name of a user-defined function subprogram. The actual name of the user-supplied function must be declared in an EXTERNAL statement in the program calling BISECT. Named constants in BISECT are given in the PARAMETER statement. If the precision of the program is to be changed, then the constants can be easily changed to double precision, since they are all given in this one statement.

Lacking in the subroutine is any attempt to check whether the error tolerance EPS is realistic for the length of the mantissa in the computer arithmetic being used. This was left out to keep the program simple, but a realistic rootfinding program in a computer library would need to include such a check.

```
C     THIS IS A DEMONSTRATION PROGRAM FOR THE
C     ROOTFINDING SUBROUTINE 'BISECT'.
C
      EXTERNAL FCN
C
C     INPUT PROBLEM PARAMETERS.
10    PRINT *, ' WHAT ARE A, B, EPSILON?'
      PRINT *, ' TO STOP, LET EPSILON=0.'
      READ *, A, B, EPSLON
      IF(EPSLON .EQ. 0.0) STOP
C
C     CALCULATE ROOT
      CALL BISECT(FCN, A ,B ,EPSLON, ROOT, IER)
C
C     PRINT ANSWERS.
      PRINT 1000, A, B, EPSLON
      PRINT 1001, ROOT, IER
      GO TO 10
1000  FORMAT(/,' A=',E11.4,5X,'B=',E11.4,5X,'EPSILON=',E9.3)
1001  FORMAT(' ROOT=',E14.7,5X,'IER=',I1)
      END

      FUNCTION FCN(X)
C
      FCN = EXP(-X) - X
      RETURN
      END

      SUBROUTINE BISECT(F, A, B, EPS, ROOT, IER)
C
C     THE PROGRAM USES THE BISECTION METHOD TO SOLVE
C     THE EQUATION
C                   F(X) = 0.
C     THE SOLUTION IS TO BE IN [A,B] AND IT IS ASSUMED
C     THAT
C               F(A)*F(B) .LE. 0.
C     THE SOLUTION IS RETURNED IN ROOT, AND IT IS TO
```

```
C     BE IN ERROR BY AT MOST EPS.
C
C     IER IS AN ERROR INDICATOR.
C     IF IER=0 ON COMPLETION OF THE ROUTINE, THEN THE
C        SOLUTION HAS BEEN COMPUTED SATISFACTORILY.
C     IF IER=1, THEN F(A)*F(B) WAS GREATER THAN 0,
C        CONTRARY TO ASSUMPTION.
C
      PARAMETER(ZERO=0.0, ONE=1.0, TWO=2.0)
C
C     INITIALIZE.
      FA = F(A)
      FB = F(B)
      SFA = SIGN(ONE, FA)
      SFB = SIGN(ONE, FB)
      IF(SFA*SFB .GT. ZERO) THEN
C         THE CHOICE OF A AND B IS IN ERROR.
          IER = 1
          RETURN
      END IF
C
C     CREATE A NEW VALUE OF C, THE MIDPOINT OF [A,B].
10    C = (A + B)/TWO
      IF(ABS(B-C) .LE. EPS) THEN
C         C IS AN ACCEPTABLE SOLUTION OF F(X)=0.
20        ROOT = C
          IER = 0
          RETURN
      END IF
C
C     THE VALUE OF C WAS NOT SUFFICIENTLY ACCURATE.
C     BEGIN A NEW ITERATION.
      FC = F(C)
      IF(FC .EQ. ZERO) GO TO 20
      SFC = SIGN(ONE, FC)
      IF(SFB*SFC .GT. ZERO) THEN
C         THE SOLUTION IS IN [A,C].
          B = C
          SFB = SFC
      ELSE
C         THE SOLUTION IS IN [C,B].
          A = C
          SFA = SFC
      END IF
      GO TO 10
      END
      SUBROUTINE NEWTON(F, DF, XINIT, EPS, ITMAX, ROOT, IER)
C
C     THIS ROUTINE CALCULATES A ROOT OF
C               F(X) = 0
C     USING NEWTON'S METHOD.
C
C     INPUT PARAMETERS:
C     F,DF:  THE NAMES OF FUNCTION SUBPROGRAMS FOR COMPUTING
C            F(X) AND DF(X)=F'(X).  THE ACTUAL NAMES USED IN
C            CALLING SUBROUTINE NEWTON MUST BE DECLARED IN
C            AN EXTERNAL STATEMENT IN THE CALLING PROGRAM.
C     XINIT: AN INITIAL GUESS OF THE SOLUTION.
C     EPS:   THE DESIRED ERROR TOLERANCE.  THE TEST FOR
C            CONVERGENCE IS
C                   ABS(X1-X0) .LE. EPS,
C            WHERE X1 IS THE CURRENT NEWTON ITERATE AND X0
C            IS THE PRECEDING ITERATE.
C     ITMAX: AN UPPER LIMIT ON THE NUMBER OF ITERATES
C            TO BE COMPUTED. WHEN THE NUMBER OF ITERATES
C            REACHES THIS NUMBER, THE PROGRAM IS TERMINATED.
C            WARNING: ITMAX IS ALTERED ON OUTPUT.
C
C     OUTPUT PARAMETERS:
C     ROOT:  CONTAINS THE COMPUTED VALUE OF THE SOLUTION.
C            REGARDLESS OF HOW THE ROUTINE IS TERMINATED,
C            SUCCESSFULLY OR OTHERWISE, ROOT WILL CONTAIN
C            THE MOST RECENTLY COMPUTED NEWTON ITERATE.
C     ITMAX: SET TO THE NUMBER OF ITERATES COMPUTED IN
C            THE ROUTINE.
C     IER:   AN ERROR INDICATOR.
C        =0  MEANS A SUCCESSFUL COMPLETION OF NEWTON.
C        =1  MEANS ITMAX ITERATES WERE COMPUTED AND THE
C            ROUTINE WAS ABORTED.
C        =2  MEANS THAT THE DERIVATIVE DF(X) BECAME ZERO
C            AT SOME NEWTON ITERATE, AND THE ROUTINE
C            WAS ABORTED.
C
      PARAMETER(ZERO=0.0)
C
C     INITIALIZE.
      X0 = XINIT
      ITNUM = 1
C
C     BEGIN MAIN LOOP.
10    DENOM = DF(X0)
      IF(DENOM .EQ. ZERO) THEN
C        DERIVATIVE EQUALS ZERO. TERMINATE ITERATION.
```

```
         IER = 2
         ROOT = X0
         ITMAX = ITNUM-1
         RETURN
      END IF
C
C     COMPUTE NEWTON ITERATION.
      X1 = X0 - F(X0)/DENOM
C
      IF(ABS(X1-X0) .LE. EPS) THEN
C        ERROR TEST SATISFIED.
         IER = 0
         ROOT = X1
         ITMAX = ITNUM
         RETURN
      END IF
C
      IF(ITNUM .LE. ITMAX) THEN
C        INITIALIZE FOR ANOTHER LOOP.
         ITNUM = ITNUM + 1
         X0 = X1
         GO TO 10
      ELSE
C        ITMAX ITERATES HAVE BEEN COMPUTED. TERMINATE.
         IER = 1
         ROOT = X1
         RETURN
      END IF
      END
```

PROBLEMS

1. Carry out the Newton iteration (4.12) with the two initial guesses $x_0 = 1.0$ and $x_0 = 2.0$. Compare the results with Table 4.2.

2. Using Newton's method, find the roots of the equations in problem 1, Section 4.1 in this chapter. Use an error tolerance of $\epsilon = 10^{-6}$.

3. (a) On most computers, the computation of $\sqrt{a}$ is based on Newton's method. Set up the Newton iteration for solving $x^2 - a = 0$, and show that it can be written in the form

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right), \qquad n \geq 0$$

A Simpson's Rule Program

We give a program for Simpson's rule, for $n = 2, 4, 8, 16, \ldots, 512$ subdivisions. When n is doubled to $2n$, all of the function values occurring in $S_n(f)$ are also used in computing $S_{2n}(f)$. We also allow for a variety of integrands in the function F, with the user specifying the integrand through the variable NUMF. Note that double-precision constants should always be terminated by D0, as a matter of good (and safe) programming practice, when writing double-precision programs. The program comments should explain the organization of the program.

```
C     TITLE: EXAMPLE OF SIMPSON'S NUMERICAL INTEGRATION METHOD.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (ZERO = 0.0D0, TWO=2.0D0, THREE=3.0D0, FOUR=4.0D0)
      COMMON/PARAM/NUMF
C
10    PRINT *, ' GIVE NUMBER OF INTEGRAND.  GIVE ZERO TO STOP.'
      READ *, NUMF
      IF(NUMF .EQ. 0) STOP
C
      PRINT *, ' GIVE INTEGRATION LIMITS A AND B.'
      READ *, A,B
      PRINT *, ' GIVE TRUE ANSWER, IF KNOWN.'
      PRINT *, ' IF UNKNOWN, GIVE ZERO.'
      READ *, TRUE
      PRINT *, ' '
      PRINT *, ' NUMBER OF INTEGRAND = ', NUMF
      PRINT *, ' INTEGRATION LIMITS = ', A, B
      PRINT *, ' TRUE = ', TRUE
C
C     PERFORM SIMPSON'S RULE.
C     INITIALIZE.
      SUMEND = F(A) + F(B)
      SUMODD = ZERO
      SUMEVN = ZERO
C
      DO 30 J=1,9
C       SET NUMBER OF NODE-POINTS AND STEPSIZE.
        N = 2**J
        H = (B - A)/N
        SUMEVN = SUMEVN + SUMODD
        SUMODD = ZERO
C       CREATE NEW NODES AND INTEGRAND VALUES.
        DO 20 K=1,N-1,2
          X = A + K*H
20        SUMODD = SUMODD + F(X)
```

```
C        CREATE SIMPSON RULE WITH N SUBDIVISIONS.  CALCULATE ERROR.
C        PRINT ERROR AND RATE OF DECREASE.
         SIMPSN = H*(SUMEND + FOUR*SUMODD + TWO*SUMEVN)/THREE
         ERROR = TRUE - SIMPSN
         IF(J .EQ. 1) THEN
             PASTER = ERROR
             PRINT 1000, N, SIMPSN, ERROR
         ELSE
             RATIO = PASTER/ERROR
             PASTER = ERROR
             PRINT 1001, N, SIMPSN, ERROR, RATIO
         END IF
1000     FORMAT(' N=',I3,3X,'SIMPSN=',1PD17.10,3X,'ERROR=',D10.3)
1001     FORMAT(' N=',I3,3X,'SIMPSN=',1PD17.10,3X,'ERROR=',D10.3,
     *          3X,'RATIO=',D10.3)
30        CONTINUE
       GO TO 10
       END

       FUNCTION F(X)
C
       IMPLICIT DOUBLE PRECISION(A-H,O-Z)
       COMMON/PARAM/NUMFCN
C
       GO TO (10,20) NUMFCN
10     F = EXP(-X*X)
       RETURN
20     F = EXP(COS(X))
       RETURN
       END
```

PROBLEMS

1. Compute $T_4(f)$ and $S_4(f)$ for the integral I in (7.5). Compute the errors $I - T_4$ and $I - S_4$; compare them to the errors $I - T_2$ and $I - S_2$, respectively.

2. Using the general form of the Simpson program, prepare a program for the trapezoidal rule. With it, prepare a table of values of $T_n(f)$ for $n = 2, 4, 8, \dots, 512$ for the following integrals. Also find the errors and the ratios by which the errors decrease.

   (a) $\displaystyle\int_0^{\pi} e^x \cos(4x)dx = \frac{e^{\pi} - 1}{17}$

   (b) $\displaystyle\int_0^1 x^{5/2}dx = \frac{2}{7}$

   (c) $\displaystyle\int_0^5 \frac{dx}{1 + (x - \pi)^2} = \tan^{-1}(5 - \pi) + \tan^{-1}(\pi)$

$$\begin{aligned} AS(x) &= AS(A \to U) + AS(b \to g) + AS(g \to x) \\ &= \frac{n(n-1)(2n-1)}{6} + \frac{n(n-1)}{2} + \frac{n(n-1)}{2} \\ &= \frac{n(n-1)(2n+5)}{6} \\ MD(x) &= \frac{n(n^2+3n-1)}{3} \end{aligned} \tag{8.39}$$

Since AS and MD are almost the same in all of these counts, only MD is discussed. These operations are also slightly more expensive in running time. For larger values of n, the operation count for Gaussian elimination is about $\frac{1}{3}n^3$. This means that as n is doubled, the cost of solving the linear system goes up by a factor of 8. In addition, most of the cost of Gaussian elimination is in the elimination step, $A \to U$, since for the remaining steps

$$MD(b \to g) + MD(g \to x) = \frac{n(n-1)}{2} + \frac{n(n+1)}{2} = n^2 \tag{8.40}$$

Thus, once the $A \to U$ step has been completed, it is much less expensive to solve the linear system. We return to this later in Section 8.5, where we give a simple and inexpensive way to estimate the error in the answer obtained by Gaussian elimination.

Computer Programs

Much mathematical research and program development has been carried out on producing efficient and accurate programs for Gaussian elimination, and your computer center should possess such a code. Some of the best of these codes are contained in the book *Linpack: User's Guide* by Dongarra, Bunch, Moler, and Stewart (1979). Good programs will carry out the elimination in the most efficient manner and will arrange operations to minimize rounding errors. They will also offer error estimates, and this is needed because many linear systems are not as well behaved as our examples have been.

We give a subroutine LINSYS for implementing Gaussian elimination as described in this section. It is intended as a pedagogical tool; actual production computing should use a code of the type described in the preceding paragraph. The comment statements in LINSYS describe the values of the input and output variables. The use of matrix notation is based on ideas introduced in the next section.

```
C     TITLE: THIS IS A DEMO PROGRAM FOR SUBROUTINE LINSYS.
C
C     IT WILL SOLVE A LINEAR SYSTEM A*X=B, GIVEN BY THE USER.
C
      PARAMETER (MAXN=20)
      DIMENSION A(MAXN,MAXN), B(MAXN)
C
C     INPUT ORDER OF LINEAR SYSTEM.
10    PRINT *, ' GIVE THE ORDER OF THE LINEAR SYSTEM.'
      PRINT *, ' THE UPPER BOUND ON THE ORDER IS ', MAXN
      PRINT *, ' GIVE ZERO TO STOP.'
      READ *, N
      IF(N .EQ. 0) STOP
C
C     INPUT LINEAR SYSTEM.
      PRINT *, ' GIVE THE COEFFICIENTS OF THE LINEAR SYSTEM,'
      PRINT *, ' ONE EQUATION AT A TIME.'
      PRINT *, ' CONCLUDE EACH EQUATION WITH ITS RIGHT-HAND CONSTANT.'
      DO 20 I=1,N
        PRINT *, ' GIVE COEFFICIENTS OF EQUATION',I
20      READ *, (A(I,J), J=1,N), B(I)
C
C     SOLVE THE LINEAR SYSTEM.
      CALL LINSYS(A,B,N,MAX_N,IER)
C
C     PRINT THE RESULTS.
      PRINT 1000, N, IER
1000  FORMAT(///,' N=',I2,5X,'IER=',I3,//,'  I       SOLUTION',/)
      PRINT 1001, (I,B(I), I=1,N)
1001  FORMAT(I3,1PE20.10)
      GO TO 10
      END

      SUBROUTINE LINSYS(MAT,B,N,MD,IER)
C     -----------------
C
C     THIS ROUTINE SOLVES A SYSTEM OF LINEAR EQUATIONS
C             A*X = B
C     THE METHOD USED IS GAUSSIAN ELIMINATION WITH
C     PARTIAL PIVOTING.
C
C     INPUT:
C     THE COEFFICIENT MATRIX  A  IS STORED IN THE ARRAY MAT.
C     THE RIGHT SIDE CONSTANTS ARE IN THE ARRAY  B.
C     THE ORDER OF THE LINEAR SYSTEM IS  N.
C     THE VARIABLE  MD  IS THE NUMBER OF ROWS THAT  MAT
C     IS DIMENSIONED AS HAVING IN THE CALLING PROGRAM.
C     THE SIZE OF  MAXPIV, GIVEN BELOW, MUST BE GREATER
C     THAN  N.  IF NOT, IT IS A FATAL ERROR.
C
C     OUTPUT:
C     THE ARRAY  B  CONTAINS THE SOLUTION  X.
```

```
C     MAT CONTAINS THE UPPER TRIANGULAR MATRIX  U
C     OBTAINED BY ELIMINATION.  THE ROW MULTIPLIERS
C     USED IN THE ELIMINATION ARE STORED IN THE
C     LOWER TRIANGULAR PART OF  MAT.
C     IER=0 MEANS THE MATRIX  A  WAS COMPUTATIONALLY
C     NONSINGULAR, AND THE GAUSSIAN ELIMINATION
C     WAS COMPLETED SATISFACTORILY.
C     IER=1 MEANS THAT THE MATRIX  A  WAS
C     COMPUTATIONALLY SINGULAR.
C
      INTEGER PIVOT
      REAL MULT, MAT
      PARAMETER(MAXPIV=100)
      DIMENSION MAT(MD,*), B(*), PIVOT(MAXPIV)
C
C     CHECK SIZE OF N VERSUS MAXPIV
      IF(MAXPIV .LT. N) THEN
          PRINT *, 'THE VARIABLE MAXPIV IN SUBROUTINE LINSYS MUST BI
          PRINT *, 'INCREASED.  THIS IS A FATAL ERROR.'
          STOP
      END IF
C
C     BEGIN ELIMINATION STEPS.
      DO 40 K=1,N-1
C       CHOOSE PIVOT ROW.
        PIVOT(K) = K
        AMAX = ABS(MAT(K,K))
        DO 10 I=K+1,N
          ABSA = ABS(MAT(I,K))
          IF(ABSA .GT. AMAX) THEN
              PIVOT(K) = I
              AMAX = ABSA
          END IF
10        CONTINUE
C
        IF(AMAX .EQ. 0.0) THEN
C           COEFFICIENT MATRIX IS SINGULAR.
            IER = 1
            RETURN
        END IF
C
        IF(PIVOT(K) .NE. K) THEN
C           SWITCH ROWS K AND PIVOT(K).
            I = PIVOT(K)
            TEMP = B(K)
            B(K) = B(I)
            B(I) = TEMP
            DO 20 J=K,N
              TEMP = MAT(K,J)
              MAT(K,J) = MAT(I,J)
20            MAT(I,J) = TEMP
        END IF
C
C       PERFORM STEP #K OF ELIMINATION.
        DO 30 I=K+1,N
          MULT = MAT(I,K)/MAT(K,K)
          MAT(I,K) = MULT
          B(I) = B(I) - MULT*B(K)
          DO 30 J=K+1,N
30          MAT(I,J) = MAT(I,J) - MULT*MAT(K,J)
40        CONTINUE
C
      IF(MAT(N,N) .EQ. 0.0) THEN
C         COEFFICIENT MATRIX IS SINGULAR.
          IER = 1
          RETURN
      END IF
C
C     SOLVE FOR SOLUTION X USING BACK SUBSTITUTION.
      DO 60 I=N,1,-1
        SUM = 0.0
        DO 50 J=I+1,N
50        SUM = SUM + MAT(I,J)*B(J)
60      B(I) = (B(I) - SUM)/MAT(I,I)
      IER = 0
      RETURN
      END
```

PROBLEMS

1. Solve the following linear systems by using Gaussian elimination without pivoting. The systems are specified by A and b, with the system then written as in (8.22).

(a)

$$A = \begin{bmatrix} 2 & 1 & -1 \\ 4 & 0 & -1 \\ -8 & 2 & 2 \end{bmatrix}, \qquad b = \begin{bmatrix} 6 \\ 6 \\ -8 \end{bmatrix}$$

(b)

$$A = \begin{bmatrix} 2 & 1 & -1 & -2 \\ 4 & 4 & 1 & 3 \\ -6 & -1 & 10 & 10 \\ -2 & 1 & 8 & 4 \end{bmatrix}, \qquad b = \begin{bmatrix} 2 \\ 4 \\ -5 \\ 1 \end{bmatrix}$$

A Program for Tridiagonal Systems

The following Fortran subroutine solves tridiagonal systems by means of the method (8.78) to (8.82). The coefficients $\{a_j , b_j , c_j\}$ are given by the user in the one-dimensional arrays A, B, and C, and the right-hand constants are supplied in F. On exit, the arrays A and B contain the constants $\{\alpha_j, \beta_j\}$ of (8.80), and F contains the solution x. In many applications, a tridiagonal system is solved with many right sides f. To allow for this option, the variable IFLAG indicates whether or not the LU factorization has already been computed and stored in the arrays, A, B, and C. See the comment statements in the program for additional information.

```
      SUBROUTINE TRIDGL(A,B,C,F,N,IFLAG,IER)
C     -----------------
C
C     THIS SOLVES A TRIDIAGONAL SYSTEM OF LINEAR EQUATIONS  M*X=F.
C
C     INPUT:
C     THE ORDER OF THE LINEAR SYSTEM IS GIVEN BY  N.
C     THE SUBDIAGONAL,DIAGONAL, AND SUPERDIAGONAL OF  M  ARE GIVEN
C     BY THE ARRAYS  A, B, AND C, RESPECTIVELY. MORE PRECISELY,
C        M(I,I-1) = A(I),  I=2,...,N
C        M(I,I)   = B(I),  I=1,...,N
C        M(I,I+1) = C(I),  I=1,...,N-1
C     IFLAG=0 MEANS THAT THE ORIGINAL MATRIX  M  IS GIVEN AS
C     SPECIFIED ABOVE.
C     IFLAG=1 MEANS THAT THE LU FACTORIZATION OF  M  IS ALREADY KNOWN
C     AND IS STORED IN  A,B, AND C. THIS WILL HAVE BEEN ACCOMPLISHED
C     BY A PREVIOUS CALL TO THIS SUBROUTINE.
C
C     OUTPUT:
C     UPON EXIT, THE LU FACTORIZATION OF  M  WILL BE STORED
C     IN  A,B, AND C.
C     THE SOLUTION VECTOR  X  WILL BE RETURNED IN  F.
C     IER=0 MEANS THE PROGRAM WAS COMPLETED SATISFACTORILY.
C     IER=1 MEANS THAT A ZERO PIVOT ELEMENT WAS ENCOUNTERED, AND
C     NO SOLUTION WAS ATTEMPTED.
C
      PARAMETER (ZERO=0.0)
      DIMENSION A(*), B(*), C(*), F(*)
C
      IF(IFLAG .EQ. 0) THEN
C         COMPUTE LU FACTORIZATION OF MATRIX M.
          DO 10 J=2,N
            IF(B(J-1) .EQ. ZERO) THEN
                IER = 1
                RETURN
```

```
               END IF
               A(J) = A(J)/B(J-1)
10             B(J) = B(J) - A(J)*C(J-1)
             IF(B(N) .EQ. ZERO) THEN
                IER = 1
                RETURN
             END IF
         END IF
C
C        COMPUTE SOLUTION X TO M*X=F.
         DO 20 J=2,N
20         F(J) = F(J) - A(J)*F(J-1)
         F(N) = F(N)/B(N)
         DO 30 J=N-1,1,-1
30         F(J) = (F(J) - C(J)*F(J+1))/B(J)
         IER = 0
         RETURN
         END
```

PROBLEMS

1. Recall the solution of the system (8.12) in Section 8.2. Using that computation, give the LU factorization of the matrix of coefficients of that system.

2. Calculate the LU factorization of the matrices in problem 1 of Section 8.2.

3. Consider the proof of Theorem 8.3 when $n = 3$. To do so, recall the general derivation (8.15) to (8.21) for Gaussian elimination with three equations. Form the product

$$LU = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ 0 & 0 & a_{33}^{(3)} \end{bmatrix}$$

Use the definitions of m_{ij} and $a_{ij}^{(i)}$ to simplify the results.

4. Recall the definition of elementary matrices from problem 20 of Section 8.3. Define

$$E_1 = \begin{bmatrix} 1 & 0 & 0 \\ -m_{21} & 1 & 0 \\ -m_{31} & 0 & 1 \end{bmatrix}, \qquad E_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -m_{32} & 1 \end{bmatrix}$$

Define $A_2 = E_1 A$, $A_3 = E_2 A_2$. Show $A_3 = U$, the upper triangular matrix obtained by Gaussian elimination. Thus $E_2 E_1 A = U$. Use this to show $A = LU$, with L derived from E_2 and E_1. How would this proof generalize to A, a matrix of order n?

computing. The part of the program actually devoted to Euler's method is relatively small; but the program's structure allows it to be easily extended to the methods taken up in later sections.

```
C     TITLE: DEMONSTRATION OF EULER'S METHOD.
C
C     THIS SOLVES THE INITIAL VALUE PROBLEM
C        Y'(X) = F(X,Y(X)) ,  X0 .LE. X .LE. B ,  Y(X0)=Y0.
C     THE FUNCTION F(X,Z) IS DEFINED BELOW, ALONG WITH THE TRUE
C     SOLUTION Y(X).  THE NUMBER OF THE PROBLEM TO BE SOLVED
C     IS SPECIFIED BY THE INPUT VARIABLE 'NUMDE', WHICH IS USED
C     IN THE FUNCTIONS 'F' AND 'Y'.  THE PROGRAM WILL REQUEST
C     THE PROBLEM PARAMETERS, ALONG WITH THE VALUES OF 'H' AND
C     'IPRINT'.  'H' IS THE STEPSIZE, AND 'IPRINT' IS THE NUMBER
C     OF STEPS BETWEEN EACH PRINTING OF THE SOLUTION.
C     USE H=0 AND NUMDE=0 TO STOP THE PROGRAM.
C
      PARAMETER (ZERO=0.0)
      COMMON/BLOCKF/NUMDE
C
C     INPUT PROBLEM PARAMETERS.
10    PRINT *, ' NUMDE = ?'
      PRINT *, ' GIVE ZERO TO STOP.'
      READ *, NUMDE
      IF(NUMDE .EQ. 0) STOP
      PRINT *, ' GIVE X0, B, AND Y0.'
      READ *, XZERO, B, YZERO
C
20    PRINT *, ' GIVE H AND IPRINT.'
      READ *, H, IPRINT
      IF(H .EQ. ZERO) GO TO 10
C
C     INITIALIZE.
      X0 = XZERO
      Y0 = YZERO
      PRINT 1000, NUMDE, XZERO, B, YZERO, H, IPRINT
1000  FORMAT(//,' EQUATION',I2,5X,'XZERO =',1PE9.2,5X,'B =',E9.2,
     *       5X,'YZERO =',E12.5,/,' STEPSIZE =',E10.3,5X,
     *       'PRINT PARAMETER =',I3,/)
C
C     BEGIN THE MAIN LOOP FOR COMPUTING THE SOLUTION OF
C     THE DIFFERENTIAL EQUATION.
30    DO 40 K=1,IPRINT
        X1 = X0 + H
        IF(X1 .GT. B) GO TO 20
        Y1 = Y0 + H*F(X0,Y0)
```

```
          X0 = X1
 40       Y0 = Y1
 C
 C     CALCULATE ERROR AND PRINT RESULTS.
       TRUE = Y(X1)
       ERROR = TRUE - Y1
       PRINT 1001, X1, Y1, ERROR
 1001  FORMAT(' X =',1PE10.3,5X,'Y(X) =',E17.10,5X,'ERROR =',E9.2)
       GO TO 30
       END

       FUNCTION F(X,Z)
 C
 C     THIS DEFINES THE RIGHT SIDE OF
 C     THE DIFFERENTIAL EQUATION.
 C
       PARAMETER (ONE=1.0, TWO=2.0)
       COMMON/BLOCKF/NUMDE
 C
       GO TO (10,20,30), NUMDE
 10    F = -Z
       RETURN
 20    F = (Z + X*X - TWO)/(X + ONE)
       RETURN
 30    F = COS(Z)**2
       RETURN
       END

       FUNCTION Y(X)
 C
 C     THIS GIVES THE TRUE SOLUTION OF
 C     THE INITIAL VALUE PROBLEM.
 C
       PARAMETER (ONE=1.0, TWO=2.0)
       COMMON/BLOCKF/NUMDE
 C
       GO TO(10,20,30), NUMDE
 10    Y = EXP(-X)
       RETURN
 20    X1=X + ONE
       Y = X*X - TWO*(X1*LOG(X1) - X1)
       RETURN
 30    Y = ATAN(X)
       RETURN
       END
```

$$
\begin{aligned}
y_{j,n+1} = y_{j,n} + \frac{h}{2}\,[f_j(x_n, y_{1,n}, y_{2,n}) \\
+ f_j(x_{n+1}, y_{1,n} + hf_1(x_n, y_{1,n}, y_{2,n}), y_{2,n} + hf_2(x_n, y_{1,n}, y_{2,n}))]
\end{aligned} \tag{9.140}
$$

for $j = 1, 2$. It is easier to consider this in the form (9.139); for programming it on a computer, the matrix form is very convenient. We leave the illustration of (9.140) to the problems.

A Computer Program for Systems

Following is a program of Euler's method for a system of N first order differential equations. The program is written for ease of use, and it can be easily changed to another method, such as (9.140). Note the use of the PARAMETER statement in setting the dimensions of the arrays used in the program. The dimension statement is changed easily by changing the parameter statement, and the code is more transparent to read.

```
C     TITLE: DEMONSTRATION OF EULER'S METHOD FOR SYSTEMS.
C
C     THIS SOLVES THE INITIAL VALUE PROBLEM FOR THE FIRST ORDER
C     SYSTEM
C        Y'(X) = F(X,Y(X)) ,  XO .LE. X .LE. B ,  Y(XO)=YO.
C     THE FUNCTION F(X,Z) IS DEFINED BELOW, ALONG WITH THE TRUE
C     SOLUTION Y(X).  THE NUMBER OF THE PROBLEM TO BE SOLVED
C     IS SPECIFIED BY THE INPUT VARIABLE 'NUMDE', WHICH IS USED
C     IN THE SUBROUTINES 'FCN' AND 'TRUE'.  THE PROGRAM WILL
C     REQUEST THE PROBLEM PARAMETERS, ALONG WITH THE VALUES OF
C     'H' AND 'IPRINT'.  'H' IS THE STEPSIZE, AND 'IPRINT' IS
C     THE NUMBER OF STEPS BETWEEN EACH PRINTING OF THE SOLUTION.
C     USE H=0 AND NUMDE=0 TO STOP THE PROGRAM.
C
C     TO INCREASE THE ORDER OF THE SYSTEMS WHICH CAN BE
C     HANDLED, INCREASE THE SIZE OF 'MAXN' IN THE PARAMETER
C     STATEMENT GIVEN BELOW.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (MAXN=3)
      DIMENSION YO(MAXN), Y1(MAXN), Y(MAXN), YZERO(MAXN), F(MAXN)
      PARAMETER (ZERO=0.0D0)
      COMMON/BLOCKF/NUMDE
C
C     INPUT PROBLEM PARAMETERS.
10    PRINT *, ' WHICH DIFFERENTIAL EQUATION?'
      PRINT *, ' GIVE ZERO TO STOP.'
      READ *, NUMDE
      IF(NUMDE .EQ. 0) STOP
      PRINT *, ' GIVE DOMAIN [X0,B] OF SOLUTION.'
      READ *, XZERO, XEND
      PRINT *, ' GIVE THE ORDER OF THE SYSTEM.  IT SHOULD BE'
      PRINT *, ' LESS THAN OR EQUAL TO MAXN=', MAXN
      READ *, N
      PRINT *, ' GIVE THE COMPONENTS OF Y0, ONE AT A TIME.'
      DO 15 J=1,N
        PRINT *, ' GIVE COMPONENT', J
15      READ *, YZERO(J)
C
20    PRINT *, ' GIVE STEPSIZE H AND PRINT PARAMETER IPRINT.'
      PRINT *, ' LET H=0 TO TRY ANOTHER DIFFERENTIAL EQUATION.'
      READ *, H,IPRINT
      IF(H .EQ. ZERO) GO TO 10
C
C     INITIALIZE.
      X0 = XZERO
      DO 25 J=1,N
25      Y0(J) = YZERO(J)
      PRINT 1000, NUMDE, XZERO, XEND, H, IPRINT
1000  FORMAT(//,' EQUATION',I2,5X,'XZERO =',1PE9.2,5X,'B =',E9.2,
     *       /,' STEPSIZE =',E10.3,5X,'PRINT PARAMETER =',I3)
      PRINT 1001, N
1001  FORMAT(' ORDER =',I1,//,' J',4X,'Y0(J)')
      PRINT 1002, (J,YZERO(J),J=1,N)
1002  FORMAT(I2,2X,1PE9.2)
      PRINT 1003
1003  FORMAT(/,5X,'X',7X,'J',6X,'APPROX Y(J)',7X,'ERROR')
C
C     BEGIN MAIN LOOP FOR COMPUTING SOLUTION OF DIFFERENTIAL EQUATION.
30    DO 40 K=1,IPRINT
        X1 = X0 + H
        IF(X1 .GT. XEND) GO TO 20
        CALL FCN(X0,Y0,F)
        DO 35 J=1,N
          Y1(J) = Y0(J) + H*F(J)
35        Y0(J) = Y1(J)
        X0 = X1
40      CONTINUE
C
C     CALCULATE ERROR AND PRINT RESULTS.
      CALL TRUE(X1,Y)
      DO 60 J=1,N
60      PRINT 1004, X1, J, Y1(J), Y(J) - Y1(J)
```

```
1004  FORMAT(F10.5,3X,I1,5X,1PE13.6,4X,E9.2)
      GO TO 30
      END

      SUBROUTINE FCN(X,Z,F)
C
C     THIS DEFINES THE RIGHT SIDE OF THE
C     DIFFERENTIAL EQUATION, F(X,Z).
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (TWO=2.0D0, THREE=3.0D0, FOUR=4.0D0, FIVE=5.0D0)
      DIMENSION Z(*), F(*)
      COMMON/BLOCKF/NUMDE
C
      GO TO (10,20), NUMDE
10    CS = COS(X)
      SN = SIN(X)
      F(1 )= Z(1) - TWO*Z(2) + FOUR*CS - TWO*SN
      F(2) = THREE*Z(1) - FOUR*Z(2) + FIVE*(CS - SN)
      RETURN
20    F(1) = Z(2)
      F(2) = Z(3)
      F(3) = -THREE*(Z(3) + Z(2)) - Z(1) - FOUR*SIN(X)
      RETURN
      END

      SUBROUTINE TRUE(X,Y)
C
C     THIS GIVES THE TRUE SOLUTION OF
C     THE INITIAL VALUE PROBLEM, Y(X).
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (TWO=2.0D0)
      DIMENSION Y(*)
      COMMON/BLOCKF/NUMDE
C
      GO TO(10,20), NUMDE
10    CS = COS(X)
      SN = SIN(X)
      Y(1) = SN + CS
      Y(2) = TWO*CS
      RETURN
20    SN = SIN(X)
      CS = COS(X)
      Y(1) = SN + CS
      Y(2) = CS - SN
      Y(3) = -SN - CS
      RETURN
      END
```

PROBLEMS

1. Let

$$A = \begin{bmatrix} 1 & -2 \\ 2 & -1 \end{bmatrix}, \qquad Y = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$$

$$G(x) = \begin{bmatrix} -2e^{-x} + 2 \\ -2e^{-x} + 1 \end{bmatrix}, \qquad Y_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Write out the two equations that make up the system

$$Y' = AY + G(x), \qquad Y(x_0) = Y_0$$

The true solution is $Y = [e^{-x}, 1]^T$.

2. Convert the system (9.137) to the general form of problem 1, giving the matrix A.

3. Convert the following higher order equations to systems of first order equations.

   (a) $y''' + 4y'' + 5y' + 2y = 2x^2 + 10x + 8$,
   $y(0) = 1, \qquad y'(0) = -1, \qquad y''(0) = 3$
   The true solution is $y(x) = e^{-x} + x^2$.

   (b) $y'' + 4y' + 13y = 40\cos(x), \qquad y(0) = 3, \qquad y'(0) = 4$
   The true solution is $y(x) = 3\cos(x) + \sin(x) + e^{-2x}\sin(3x)$.

4. Convert the following system of second order equations to a larger system of first order equations. It arises from studying the gravitational attraction of one mass by another.

$$x''(t) = \frac{-cx(t)}{r(t)^3}, \qquad y''(t) = \frac{-cy(t)}{r(t)^3}, \qquad z''(t) = \frac{-cz(t)}{r(t)^3}$$

with c positive constant and $r(t) = [x(t)^2 + y(t)^2 + z(t)^2]^{1/2}$, $t =$ time.

5. Using Euler's method, solve the system in problem 1. Use stepsizes of $h = 0.1, 0.05, 0.025$, and solve for $0 \le x \le 10$. Use Richardson's error formula to estimate the error for $h = 0.025$.

6. Repeat problem 5 for the systems in problem 3.

7. Modify the Euler program of this section to implement the Runge-Kutta method given in (9.139). With this program, repeat problems 5 and 6.

APPENDIX C

NUMERICAL ANALYSIS SOFTWARE PACKAGES

The programs included in this book are designed to give insight into the numerical methods being studied. They are not intended to be used in a production setting that requires highly accurate, efficient, robust, and convenient codes. For such computer codes, one should take advantage of the many high-quality numerical analysis program packages that have been written in the past 15 to 20 years. In this appendix we list some of these packages, along with information on obtaining them. Note that most of these packages are written in Fortran, and thus they are intended for use in a Fortran programming environment.

There are two widely used general libraries in numerical analysis, each containing programs for solving most standard numerical analysis problems. Both libraries are commercially available on most lines of computers and minicomputers, and both have subsets available on some microcomputers. In addition, both companies have additional numerical analysis packages that are not discussed in this appendix.

1. *IMSL (International Mathematics and Statistics Library).* This library contains over 900 subprograms for solving problems in numerical analysis and