roughly equalize the error in each subinterval), and $v(x)$ is recomputed. If not, each subinterval is halved, a new solution $v^*(x)$ is computed, and the error in $v^*(x)$ is estimated using $v(x)$, $v^*(x)$, and (3). See [2, 3] for more detailed expositions and theoretical justification. The sound basis for the adaptive mesh-selection procedure helps make COLSYS competitive with the initial-value-type codes that rely on robust IVP solvers [2].

The components $v_n(x)$ of the collocation solution are expressed in terms of a B-spline basis. The evaluation of the B-splines and their derivatives is done using de Boor's algorithms [6] appropriately modified to make use of various savings for our particular application [4]. The B-splines are used, besides evaluating the solution $\mathbf{v}(x)$ and its derivatives, to construct the collocation equations. This results in a linear (or linearized) system of equations, the unknowns being the B-spline coefficients. The system is "almost block diagonal" [2] and the package of de Boor–Weiss [7] is used for its decomposition and solution.

Nonlinear problems are solved using the damped Newton's method of quasi-linearization [3]. Thus at each iteration a linearized problem is solved by collocation as described above. The damping or relaxation factor is controlled by a scheme that is a slight modification of that suggested by Deuflhard [8]. If the problem (1), (2) is not prespecified by the user as being "sensitive", and if nonlinear convergence on a mesh has just been obtained, then on the next mesh a modified Newton method with a fixed Jacobian is performed as long as the residual monotonically decreases at a sufficiently rapid rate [3].

To use COLSYS, the user must specify a set of tolerances tol_j and pointers $l\mathrm{tol}_j, j = 1, \ldots, n\mathrm{tol}$. The successful stopping criterion for COLSYS is that

$$\|z_l(\mathbf{u}) - z_l(\mathbf{v})\|_{(i)} \le \mathrm{tol}_j + \|z_l(\mathbf{v})\|_{(i)} \cdot \mathrm{tol}_j,$$
$$l = l\mathrm{tol}_j, \quad j = 1, \ldots, n\mathrm{tol}, \quad i = 1, \ldots, N. \tag{5}$$

4. USE OF COLSYS THROUGH EXAMPLES

It is intended that a user be able to use COLSYS easily, including its more sophisticated features if needed, after reading its algorithm description. Here we demonstrate the use of the code on three examples.

All three examples presented here were run on an Amdahl V/6-II computer using double precision (14 hexadecimal digits). The listings, though, are of the single-precision version.

Example 1. The following problem describes a uniformly loaded beam of variable stiffness, simply supported at both ends [10]

$$(x^3u'')'' = x^3u'''' + 6x^2u''' + 6xu'' = 1, \qquad 1 \le x \le 2. \tag{6}$$

$$u = u'' = 0 \qquad \text{at} \quad x = 1, 2. \tag{7}$$

This has the exact solution

$$u(x) = \frac{1}{4}(10 \ln 2 - 3)(1 - x) + \frac{1}{2}\left[\frac{1}{x} + (3 + x)\ln x - x\right]. \tag{8}$$

We begin with this very easy problem in order to demonstrate a complete problem setup, with tolerances on u and u'' only, and then to verify the error estimates using the exact solution. We use (the default) five collocation points per subinterval and the first mesh contains only one subinterval; that is, it is initially a polynomial collocation (of order 9). The maximum error magnitude in the approximate solution is evaluated at 100 equidistant points after returning from COLSYS and compared to the estimates from the code. The complete problem setup follows.

```
      REAL FSPACE(2000), ZETA(4), TOL(2), Z(4), U(4), ERR(4)
      INTEGER ISPACE(200), M(1), IPAR(11), LTOL(2)
      EXTERNAL FSUB, DFSUB, GSUB, DGSUB, DUMMY
C
      WRITE (6,99)
   99 FORMAT(1H1, 35H EXAMPLE OF A SIMPLE PROBLEM SETUP.
     .         /  46H  UNIFORMLY LOADED BEAM OF VARIABLE STIFFNESS,
     .         /  32H  SIMPLY SUPPORTED AT BOTH ENDS. /)
C
C     ONE DIFFERENTIAL EQUATION OF ORDER 4.
      M(1) = 4
C     GIVE LOCATION OF BOUNDARY CONDITIONS
      ZETA(1) = 1.
      ZETA(2) = 1.
      ZETA(3) = 2.
      ZETA(4) = 2.
C     SET UP PARAMETER ARRAY.
C     USE DEFAULT VALUES FOR ALL PARAMETERS EXCEPT FOR INITIAL
C     MESH SIZE, NO. OF TOLERANCES AND SIZES OF WORK ARRAYS
      DO 10 I=1,11
   10   IPAR(I) = 0
      IPAR(3) = 1
      IPAR(4) = 2
      IPAR(5) = 2000
      IPAR(6) = 200
C     TWO ERROR TOLERANCES (ON U AND ITS SECOND DERIVATIVE)
      LTOL(1) = 1
      LTOL(2) = 3
      TOL(1) = 1.E-7
      TOL(2) = 1.E-7
C
      CALL COLSYS (1, M, 1., 2., ZETA, IPAR, LTOL, TOL,
     .             FIXPNT, ISPACE, FSPACE, IFLAG, FSUB,
     .             DFSUB, GSUB, DGSUB, DUMMY)
C
      IF (IFLAG .NE. 1)  STOP
C     CALCULATE THE ERROR AT 101 POINTS USING THE KNOWN
C     EXACT SOLUTION
      X = 1.
      DO 20 I=1,4
   20    ERR(I) = 0.
      DO 40 J=1,101
         CALL APPSLN (X, Z, FSPACE, ISPACE)
         CALL EXACT (X, U)
         DO 30 I=1,4
   30       ERR(I) = AMAX1(ERR(I), ABS(U(I)-Z(I)))
```

```
 40     X = X + .01
    WRITE(6,100) (ERR(I),I=1,4)
100 FORMAT(/27H ERROR TOLERANCES SATISFIED//22H THE EXACT ERRORS ARE,
   .       / 7X,4E12.4)
    STOP
    END


        SUBROUTINE FSUB (X, Z, F)
        REAL Z(4), F(1), X
        F(1) = (1. - 6.*X**2*Z(4) - 6.*X*Z(3)) / X**3
        RETURN
        END

        SUBROUTINE DFSUB (X, Z, DF)
        REAL Z(4), DF(1,4), X
        DF(1,1) = 0.
        DF(1,2) = 0.
        DF(1,3) = -6./X**2
        DF(1,4) = -6./X
        RETURN
        END

        SUBROUTINE GSUB (I, Z, G)
        REAL Z(4), G
        GO TO (1, 2, 1, 2), I
      1 G = Z(1) - 0.
        RETURN
      2 G = Z(3) - 0.
        RETURN
        END

        SUBROUTINE DGSUB (I, Z, DG)
        REAL Z(4), DG(4)
        DO 10 J=1,4
     10    DG(J) = 0.
        GO TO (1, 2, 1, 2), I
      1 DG(1) = 1.
        RETURN
      2 DG(3) = 1.
        RETURN
        END

        SUBROUTINE EXACT(X, U)
        REAL U(4)
C       EXACT SOLUTION
        U(1) = .25* (10.*ALOG(2.)-3.) * (1.-X) +
       .       .5* (1./X+ (3.+X)*ALOG(X) - X)
        U(2) = -.25* (10.*ALOG(2.) - 3.) + .5 *
       .        (-1./X/X + ALOG(X) + (3.+X)/X - 1.)
        U(3) = .5 * (2./X**3 + 1./X -3./X/X)
        U(4) = .5 * (-6./X**4 - 1./X/X + 6./X**3)
        RETURN
        END

        SUBROUTINE DUMMY
        END
```

The resulting output is listed next. Owing to the simplicity of this problem, no mesh selection is found necessary.

```
EXAMPLE OF A SIMPLE PROBLEM SETUP:
UNIFORMLY LOADED BEAM OF VARIABLE STIFFNESS,
SIMPLY SUPPORTED AT BOTH ENDS.

THE NEW MESH (OF   1 SUBINTERVALS):
  1.000000  2.000000

THE NEW MESH (OF   2 SUBINTERVALS):
  1.000000  1.500000  2.000000

THE ESTIMATED ERRORS ARE:
U( 1):  0.7611E-08  0.2033E-06  0.2141E-05  0.5427E-04

THE NEW MESH (OF   4 SUBINTERVALS):
  1.000000  1.250000  1.500000  1.750000  2.000000

THE ESTIMATED ERRORS ARE:
U( 1):  0.6982E-10  0.3171E-08  0.8164E-07  0.3777E-05

ERROR TOLERANCES SATISFIED

THE EXACT ERRORS ARE:
  0.1739E-09  0.6268E-08  0.2184E-06  0.9574E-05
```

Example 2. Consider the following pair of second-order differential equations for ϕ and ψ,

$$\epsilon^4/\mu\left[\phi'' + \frac{1}{x}\phi' - \frac{1}{x^2}\phi\right] + \psi\left(1 - \frac{1}{x}\phi\right) - \phi = -\gamma x\left(1 - \frac{1}{2}x^2\right), \qquad 0 < x < 1 \tag{9}$$

$$\mu\left[\psi'' + \frac{1}{x}\psi' - \frac{1}{x^2}\psi\right] - \phi\left(1 - \frac{1}{2x}\phi\right) = 0, \tag{10}$$

subject to the boundary conditions

$$\phi = x\psi' - 0.3\psi + 0.7x = 0, \quad \text{at} \quad x = 0, 1. \tag{11}$$

These equations describe the small finite deformation of a thin shallow spherical cap of constant thickness subject to a quadratically varying axisymmetric external pressure distribution superimposed on a uniform internal pressure distribution [17]. Here ϕ is the dimensionless meridianal angle change of the deformed shell and ψ is a stress function. For $\gamma = 1.1$ and $\mu = O(\epsilon) \ll 1$, there is the possibility that a dimple will form, corresponding to an interior transition layer in ϕ. Also a boundary layer appears in ϕ at the clamped edge $x = 1$. The mesh refinement procedure in COLSYS automatically adapts the subinterval distribution to accommodate the solution behavior. Also, nothing needs to be done about the singular coefficients in (9), (10).

For $\epsilon = \mu = 10^{-3}$, two solutions for (9)–(11) are obtained by starting the nonlinear iteration from two different points.

A. The initial approximation is $\phi \equiv \psi \equiv 0$. The corresponding driver contains the following segment:

```
      .
      .
      .
GAMMA = 1.1
EPS = .001
DMU = EPS
EPS4MU = EPS**4/DMU
XT = SQRT(2.*(GAMMA-1.)/GAMMA)
```

```
C
C     IPAR  VALUES
C     A NONLINEAR PROBLEM
      IPAR(1) = 1
C     4 COLLOCATION POINTS PER SUBINTERVAL
      IPAR(2) = 4
C     INITIAL UNIFORM MESH OF 10 SUBINTERVALS
      IPAR(3) = 10
      IPAR(8) = 0
C     DIMENSION OF REAL WORK ARRAY  FSPACE  IS 40000
      IPAR(5) = 40000
C     DIMENSION OF INTEGER WORK ARRAY  ISPACE  IS 2500
      IPAR(6) = 2500
C     (THESE DIMENSIONS OF  FSPACE  AND  ISPACE
C      ENABLE  COLSYS  TO USE MESHES OF UP TO 192 INTERVALS.)
C     PRINT FULL OUTPUT.
      IPAR(7) = -1
C     INITIAL APPROXIMATION FOR NONLINEAR ITERATION IS 0 (DEFAULT)
      IPAR(9) = 0
C     A REGULAR PROBLEM
      IPAR(10) = 0
C     NO FIXED POINTS IN THE MESH
      IPAR(11) = 0
C     TOLERANCES ON  ALL COMPONENTS
      IPAR(4) = 4
      DO 10 I=1,4
        LTOL(I) = I
   10   TOL(I) = 1.E-5
          .
          .
          .



      SUBROUTINE FSUB (X, Z, F)
      DIMENSION Z(4), F(2)
      COMMON EPS, DMU, EPS4MU, GAMMA, XT
      F(1) = Z(1)/X/X - Z(2)/X + (Z(1) - Z(3)*(1.-Z(1)/X) -
     .       GAMMA*X*(1.-X*X/2.)) / EPS4MU
      F(2) = Z(3)/X/X - Z(4)/X + Z(1)*(1.-Z(1)/2./X) / DMU
      RETURN
      END

      SUBROUTINE DFSUB (X, Z, DF)
      DIMENSION Z(4), DF(2,4)
      COMMON EPS, DMU, EPS4MU, GAMMA, XT
      DF(1,1) = 1./X/X +(1. + Z(3)/X) / EPS4MU
      DF(1,2) = -1./X
      DF(1,3) = -(1.-Z(1)/X) / EPS4MU
      DF(1,4) = 0.
      DF(2,1) = (1. - Z(1)/X) / DMU
      DF(2,2) = 0.
      DF(2,3) = 1./X/X
      DF(2,4) = -1./X
      RETURN
      END
```

The obtained solution ϕ is plotted in Figure 1 (number I). There is no dimple. The last mesh for this computation has 38 subintervals, 22 of which are contained in (0.999, 1].

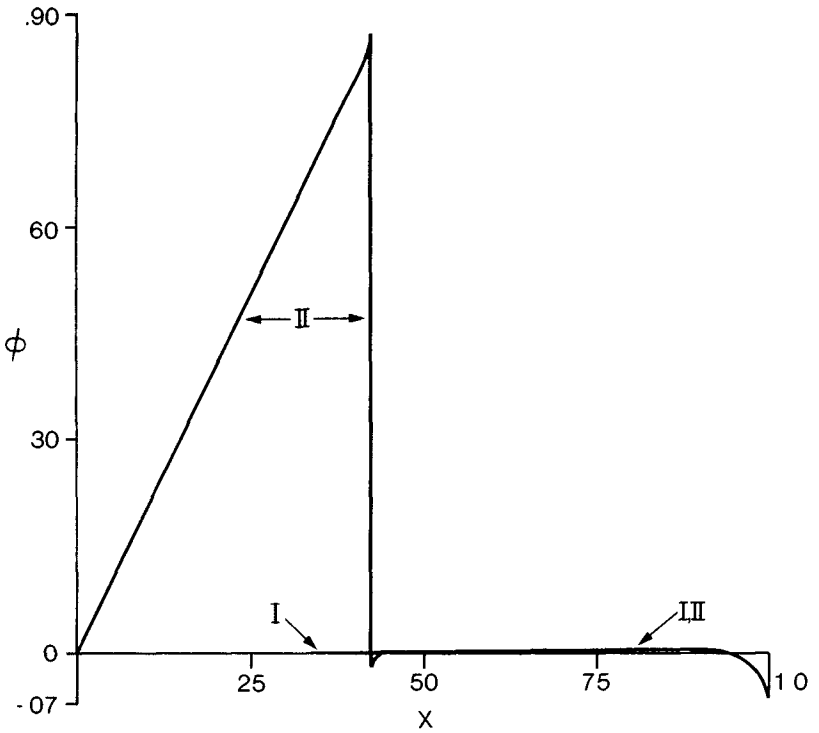

Fig. 1. Example 2.

B. The initial approximation is

$$\phi = \begin{cases} 2x, & x \le x_t \\ 0, & x > x_t \end{cases}$$

$$\psi = \begin{cases} -2x + \gamma x(1 - \frac{1}{2}x^2), & x \le x_t \\ -\gamma x(1 - \frac{1}{2}x^2), & x > x_t \end{cases}$$

where $x_t = \sqrt{2(\gamma - 1)/\gamma}$ (cf. [17]). The driving program is identical to the previous one except that we set IPAR(9) = 1 and define

```
      SUBROUTINE SOLUTN (X, Z, DMVAL)
      COMMON EPS, DMU, EPS4MU, GAMMA, XT
      DIMENSION Z(4) , DMVAL(2)
      CONS = GAMMA * X * (1.-.5*X*X)
      DCONS = GAMMA * (1. - 1.5*X*X)
      D2CONS = -3. * GAMMA * X
      IF (X .GT. XT) GO TO 10
      Z(1) = 2. * X
      Z(2) = 2.
      Z(3) = -2.*X + CONS
      Z(4) = -2. + DCONS
      DMVAL(2) = D2CONS
      GO TO 20
   10 Z(1) = 0.
      Z(2) = 0.
      Z(3) = -CONS
```

```
      Z(4) = -DCONS
      DMVAL(2) = -D2CONS
   20 DMVAL(1) = 0.
      RETURN
      END
```

The solution ϕ obtained in this case has a dimple. It is plotted in Figure 1 (number II). The last mesh for this computation has 160 subintervals, 85 of which are in (0.41, 0.45) and 36 in (0.999, 1].

Example 3. The following model describes the velocities in a boundary layer produced by the rotating flow of a viscous incompressible fluid over a stationary infinite disk [12].

$$G'' + \frac{3-n}{2} HG' + (n-1)H'G - s(G-1) = 0 \tag{12}$$

$$H''' + \frac{3-n}{2} HH'' + n(H')^2 - 1 + G^2 - sH' = 0 \tag{13}$$

$$G(0) = H(0) = H'(0) = 0, \qquad G(\infty) = 1, \qquad H'(\infty) = 0. \tag{14}$$

The solution functions G and H oscillate, and it is difficult to find initial approximate solutions and meshes that lead to convergence of the nonlinear iteration when $n \uparrow 1$ and $s \downarrow 0$. Also, the finite right end of the interval of integration, L, which adequately represents ∞, increases with n.

We start with the initial approximation $G = 1 - e^{-x}$, $H = -x^2e^{-x}$, and solve for $n = 0.2$, $s = 0.2$, $L = 60$. We then use simple continuation: The obtained solution is used as an initial approximation to solve for $n = 0.2$, $s = 0.1$, $L = 120$, and this solution in turn is used to solve for $n = 0.2$, $s = 0.05$, $L = 200$. In order to facilitate this process, all problems are first mapped from $[0, L]$ onto the unit interval. Thus the problems solved are

$$G'' = L^2 s(G-1) - L\left(\frac{3-n}{2} HG' + (n-1)H'G\right), \qquad 0 < x < 1 \tag{15}$$

$$H''' = L^3(1-G^2) + L^2 sH' - L\left(\frac{3-n}{2} HH'' + n(H')^2\right) \tag{16}$$

$$G(0) = H(0) = H'(0) = 0, \qquad G(1) = 1, \qquad H'(1) = 0. \tag{17}$$

The program which does all of this follows.

```
      REAL ZETA(5), FSPACE(40000), TOL(2), SVAL(3), ELVAL(3)
      INTEGER M(2), ISPACE(2500), LTOL(2), IPAR(11)
      REAL Z(5)
      COMMON EN, S, EL, CONS
      EXTERNAL FSUB, DFSUB, GSUB, DGSUB, SOLUTN
      DATA SVAL/.2, .1, .05/, ELVAL/60., 120., 200./
C
      EN = .2
      CONS = .5 * (3.-EN)
      NCOMP = 2
      M(1) = 2
```

```
      M(2) = 3
      ALEFT = 0.
      ARIGHT = 1.
C
      ZETA(1) = 0.
      ZETA(2) = 0.
      ZETA(3) = 0.
      ZETA(4) = 1.
      ZETA(5) = 1.
C
      IPAR(1) = 1
      IPAR(2) = 4
      IPAR(3) = 10
      IPAR(4) = 2
      IPAR(5) = 40000
      IPAR(6) = 2500
      IPAR(7) = 0
      IPAR(8) = 0
      IPAR(9) = 1
      IPAR(10) = 0
      IPAR(11) = 0
C
      LTOL(1) = 1
      LTOL(2) = 3
      TOL(1) = 1.E-5
      TOL(2) = 1.E-5
C
C     SOLVE A CHAIN OF 3 PROBLEMS
C
      DO 777 IJK = 1,3
         S = SVAL(IJK)
         EL = ELVAL(IJK)
         IF (IJK .EQ. 1)  GO TO 701
C        SET CONTINUATION PARAMETERS
         IPAR(9) = 3
         IPAR(3) = ISPACE(1)
  701    CONTINUE
         WRITE (6,100) EN, S, EL
  100    FORMAT(1H1,38H ROTATING FLOW OVER A STATIONARY DISK.
     .          /19H  PARAMETERS -  N =,F5.2, 6H   S =,F5.2,
     .          6H   L =,F6.1/)
C
         CALL COLSYS (NCOMP, M, ALEFT, ARIGHT, ZETA, IPAR, LTOL,
     .               TOL, FIXPNT, ISPACE, FSPACE, IFLAG,
     .               FSUB, DFSUB, GSUB, DGSUB, SOLUTN)
C
         IF (IFLAG .NE. 1)  STOP
C        HERE WE MAY PRINT OUT OR PLOT VALUES OF THE
C        SOLUTION FOR EACH OF THE 3 PROBLEMS SOLVED
  777 CONTINUE
      STOP
      END
```

The obtained G and H for $n = 0.2$, $s = 0.05$, $L = 200$ are graphed in Figure 2.

5. ADDITIONAL COMMENTS

The code COLSYS has been designed to handle a variety of difficulties arising in practical problems. Consequently, documentation is provided to facilitate sophisticated as well as straightforward usages. Example 1 presents a simple use of the code, while Examples 2 and 3 demonstrate some of the more advanced features.

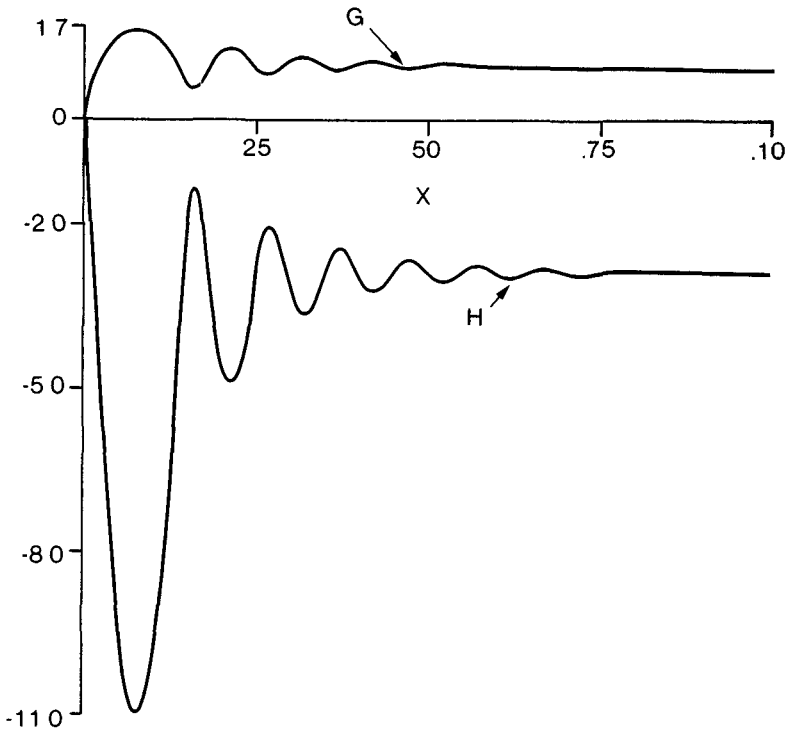

Fig. 2. Example 3.

In general COLSYS performs effectively [1], [2], [3]. Still, a number of additional features could be incorporated in the code to improve its performance and reliability. Specifically, there is currently no facility for handling singular Jacobians or detecting excessive roundoff errors. For very large problems the storage requirements may become excessive (as is typical of non-initial-value codes), and proper modification of the linear system solver can reduce these requirements somewhat [2]. Furthermore, the only error tolerances that have been extensively tested are in the intermediate range 10^{-4} to 10^{-9} in ~ 16 decimal digit arithmetic. Also, harmless underflows (occurring particularly in the B-spline subroutines) are ignored. We intend to address some of these issues in the near future.

COLSYS does not incorporate automatic continuation; however, enough information is returned to allow convenient implementation of simple continuation by the user, as demonstrated in Example 3. For this purpose, use of IPAR(9) = 3 is recommended so that the first mesh for the current problem is constructed from every second point of the final mesh for the former problem and the initial approximation for the current nonlinear iteration is just the solution of the previous problem. Thus continuation is not used for linear problems.

Unlike other codes the mesh selection procedure is free to change every interior mesh point during refinement. This provides for flexible adjustment of meshes to incorporate extreme characteristics of the solution with relatively small meshes,

as demonstrated in Example 2. In some cases, however, it may be desirable to alter this process. This can be done in two ways:

(a) Setting IPAR(8) = 2 results in repeated mesh halving and prevents mesh redistribution altogether. An initial mesh can be defined by the user in FSPACE (see Section 4). It should be pointed out, however, that usually the automatic mesh selector produces better meshes than user-generated ones as the error becomes small.
(b) Using IPAR(11) and the array FIXPNT, certain specified points are forced to be part of every mesh. This can be used to indicate boundary layers, interior boundary points (interfaces) where the solution has less smoothness, or any points where the user does not want the differential equation evaluated. However, fixed points should be used sparingly, as they may downgrade the performance of the mesh refinement algorithm.

Even given the power of the automatic mesh selection, as demonstrated in the last section, a good initial mesh provided by the user (using IPAR(8) = 1) can improve performance considerably. In fact, the nonlinear iteration may not converge on an inadequate initial mesh, causing repeated mesh halvings until a sufficiently fine mesh in the region(s) of difficulty is obtained. It can also happen that an initial user-provided mesh having a few well-placed points gives a crude approximation such that the next automatically selected mesh "loses" this desirable placement. It is important to realize that a sufficiently good start for the nonlinear iteration must sometimes consist of both a good initial solution profile and an adequate initial mesh. Failure to have one of these may cause the code to repeatedly halve the mesh without obtaining convergence of the nonlinear iteration until storage limitations are reached. This should not be interpreted by the user as a storage problem.

The user is required to provide two Jacobians: a rectangular NCOMP × MSTAR matrix DF(I, J) in subroutine DFSUB and a set of MSTAR vectors DG of length MSTAR in subroutine DGSUB. The partial derivatives required may at times be difficult or impossible to evaluate, and they can instead be approximated by numerical differentiation. The following FORTRAN segment could then be used for DFSUB:

```
      DIMENSION DF(NCOMP, MSTAR), WORK1(NCOMP),
     + WORK2(NCOMP), Z(MSTAR)
C     THE DIMENSION VALUES HAVE TO BE EXPLICITLY INSERTED
      EPS = 1.E - 7
      DO 10 J = 1, MSTAR
         Z(J) = Z(J) + EPS
         CALL FSUB(X, Z, WORK1)
         Z(J) = Z(J) - 2. * EPS
         CALL FSUB(X, Z, WORK2)
         Z(J) = Z(J) + EPS
         DO 10 I = 1, NCOMP
   10 DF(I, J) = (WORK1(J) - WORK2(J)) * .5/EPS
```

Here, EPS is roughly the square root of the machine unit roundoff; a more sophisticated differencing could adapt EPS to the size of Z(J). While the numer-

ical differentiation gives only an approximate Jacobian and may be computationally less efficient, the accuracy of the final computed solution is generally not affected.

To control the nonlinear iteration, we suggest that a user always use the regular mode (IPAR(10) = 0) first. If no convergence occurs, particularly after convergence on a previous mesh has been obtained and/or a good initial approximation was given, then the sensitive mode (IPAR(10) = 1) may be useful.

Unlike other codes, COLSYS is designed to handle a mixed-order system without explicitly converting to a first-order system (as long as the orders are ≤ 4). For collocation, this direct treatment is more efficient both in terms of storage requirements and execution time. However, the requirement $k > m_d$ causes the order of the method to be at least $2m_d + 1$, and for higher orders the error estimates may tend to be less reliable (cf. Example 1). The restriction $m_d \leq 4$, required for efficient implementation, is adequate for most practical problems. An equation of order higher than 4 should be converted by the user to lower order ones that satisfy the above restriction, with d being kept minimal for efficient execution (cf. [2]).

An important feature of the code is that the approximate solution values at any points, and hence solution plots, are readily available. Plots with high resolution such as those in Figures 1 and 2 can be easily generated by using the mesh points plus a sufficient number of equally spaced points. In fact, solution values at mesh points are frequently more accurate than elsewhere, since the theoretical rate of convergence at these points is higher [2].

For the value of IPAR(2) = k, the number of collocation points per subinterval, a reasonable choice is the default value $k = \max(m_d + 1, 5 - m_d)$, obtained by the user setting IPAR(2) = 0. Note that the order of the method does not vary during a run of COLSYS.

The restrictions d = NCOMP ≤ 20 and m^* = MSTAR ≤ 40 currently imposed are sufficient to accommodate most usages. In case they need to be increased, the following changes should be made in COLSYS: (a) For arrays in every subroutine currently dimensioned 20 and 40, modify their dimensions to the new limits on NCOMP and MSTAR, respectively; (b) modify the checks on input correctness in subroutine COLSYS appropriately; (c) in LSYSLV, change the dimension of DF from 800 to the new limit on NCOMP * MSTAR; (d) in BLDBLK, change the dimension of BASEF from 620 to the new desired limit on

$$8 * \text{MSTAR} + 7 * \text{NCOMP} + \max \sum_{J=1}^{\text{NCOMP}} \text{M(J)}^2.$$

Finally, various output options are available for COLSYS [using IPAR(7)]. Restricted output is displayed in Example 1. The "full output" option also displays the input data, nonlinear iteration and mesh-selection control parameters, and the solution values at the mesh points for intermediate meshes.

Note Added in Proof: Many practical problems that are not in the standard form accepted by COLSYS can be converted into such form. For a survey of conversion devices see, for example, [4a].

REFERENCES

1. ASCHER, U. Solving boundary value problems with a spline-collocation code. *J Comput. Phys. 34* (1980), 401–413.
2. ASCHER, U., CHRISTIANSEN, J., AND RUSSELL, R.D. A collocation solver for mixed order systems of boundary value problems. Tech. Rep. 77-13, Computer Science Dep., Univ British Columbia, Vancouver, Canada, 1977. (See also *Math. Comput. 33* (1979), 659–679.)
3. ASCHER, U., CHRISTIANSEN, J., AND RUSSELL, R.D. COLSYS–A collocation code for boundary value problems. In *Codes for Boundary Value Problems*, B. Childs et al. (Eds.), Lecture Notes in Computer Science 76, Springer-Verlag, New York, 1979
4. ASCHER, U., AND RUSSELL, R.D. Evaluation of B-splines for solving systems of boundary value problems. Tech. Rep. 77-14, Computer Science Dep., Univ. British Columbia, Vancouver, Canada, 1977.

4a. ASCHER, U., AND RUSSELL, R.D. Reformation of boundary value problems in "standard" form. *SIAM Rev 23* (April 1981).

5. BULIRSCH, R., STOER, J , AND DEUFLHARD, P. Numerical solution of nonlinear two-point boundary value problems I. *Numer. Math. Handbook Series Approximation* (1976).

5a. CODES FOR BOUNDARY VALUE PROBLEMS B. Childs et al (Eds.), Lecture Notes in Computer Science 76, Springer-Verlag, New York, 1979.

6. DE BOOR, C. On calculating with B-splines. *J. Approx. Theory 6* (1972), 50–62
7. DE BOOR, C., AND WEISS, R. SOLVEBLOK: A package for solving almost block diagonal linear systems. *ACM Trans Math. Softw. 6*, 1 (March 1980), 80–87.
8. DEUFLHARD, P. A stepsize control for continuation methods with special application to multiple shooting techniques. TUM-Math-7627, Munchen, Germany, 1976.
9. DIEKOFF, H.J., LORY, P., OBERLE, H.J., PESCH, H.J., RENTROP, P., AND SEYDEL, R. Comparing routines for the numerical solution of initial value problems of ordinary differential equations in multiple shooting. *Numer. Math. 27* (1977), 449–469.
10. GAWAIN, T.H., AND BALL, R.E. Improved finite difference formulas for boundary value problems. *Int. J. Numer. Meth. Eng. 12* (1978), 1151–1160.
11. GLADWELL, I. Shooting codes in the NAG library. In B. Childs et al. (Eds.), Lecture Notes in Computer Science 76, Springer-Verlag, New York, 1979.
12. HOLT, J.F. Numerical solution of nonlinear two-point boundary problems by finite difference methods. *Commun. ACM 7*, 6(1964), 366–373.
13. LENTINI, M., AND PEREYRA, V An adaptive finite difference solver for nonlinear two point boundary problems with mild boundary layers. *SIAM J. Numer Anal 14* (1977), 91–111.
14 LENTINI, M., AND PEREYRA, V. PASVA3: An adaptive finite difference program for first order, nonlinear ordinary boundary problems. In B. Childs et al. (Eds.), Lecture Notes in Computer Science 76, Springer-Verlag, New York, 1979.
15. SCOTT, M.L., AND WATTS, H.A. Computational solutions of linear two-point boundary problems via orthonormalization. *SIAM J. Numer. Anal. 14* (1977), 40–70.
16. SCOTT, M.L., AND WATTS, H.A. Superposition, orthonormalization, quasilinearization and two-point boundary value problems. In B. Childs et al. (Eds.), Lecture Notes in Computer Science 76, Springer-Verlag, New York, 1979.
17. WAN, F. The dimpling of spherical caps. Tech. Rep. 78-6, Inst. of Applied Mathematics, Univ. British Columbia, Vancouver, Canada, 1978.

Received February 1979; revised April 1979; accepted October 1979.