Mathematics Notes
Note 10
January 1970

MLR

A Subroutine to Solve M Simultaneous Linear Equations in N Unknowns

Richard W. Sassman
Northrop Corporate Laboratories
Pasadena, California

ABSTRACT

This note describes a computer subroutine that solves a system of M linear equations in N unknowns. The matrix of the coefficients is reduced to upper triangular form by a series of unitary transformations thereby preserving the condition of the matrix in the reduction. After an initial solution is obtained an improved solution is obtained by iteration.

MLR

A Subroutine to Solve M Simultaneous Linear Equations in N Unknowns

INTRODUCTION

The present note describes a computer subroutine MLR that solves a system of M simultaneous linear equations in N unknowns ($M \geq N$). The subroutine is written in Fortran IV for the CDC 6600 computer. The matrix of coefficients of the M simultaneous equations is reduced to upper triangular form by a series of unitary transformations. Since the transformations are unitary, the condition of the matrix is preserved in the reduction. Once the matrix is triangularized the solution for any number of right-hand sides can be obtained by back-solving.

To increase the accuracy of the solution an iteration procedure is used which involves substituting a solution back into the original equations and using the difference between the result and the right-hand side as a new right-hand side to calculate a correction for the previous solution. By this means an accuracy of fifteen significant figures in the largest element of the solution is assured.

In the case of a square matrix, n square roots n divisions and $(n - 1)(2n^2 + 5n + 9)/3 \approx 2n^3/3$ multiplications are required in the reduction. Inversion of the reduced matrix requires $\binom{n+2}{3} + 2n^2$ multiplications and 2n divisions for each iteration.

ALGORITHM

Let

a_k = the k^{th} column of the matrix

$\alpha_k = \sqrt{a_k \cdot a_k}$

$-v_k = e_k \cdot a_k e_k / |e_k \cdot a_k|$

e_k = the k^{th} column of the identity matrix

$$u_k = \frac{a_k + \alpha_k v_k}{\sqrt{2\alpha_k(\alpha_k + v_k \cdot a_k)}}$$

then

$$-a_1 + \frac{(a_1 + \alpha_1 v_1) \cdot a_1 (a_1 + \alpha_1 v_1)}{\alpha_1 (\alpha_1 + v_1 \cdot a_1)} = \alpha_1 v_1 ,$$

that is, if one subtracts from each column k of the matrix

$$(a_1 + \alpha_1 v_1) \cdot a_k (a_1 + \alpha_1 v_1) / [\alpha_1 (\alpha_1 + v_1 \cdot a_1]$$

the result is that the first column of the matrix is zero except for the first element. One continues after suppressing the first column and first row of the transformed matrix and after n - 1 steps the matrix is triangularized. The transformation is unitary because it is formed from the unit vector u_k and the identity matrix I.

USE OF MLR

The subroutine MLR is used in conjunction with a main program which calls MLR using a call statement of the form

```
CALL MLR (M,N,ISW)
```

and a common statement of the form

```
COMMON /A/ A(M,N), B(M,N), X(N), Y(M), R(M), W(N), V(N)
```

where

M - an integer variable, is the number of rows in the matrix A. The numerical value of this variable must be assigned by the main program prior to calling MLR.

N - an integer variable, is the number of columns in the matrix A. The numerical value of this variable must be assigned by the main program prior to calling MLR. N must be less than or equal to M.

ISW - an integer variable, is the parameter of a COMPUTED GO TO statement in MLR and may be used in the same manner in the main program. Prior to the first call to MLR, ISW must be assigned the value 1 by the main program. Upon return from MLR to the main program, ISW will have been assigned the value

2 if a solution has been obtained and the value 3 if a singular matrix was detected. If ISW has the value 2 upon return from MLR and a subsequent call is made to MLR without changing ISW, the matrix reduction process will be skipped. Thus solutions can be obtained for additional right-hand sides without the necessity of reducing the matrix each time.

A - a floating point array, is the matrix to be inverted. The numerical values of this array must be assigned by the main program prior to calling MLR.

B - a floating point array. On return from MLR, B_{ij} contains the unitary transformation for $j \leq i$ and contains the reduced matrix less the diagonal for $j > i$.

X - a floating point array. On return from MLR, X contains the solution to the equations.

Y - a floating point array, is the right-hand side of the equations. The numerical values of this array must be assigned by the main program prior to calling MLR.

R - a floating point array. On return from MLR, this array contains the residual vector Y - AX.

W - a floating point array. On return from MLR, this array contains the diagonal of the reduced matrix.

V - a floating point array. On return from MLR, V_k contains $-\alpha_k(\alpha_k + v_k \cdot a_k)$.

REFERENCE

A. S. Householder, "Unitary Triangularization of a Non Symmetric Matrix," Journal of the Association for Computing Machinery, Oct. 1958.

APPENDIX - PROGRAM LISTING

```
      SUBROUTINE MLR(M,N,ISW)
      COMMON /A/ A(20,20),B(20,20),X(20),Y(20),R(20),W(20),V(20)
      DOUBLE RD
      INTEGER P
      GO TO (1,2), ISW
C     BEGIN TRIANGULARIZATION
C     MOVE MATRIX A TO B
    1 DO 10 I = 1, M
      DO 10 J = 1, N
   10 B(I,J) = A(I,J)
      DO 60 J = 1, N
      S = 0.
C     CALCULATE NORM SQUARED OF COLUMN J
      DO 20 K = J, M
   20 S = S + B(K,J)**2
      IF (S. GT. 0.) GO TO 21
C     IF THE NORM IS ZERO THE MATRIX IS SINGULAR
      ISW = 3
      PRINT 9
    9 FORMAT (23HOTHE MATRIX IS SINGULAR)
      RETURN
C     CALCULATE THE DIAGONAL ELEMENT OF MATRIX T. (W(J))
C     CALCULATE THE DIAGONAL ELEMENT OF MATRIX U. (B(J,J))
C     CALCULATE THE ELEMENT OF VECTOR V.
   21 W(J) = SIGN(SQRT(S),B(J,J))
      B(J,J) = B(J,J) + W(J)
      V(J) = - B(J,J)*W(J)
      IF (J. EQ. N) GO TO 60
      IB = J + 1
      DO 50 I = IB,N
      S = 0.
      DO 30 K = J, M
   30 S = S + B(K,J)*B(K,I)
```

```
            S = S/V(J)
            DO 40 K = J, M
         40 B(K,I) = B(K,I) + S*B(K,J)
         50 CONTINUE
         60 CONTINUE
C           BEGIN ITERATION
C           MOVE VECTOR Y TO R
          2 DO 70 I = 1, M
         70 R(I) = Y(I)
C           ZERO VECTOR X
            DO 80 I = 1, N
         80 X(I) = 0.
            TEST = 1.E+300
C           OPERATE ON R WITH THE MATRIX U
          3 DO 110 J = 1, N
            S = 0.
            DO 90 K = J, M
         90 S = S + R(K)*B(K,J)
            S = S/V(J)
            DO 100 K = J, M
        100 R(K) = R(K) + S*B(K,J)
        110 CONTINUE
C           BACKSOLVE MATRIX FOR INCREMENT IN X
            DO 130 J = 1, N
            K = N + 1 - J
            R(K) = R(K)/W(K)
            IF (J. EQ. N) GO TO 130
            IB = J + 1
            DO 120 I = IB, N
            P = N + 1 - I
        120 R(P) = R(P) + B(P,K)*R(K)
        130 CONTINUE
            DO 140 J = 1, N
        140 X(J) =  X(J) - R(J)
```

```
C           CALCULATE RESIDUAL VECTOR
            DO 160 I = 1, M
            RD = - Y(I)
            DO 150 J = 1, N
  150       RD = RD + A(I,J)*X(J)
  160       R(I) = - RD
C           CALCULATE NORM SQUARED OF RESIDUAL
            RSQ = 0.
            DO 170 I = 1, M
  170       RSQ = RSQ + R(I)**2
C           TEST FOR IMPROVEMENT
            IF (RSQ. GE. TEST) GO TO 4
            TEST = RSQ
            GO TO 3
C           SET SWITCH TO SKIP THE TRIANGULIZATION STEP
    4       ISW = 2
            RETURN
            END
```