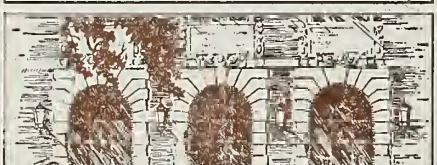

Center for Advanced Computation

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

CAC Document No. 134

COMPUTATIONAL TECHNIQUES FOR INPUT-OUTPUT ECONOMETRIC MODELS

by

Killion Noh and Ahmed Sameh

September 1974

CAC DOCUMENT NO. 134

COMPUTATIONAL TECHNIQUES

FOR

INPUT-OUTPUT ECONOMETRIC MODELS

By

Killion Noh

and

Ahmed Sameh

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

September, 1974

This work was supported by a grant from the U. S. Atomic Energy Commission with the cooperation of the National Science Foundation.

ABSTRACT

In an input-output econometric model we are often concerned with solving the system of n equations (I - A) x = y, repeatedly for various changes in the elements of A. This system of equations expresses gross output requirements (x) as a function of final demand (y) and the technological structure of the economy (A); changes in the elements of A can come about for a variety of reasons. In this paper we present techniques for solving such large systems of equations, and for updating the solution to account for changes in A. The methods presented effect substantial savings in computing time and storage requirements over those conventionally employed.

TABLE OF CONTENTS

1. INTRODUCTION

In an input-output econometric model we are often concerned with solving the system of n equations $(I - A)\,x = y$, repeatedly for various changes in the elements of A. This system of equations expresses gross output requirements (x) as a function of final demand (y) and the technological structure of the economy, and changes in the elements of A can come about for a variety of reasons. In this paper we present techniques for solving such large systems of equations, and for updating the solution to account for changes in A. The methods presented effect substantial savings in computing time and storage requirements over those conventionally employed.

Several methods [1] can be used in solving the system

$$(I - A)\,x = y \tag{1.1}$$

Gaussian elimination requires $\frac{1}{3}\,n^3$ multiplications, Householder triangularization requires $\frac{2}{3}\,n^3$ multiplications and n square roots, and Givens triangularization requires $\frac{4}{3}\,n^3$ multiplications and $\frac{1}{2}\,n^2$ square roots. In this paper we use a modification of Givens method developed by Gentleman [2] that requires only n^3 multiplications and no square roots. There are two reasons for such a choice. The first is that the matrices $B = I - A$ are usually dense and of large size, hence on some computers we may have to resort to secondary storage; and since these input-output matrices are often stored by rows Givens transformations are more suitable than those of Householder. Second, some of the procedures in updating the solution due to changes in B such as removing a row or a column are inherently unstable, so in order to minimize such effects it is desirable to use orthogonal transformations.

The system (1.1) is solved by Givens transformations as follows. We construct an orthogonal matrix Z as the product of Givens transformations such that [2],

$$ZB = D^{\frac{1}{2}}R \tag{1.2}$$

where D is a diagonal matrix, and R is unit upper triangular. From (1.1) and (1.2) we have

$$D^{\frac{1}{2}}Rx = Zy$$

i.e.,

$$Rx = \hat{y} \tag{1.3}$$

which is solved by backsubstitution.

Note that we do not store Z, the Givens transformations are multiplied by the final demand vector y as they are produced. When the matrix B is well-conditioned instead of solving (1.1) we may solve the normal equation

$$B^tBx = B^ty \tag{1.4}$$

Of course the condition number of the coefficient matrix is the square of the original one; however, this is of no serious consequence since B is very well-conditioned. Using (1.2) we may write (1.4) in the form,

$$R^tDRx = B^ty \tag{1.5}$$

Note that we do not store Z, the only storage required is that for B, R, D, x, and y, i.e. $2n^2 + 3n$ words. Finally x can be obtained by solving sequentially the systems,

$$R^tx_2 = B^ty, \quad Dx_1 = x_2, \quad Rx = x_1 \tag{1.6}$$

When a new right hand side $\tilde{y}$ is given with B unchanged, (1.6) is again used where we replace B^ty by $B^t\tilde{y}$.

Various methods have been developed for updating matrix factorization due to changes in the elements of the matrix. We refer mainly to the papers by Gill and Murray [3] and Golub, et. al. [4]. Throughout this paper we use Gentleman's square root free Givens transformation for the updating of the solution x in (1.1) due to row, column, and a single element changes in B.

2. SQUARE ROOT FREE GIVENS REDUCTION

We describe briefly the factorization (1.2). Let us consider a Givens transformation [2] that rotates two row vectors such that

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \sqrt{\alpha} & \sqrt{\alpha}\, u_2 & \ldots\ldots & \sqrt{\alpha}\, u_n \\ \sqrt{\beta}\, v_1 & \sqrt{\beta}\, v_2 & \ldots\ldots & \sqrt{\beta}\, v_n \end{bmatrix} = \begin{bmatrix} \sqrt{\tilde{\alpha}} & \sqrt{\tilde{\alpha}}\, \tilde{u}_2 & \ldots\ldots & \sqrt{\tilde{\alpha}}\, \tilde{u}_n \\ 0 & \sqrt{\tilde{\beta}}\, \tilde{v}_2 & \ldots\ldots & \sqrt{\tilde{\beta}}\, \tilde{v}_n \end{bmatrix} \tag{2.1}$$

where

$$\begin{aligned} \tilde{\alpha} &= \alpha + \beta v_1^2 \\ \tilde{\beta} &= \alpha\beta/\tilde{\alpha} \\ c &= \alpha/\tilde{\alpha} \\ s &= \beta v_1/\tilde{\alpha} \end{aligned}$$

$$\left.\begin{aligned} \tilde{u}_i &= c u_i + s v_i \\ \tilde{v}_i &= v_i - v_1\, u_i \end{aligned}\right\} \qquad i = 2, \ldots, n \tag{2.2}$$

Note that no square root evaluations are involved in these formulae and (2.1) may be written as

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \alpha & \\ & \beta \end{bmatrix}^{\frac{1}{2}} \begin{bmatrix} 1 & u_2 & \ldots & u_n \\ v_1 & v_2 & \ldots & v_n \end{bmatrix} = \begin{bmatrix} \tilde{\alpha} & \\ & \tilde{\beta} \end{bmatrix}^{\frac{1}{2}} \begin{bmatrix} 1 & \tilde{u}_2 & \ldots & \tilde{u}_n \\ 0 & \tilde{v}_2 & \ldots & \tilde{v}_n \end{bmatrix} \tag{2.3}$$

We define an n-dimensional Givens transformation Z_k^i by,

$$
Z_k^i = \begin{bmatrix}
1 & & & & & & & \\
& \ddots & & & & & & \\
& & 1 & & & & & \\
& & & c & \cdots & s & & \\
& & & \vdots & 1 \ddots 1 & \vdots & & \\
& & & -s & \cdots & c & & \\
& & & & & & 1 & \\
& & & & & & & \ddots \; 1
\end{bmatrix}
\begin{matrix} \\ \\ \\ i\text{-th row} \\ \\ k\text{-th row} \\ \\ \\ \end{matrix}
\tag{2.4}
$$

in which $c \equiv \cos \alpha_{ki}$, $s \equiv \sin \alpha_{ki}$, and the angle α_{ki} is chosen such that the element in the position (k, i) of the matrix Z_k^i B is eliminated. Let

$$Z = Z_{i+1}^i \;\cdots\; Z_{n-1}^i \; Z_n^i$$

and,

$$Z = Z_{n-1} \;\cdots\; Z_2 \; Z_1 \tag{2.5}$$

thus,

$$ZB = D^{\frac{1}{2}}R$$

where D is diagonal, and R is a unit upper triangular matrix.

As is mentioned in Section 1, $2n^2 + 3n$ storage locations and n^3 multiplications without square root operations are necessary for Givens reduction. In this paper and the attached programs we are assuming that the matrices B and R, and the diagonal vector D are stored in the high speed memory of the computer and are available for updating the solutions due to changes in B, which will be discussed in the following sections.

3. COLUMN MODIFICATION

Given the system (1.1) and the factorization (1.2) we wish to determine the new factorization

$$\tilde{B}^t\tilde{B} = \tilde{R}^t\tilde{D}\tilde{R} \tag{3.1}$$

in less than n^3 operations, where $\tilde{B}$ is different from B in only one column ([3], [4]).

We first consider deleting a column b_i from B. Let

$$B_1 = [b_1, \ldots, b_{i-1}, \ b_{i+1}, \ \ldots, \ b_n]$$

be an n x (n-1) matrix obtained from B by deleting the i-th column, thus

$$B_1^t \ B_1 = R_o^t \ D \ R_o$$

where R_o is R with the i-th column deleted. If we partition R_o as

$$R_o = \begin{bmatrix} H_1 & \vdots & H_2 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & H_3 \end{bmatrix}$$

then the (i-1) x (i-1) submatrix H_1 is unit upper triangular, and H_3 is an upper Hessenberg matrix of order (n-i). We wish to find an orthogonal transformation Z such that

$$\begin{aligned} B_1^t \ B_1 &= R_o^t \ D^{\frac{1}{2}} Z^t Z D^{\frac{1}{2}} R_o \\ &= R_1^t D_1 R_1 \end{aligned} \qquad (3.3)$$

where

$$D_1^{\frac{1}{2}} R_1 = Z D^{\frac{1}{2}} R_o$$

in which R_1 is an (n-1) x (n-1) unit upper triangular matrix. This can be done by defining Z as a product of the following Givens transformations:

$$Z = Z_n^{n-1} \ . \ . \ . \ . \ Z_{i+2}^{i+1} \ \ Z_{i+1}^{i}$$

That is, Z reduces the upper Hessenberg matrix H_3 to a unit upper triangular matrix. Thus, deleting a column from B and obtaining a modified factorization involve only the temporary storage of the (n-i) x (n-i) lower principal submatrix of R_o.

We now consider adding a column b to B_1. Let

$$\tilde{B} = \begin{bmatrix} B_1 & \vdots & b \end{bmatrix}$$
$$\tilde{R} = \begin{bmatrix} R_1 & \vdots & \\ \cdots\cdots & \vdots & r \\ 0 & \vdots & \end{bmatrix} \tag{3.4}$$

and

$$\tilde{D} = \begin{bmatrix} D_1 & \vdots & 0 \\ \cdots\cdots & \vdots & \cdots\cdots \\ 0 & \vdots & d \end{bmatrix}$$

in which an n-vector r with its n-th component unity and a scalar d are to be determined such that

$$\tilde{B}^t\tilde{B} = \tilde{R}^t\tilde{D}\tilde{R}$$

From (3.4),

$$\tilde{B}^t\tilde{B} = \begin{bmatrix} B_1^t \\ \cdots\cdots \\ b^t \end{bmatrix} \begin{bmatrix} B_1 & \vdots & b \end{bmatrix}$$
$$= \begin{bmatrix} B_1{}^tB_1 & \vdots & B_1{}^tb \\ \cdots\cdots & \vdots & \cdots\cdots \\ b^tB_1 & \vdots & b^tb \end{bmatrix} \tag{3.5}$$

and

$$\tilde{R}^t\tilde{D}\ \tilde{R} = \begin{bmatrix} R_1{}^t & \vdots & 0 \\ \cdots\cdots & \vdots & \cdots\cdots \\ & r^t & \end{bmatrix} \begin{bmatrix} D_1 & \vdots & 0 \\ \cdots\cdots & \vdots & \cdots\cdots \\ 0 & \vdots & d \end{bmatrix} \begin{bmatrix} R_1 & \vdots & \\ \cdots\cdots & \vdots & r \\ 0 & \vdots & \end{bmatrix}$$
$$= \begin{bmatrix} R_1{}^tD_1R_1 & \vdots & [R_1{}^tD_1 \ \vdots \ 0]\, r \\ \cdots\cdots & \vdots & \cdots\cdots \\ r^t\tilde{D}\begin{bmatrix} R_1 \\ \cdots \\ 0 \end{bmatrix} & \vdots & r^t\tilde{D}r \end{bmatrix} \tag{3.6}$$

Comparing right hand sides of (3.5) and (3.6), we obtain

$$\begin{bmatrix} R_1^t D_1 & \vdots & 0 \\ & r^t D r & \end{bmatrix} r = B_1^t b \qquad (3.7)$$
$$= b^t b$$

from which r and d are computed.

Thus, the modified factorization (3.1) is completely determined and the modified solution $\tilde{x}$ is obtained from

$$\tilde{R}^t \tilde{D} \tilde{R} \tilde{x} = \tilde{B}^t y \qquad (3.8)$$

by forward and backward substitution as before.

The maximum number of operations necessary for a column modification is only $5n^2$ multiplications. Note that only 3n extra storages for x, y, and a new column in addition to those for B, D, and R are required for deleting and adding a column, and updating D and R.

4. ROW MODIFICATION

Let us first consider deleting a row b_i^t from B,

$$B_1 = B - \begin{bmatrix} 0 \\ \cdots\cdots \\ b_i^t \\ \cdots\cdots \\ 0 \end{bmatrix}$$

then

$$B_1^t B_1 = B^t B - b_i b_i^t$$
$$= R^t D R - b_i b_i^t$$

where R and D are the original factors of B and we assume that they are available. We wish to determine a unit upper triangular matrix R_1 and a diagonal matrix D_1 such that

$$R_1^t D_1 R_1 = R^t DR - b_i b_i^t \qquad (4.1)$$

To do this ([3], [4]), solve

$$R^t D v = b_i \qquad (4.2)$$

for v and let

$$s^2 = 1 - \|v\|_2^2$$

Now we construct an (n+1) x (n+1) matrix

$$R_o = \begin{bmatrix} s & 0 \\ D^{\frac{1}{2}}v & D^{\frac{1}{2}}R \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & D^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} s & 0 \\ v & R \end{bmatrix}$$

and reduce it to a unit upper triangular matrix,

$$Z \begin{bmatrix} 1 & 0 \\ 0 & D^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} s & 0 \\ v & R \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & D_1^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} 1 & r^t \\ 0 & R_1 \end{bmatrix} \qquad (4.3)$$

where

$$Z = Z_2^1 \ldots\ldots Z_n^1 \; Z_{n+1}^1$$

and, R_1 and D_1 are the desired factorization of B_1. In order to show that, we have from (4.3)

$$\begin{bmatrix} s^2 + v^tDv & v^tDR \\ R^tDv & R^tDR \end{bmatrix} = \begin{bmatrix} 1 & r^t \\ r & rr^t + R_1{}^tD_1R_1 \end{bmatrix}$$

comparing both sides, we obtain

$$v^tDv + s^2 = 1$$

$$R^tDv = r$$

$$R^tDR = rr^t + R_1{}^tD_1R_1$$

Therefore

$$r = b_i$$

and, the relation (4.1) is satisfied.

For adding a row g_i^t to B_1, let

$$\tilde{B} = B_1 + \begin{bmatrix} 0 \\ g_i^t \\ 0 \end{bmatrix}$$

Then

$$\tilde{B}^t\tilde{B} = B_1{}^tB_1 + g_ig_i{}^t \tag{4.4}$$

Now construct an (n + 1) x n matrix

$$\begin{bmatrix} D_1{}^{\frac{1}{2}}R_1 \\ \cdots\cdots \\ g_i^t \end{bmatrix}$$

and reduce it to a unit upper triangular matrix,

$$Z \begin{bmatrix} D_1{}^{\frac{1}{2}}R_1 \\ \cdots\cdots \\ g_i^t \end{bmatrix} = \begin{bmatrix} \tilde{D}^{\frac{1}{2}}\tilde{R} \\ \cdots\cdots \\ 0 \end{bmatrix} \tag{4.5}$$

where

$$Z = Z^n_{n+1} \quad . \; . \; . \quad Z^2_{n+1} \quad Z^1_{n+1}$$

From (4.5) we obtain

$$R_1{}^tD_1R_1 + g_ig_i{}^t = \tilde{R}^t\tilde{D}\tilde{R} \tag{4.6}$$

Therefore, from (4.4) and (4.6)

$$\tilde{B}^t\tilde{B} = \tilde{R}^t\tilde{D}\tilde{R}$$

Thus the modified factorization is completely determined and the modified solution is obtained from

$$\tilde{R}^t\tilde{D}\tilde{R}\tilde{x} = \tilde{B}^ty$$

as before.

The number of operations required for the row modification is $5\frac{1}{2}n^2$ multiplications. Again, only 4n extra storage locations are necessary in addition to those for B, D, and R.

For an alternative method for row modification see Sameh and Bezdek [5].

5. ELEMENT MODIFICATION

When only one element of the matrix B is changed and we wish to update the solution, the Sherman-Morrison formula [6] may be utilized since only n^2 multiplications are required in the process.

Let us consider the problem in which we wish to solve

$$(B + \alpha\, e_i\, e_j^t)\, \tilde{x} = y \tag{5.1}$$

where α is a scalar, and e_i and e_j are the i-th and the j-th column vectors of an identity matrix, respectively.

From (5.1) and the Sherman-Morrison formula, we obtain

$$\begin{aligned}\tilde{x} &= (B + \alpha e_i\, e_j^t)^{-1} y \\ &= (B^{-1} - \beta B^{-1}\, e_i\, e_j^t\, B^{-1})\, y \\ &= B^{-1} y - \beta\, B^{-1}\, e_i\, e_j^t\, B^{-1} y \end{aligned} \tag{5.2}$$

where

$$\beta = (\alpha^{-1} + e_j^t\, B^{-1}\, e_i)^{-1} \tag{5.3}$$

Assuming the original solution is known, $x = B^{-1} y$, (5.2) becomes

$$\tilde{x} = x - \beta\, B^{-1}\, e_i\, e_j^t\, x \tag{5.4}$$

If we solve for w in

$$B\, w = e_i \tag{5.5}$$

using the factorization (1.5), then

$$\beta = (\alpha^{-1} + w_j)^{-1}$$

and

$$\tilde{x} = x - \beta\, x_j\, w \tag{5.6}$$

Although n^2 multiplications are necessary for the element modifications, if the i-th column of B^{-1} is available, the number of multiplications is reduced to n.

Only 2n extra storage locations are necessary in addition to those for B, D, and R.

Until now, we have assumed that the computer core memory is large enough so that the matrices B, R, and D can simultaneously be contained in it. However, since we are dealing with matrices of large size, it may happen that the core memory cannot accommodate all or part of them simultaneously. In this case, the matrices can be stored by rows in the secondary storage, and several rows of B or R are brought into the core whenever necessary.

For applying Givens transformation for triangularization of B or updating R, only as many rows of B or R that the core memory can accommodate are brought in since only two rows of B or R are necessary for a single Givens transformation.

In order to compute $B^t y$, which appears in (3.7), (3.8), and Section 4, each row vector of the matrix, B_i in the core is multiplied by y_i, i-th component of y, i.e.

$$B^t y = \sum_{i=1}^{n} B_i y_i$$

REFERENCES

[1] J. H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford, 1965.

[2] W. M. Gentleman, "Least Squares Computations by Givens Transformations Without Square Roots", J. Inst. Maths. Applics., Vol. 12 (1973), pp. 329-336.

[3] P. E. Gill and W. Murray, "A Numerically Stable Form of the Simplex Algorithm", National Physical Laboratory, DNAM Report No. 87, Teddington, England, 1972.

[4] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, "Methods for Modifying Matrix Factorizations", Math. Comp., Vol. 28, (1974), pp. 505-535.

[5] A. E. Sameh and R. H. Bezdek, "Methods for Increasing the Computational Efficiency of Input-Output and Related Large Scale Matrix Operations", CAC Document No. 66, Center for Advanced Computation, University of Illinois, 1973.

[6] A. S. Householder, The Theory of Matrices in Numerical Analysis, New York, 1964.

```
      SUBROUTINE GIVGEN(NN,N,A,B,D,R,X,EPS)
C
C     THIS SUBROUTINE SOLVES A SYSTEM OF LINEAR EQUATIONS A*X=B BY
C     GIVENS TRANSFORMATIONS WITHOUT SQUARE ROOT EVALUATIONS.
C     IT DETERMINES THE FACTORIZATION Z*A=D**(1/2)*R, WHERE Z IS AN
C     ORTHOGONAL MATRIX, D IS A DIAGONAL MATRIX, AND R IS A UNIT UPPER
C     TRIANGULAR MATRIX. Z IS NOT STORED. SOLUTION IS THEN OBTAINED
C     BY A BACKWARD SUBSTITUTION. IF ANY ELEMENT OF A IS LESS THAN EPS,
C     IT IS REPLACED BY ZERO.
C
C     INPUT
C        NN   DECLARED DIMENSION OF MATRIX A
C        N    ORDER OF MATRIX A
C        A    MATRIX OF ORDER N
C        B    RIGHT HAND SIDE VECTOR
C
C     OUTPUT
C        X    SOLUTION VECTOR
C        D    DIAGONAL VECTOR
C        R    UNIT UPPER TRIANGULAR MATRIX OF ORDER N
C        EPS  MACEPS*(NORM OF A)
C
      REAL*8 A(NN,N),B(N),D(N),R(NN,N),X(N)
      REAL*8 ALPHA,BETA,UI,VI,CBAR,SBAR,TEMP,ANORM,MACHEP,DMAX1,
     X       DABS,EPS
      DATA MACHEP/Z341000000000000/
      DO 5 J=1,N
      D(J)=1.0D0
      X(J)=B(J)
      DO 5 I=1,N
    5 R(I,J)=A(I,J)
C
C     COMPUTE INFINITE NORM OF A AND EPS=MACHEPS*ANORM
      ANORM=0.0D0
      DO 15 I=1,N
      TEMP=0.0D0
      DO 10 J=1,N
   10 TEMP=TEMP+DABS(R(I,J))
   15 ANORM=DMAX1(ANORM,TEMP)
      EPS=MACHEP*ANORM
C
      NM1=N-1
      DO 100 J=1,NM1
      JP1=J+1
      NJ=N+JP1
      I=J
      UI=R(I,J)
C
C     SPECIAL CASES FOR GIVENS TRANSFORMATIONS
      IF (DABS(UI).LT.EPS) GO TO 25
      DO 20 M=JP1,N
   20 R(I,M)=R(I,M)/UI
      X(I)=X(I)/UI
      R(I,J)=1.0D0
      D(I)=UI*UI*D(I)
      GO TO 30
   25 R(I,J)=0.0D0
   30 CONTINUE
      DO 100 L=JP1,N
      K=NJ-L
```

```
      VI=R(K,J)
      IF (DABS(VI).GE.EPS) GO TO 35
      R(K,J)=0.0D0
      GO TO 90
   35 CONTINUE
      IF (DABS(UI).GE.EPS) GO TO 50
      DO 40 M=J,N
      TEMP=-R(I,M)
      R(I,M)=R(K,M)
   40 R(K,M)=TEMP
      TEMP=-X(I)
      X(I)=X(K)
      X(K)=TEMP
      TEMP=D(I)
      D(I)=D(K)
      D(K)=TEMP
      DO 45 M=JP1,N
   45 R(I,M)=R(I,M)/VI
      X(I)=X(I)/VI
      D(I)=VI*VI*D(I)
      R(I,J)=1.0D0
      GO TO 90
   50 CONTINUE
C
C     GIVENS TRANSFORMATIONS
      ALPHA=D(I)
      BETA=D(K)
      D(I)=ALPHA+BETA*VI*VI
      D(K)=ALPHA*BETA/D(I)
      R(I,J)=1.0D0
      R(K,J)=0.0D0
      CBAR=ALPHA/D(I)
      SBAR=BETA*VI/D(I)
      DO 55 M=JP1,N
      TEMP=R(K,M)-VI*R(I,M)
      R(I,M)=CBAR*R(I,M)+SBAR*R(K,M)
   55 R(K,M)=TEMP
      TEMP=X(K)-VI*X(I)
      X(I)=CBAR*X(I)+SBAR*X(K)
      X(K)=TEMP
   90 CONTINUE
  100 CONTINUE
      TEMP=R(N,N)
      X(N)=X(N)/TEMP
      D(N)=TEMP*TEMP*D(N)
      R(N,N)=1.0D0
C
C     BACKWARD SUBSTITUTION
      DO 130 I=1,N
      J=N-I+1
      JP1=J+1
      TEMP=0.0D0
      IF (JP1.GT.N) GO TO 120
      DO 110 M=JP1,N
  110 TEMP=TEMP+R(J,M)*X(M)
  120 X(J)=X(J)-TEMP
  130 CONTINUE
      RETURN
      END
      SUBROUTINE COLMOD(NN,N,A,B,COL,ICOL,D,R,X,EPS)
```

```
C
C     THIS SUBROUTINE UPDATES THE SOLUTION X, DIAGONAL MATRIX D, AND
C     UNIT UPPER TRIANGULAR MATRIX R WHEN A COLUMN OF A IS CHANGED,
C     PROVIDED THAT THE FACTORS R AND D, OBTAINED FROM 'GIVGEN',
C     ARE KNOWN.
C
C     INPUT
C        NN   DECLARED DIMENSION OF MATRIX A
C        N    ORDER OF MATRIX A
C        A    MATRIX OF ORDER N
C        B    RIGHT HAND SIDE VECTOR
C        COL  NEW COLUMN VECTOR
C        ICOL THE ICOL-TH COLUMN OF A IS CHANGED
C        D    DIAGONAL VECTOR OBTAINED FROM 'GIVGEN'
C        R    UNIT UPPER TRIANGULAR MATRIX OBTAINED FROM 'GIVGEN'
C        EPS  MACHEPS*(NORM OF A) OBTAINED FROM 'GIVGEN'
C
C     OUTPUT
C        X    UPDATED SOLUTION VECTOR
C        D    UPDATED DIAGONAL VECTOR
C        R    UPDATED UNIT UPPER TRIANGULAR MATRIX
C
      REAL*8 A(NN,N),B(N),COL(N),D(N),R(NN,N),X(N)
      REAL*8 ALPHA,BETA,UI,VI,CBAR,SBAR,TEMP,EPS,DABS
C
C     REMOVE ICOL-TH COLUMN OF R TO OBTAIN R0
      NM1=N-1
      IF (ICOL.EQ.N) GO TO 210
      DO 20 J=ICOL,NM1
      JP1=J+1
      DO 20 I=1,JP1
   20 R(I,J)=R(I,JP1)
C
C     REDUCTION OF R0 TO UPPER TRIANGULAR FORM R1
      DO 200 I=ICOL,NM1
      K=I+1
      UI=R(I,I)
      VI=R(K,I)
      IF (DABS(UI).LT.EPS) GO TO 120
      IF (I.EQ.NM1) GO TO 115
      DO 110 J=K,NM1
  110 R(I,J)=R(I,J)/UI
  115 R(I,I)=1.0D0
      D(I)=UI*UI*D(I)
      GO TO 125
  120 R(I,I)=0.0D0
  125 CONTINUE
      IF (DABS(VI).GE.EPS) GO TO 130
      R(K,I)=0.0D0
      GO TO 190
  130 CONTINUE
      IF (DABS(UI).GE.EPS) GO TO 150
      DO 135 J=I,NM1
      TEMP=-R(I,J)
      R(I,J)=R(K,J)
  135 R(K,J)=TEMP
      TEMP=D(I)
      D(I)=D(K)
      D(K)=TEMP
      IF (I.EQ.NM1) GO TO 145
```

```
      DO 140 J=K,NM1
  140 R(I,J)=R(I,J)/VI
  145 CONTINUE
      D(I)=VI*VI*D(I)
      R(I,I)=1.0D0
      GO TO 190
  150 CONTINUE
      ALPHA=D(I)
      BETA=D(K)
      D(I)=ALPHA+BETA*VI*VI
      D(K)=ALPHA*BETA/D(I)
      R(I,I)=1.0D0
      R(K,I)=0.0D0
      CBAR=ALPHA/D(I)
      SBAR=BETA*VI/D(I)
      IF (I.EQ.NM1) GO TO 190
      DO 155 J=K,NM1
      TEMP=R(K,J)-VI*R(I,J)
      R(I,J)=CBAR*R(I,J)+SBAR*R(K,J)
  155 R(K,J)=TEMP
  190 CONTINUE
  200 CONTINUE
C
C     (TRANSPOSE OF A1)*COL=X
  210 CONTINUE
      ICOLM1=ICOL-1
      IF (ICOL.EQ.1) GO TO 235
      DO 230 J=1,ICOLM1
      TEMP=0.0D0
      DO 220 I=1,N
  220 TEMP=TEMP+A(I,J)*COL(I)
  230 X(J)=TEMP
      IF (ICOL.EQ.N) GO TO 260
  235 CONTINUE
      DO 250 J=ICOL,NM1
      JP1=J+1
      TEMP=0.0D0
      DO 240 I=1,N
  240 TEMP=TEMP+A(I,JP1)*COL(I)
  250 X(J)=TEMP
  260 CONTINUE
C
C     COMPUTE THE LAST COLUMN OF RTILDE AND THE LAST COMPONENT OF DTILDE
      DO 320 J=1,NM1
      JM1=J-1
      TEMP=0.0D0
      IF (J.EQ.1) GO TO 320
      DO 310 I=1,JM1
  310 TEMP=TEMP+R(I,J)*R(I,N)
  320 R(J,N)=X(J)-TEMP
      DO 330 I=1,NM1
  330 R(I,N)=R(I,N)/D(I)
      TEMP=0.0D0
      DO 340 I=1,N
  340 TEMP=TEMP+COL(I)*COL(I)
      BETA=0.0D0
      DO 350 I=1,NM1
  350 BETA=BETA+R(I,N)*R(I,N)*D(I)
      D(N)=TEMP-BETA
      R(N,N)=1.0D0
```

```
C
C     COMPUTE RHS=(ATILDE)T*B
      IF (ICOL.EQ.1) GO TO 385
      DO 380 J=1,ICOLM1
      TEMP=0.0D0
      DO 370 I=1,N
  370 TEMP=TEMP+A(I,J)*B(I)
  380 X(J)=TEMP
      IF (ICOL.EQ.N) GO TO 405
  385 CONTINUE
      DO 400 J=ICOL,NM1
      JP1=J+1
      TEMP=0.0D0
      DO 390 I=1,N
  390 TEMP=TEMP+A(I,JP1)*B(I)
  400 X(J)=TEMP
  405 TEMP=0.0D0
      DO 410 I=1,N
  410 TEMP=TEMP+COL(I)*B(I)
      X(N)=TEMP
C
C     FORWARD AND BACKWARD SUBSTITUTIONS
      DO 430 J=1,N
      JM1=J-1
      TEMP=0.0D0
      IF (J.EQ.1) GO TO 430
      DO 420 I=1,JM1
  420 TEMP=TEMP+R(I,J)*X(I)
  430 X(J)=X(J)-TEMP
      DO 440 I=1,N
  440 X(I)=X(I)/D(I)
      DO 460 I=1,N
      J=N-I+1
      JP1=J+1
      TEMP=0.0D0
      IF (JP1.GT.N) GO TO 460
      DO 450 K=JP1,N
  450 TEMP=TEMP+R(J,K)*X(K)
  460 X(J)=X(J)-TEMP
C
C     REORDERING OF X(1),...,X(N)
      IF (ICOL.EQ.N) GO TO 480
      TEMP=X(N)
      DO 470 I=ICOL,NM1
      J=N-I+ICOL
  470 X(J)=X(J-1)
      X(ICOL)=TEMP
  480 CONTINUE
      RETURN
      END
      SUBROUTINE ROWMOD(NN,N,A,B,ROW,IROW,D,R,X,EPS,V)
C
C     THIS SUBROUTINE UPDATES THE SOLUTION X, DIAGONAL MATRIX D, AND
C     UNIT UPPER TRIANGULAR MATRIX R WHEN A ROW OF MATRIX A IS CHANGED,
C     PROVIDED THAT THE FACTORS R AND D, OBTAINED FROM 'GIVGEN',
C     ARE KNOWN.  V IS A TEMPORARY STORAGE VECTOR.
C
C     INPUT
C        NN   DECLARED DIMENSION OF MATRIX A
C        N    ORDER OF MATRIX A
```

```
C         A    MATRIX OF ORDER N
C         B    RIGHT HAND SIDE VECTOR
C         ROW  NEW ROW VECTOR
C         IROW THE IROW-TH ROW OF A IS CHANGED
C         D    DIAGONAL VECTOR OBTAINED FROM 'GIVGEN'
C         R    UNIT UPPER TRIANGULAR MATRIX OBTAINED FROM 'GIVGEN'
C         EPS  MACHEPS*(NORM OF A) OBTAINED FROM 'GIVGEN'
C
C      OUTPUT
C         X    UPDATED SOLUTION VECTOR
C         D    UPDATED DIAGONAL VECTOR
C         R    UPDATED UNIT UPPER TRIANGULAR MATRIX
C
      REAL*8 A(NN,N),B(N),ROW(N),D(N),R(NN,N),X(N),V(N)
      REAL*8 ALPHA,BETA,UI,VI,CBAR,SBAR,TEMP,DI,DK,S,EPS,DABS
C
C      SOLVE (RT)*(D)*V=BI FOR V
C
      DO 10 J=1,N
   10 X(J)=A(IROW,J)
      DO 30 J=1,N
      JM1=J-1
      TEMP=0.0D0
      IF (J.EQ.1) GO TO 30
      DO 20 I=1,JM1
   20 TEMP=TEMP+R(I,J)*V(I)
   30 V(J)=X(J)-TEMP
      DO 40 I=1,N
   40 V(I)=V(I)/D(I)
C
C      REDUCTION OF R0 TO UNIT UPPER TRANGULAR MATRIX R1
      DO 45 I=1,N
   45 X(I)=0.0D0
      DI=1.0D0
      S=0.0D0
      DO 200 I=1,N
      K=N-I+1
      UI=S
      VI=V(K)
      IF (DABS(VI).GE.EPS) GO TO 135
      V(K)=0.0D0
      GO TO 190
  135 CONTINUE
      IF (DABS(UI).GE.EPS) GO TO 150
      DO 140 J=K,N
      TEMP=-X(J)
      X(J)=R(K,J)
  140 R(K,J)=TEMP
      TEMP=DI
      DI=D(K)
      D(K)=TEMP
      DO 145 J=K,N
  145 X(J)=X(J)/VI
      DI=VI*VI*DI
      S=1.0D0
      V(K)=0.0D0
      GO TO 190
  150 CONTINUE
      ALPHA=DI
      BETA=D(K)
```

```
      DI=ALPHA+BETA*VI*VI
      D(K)=ALPHA*BETA/DI
      S=1.0D0
      V(K)=0.0D0
      CBAR=ALPHA/DI
      SBAR=BETA*VI/DI
      DO 160 J=K,N
      TEMP=R(K,J)-VI*X(J)
      X(J)=CBAR*X(J)+SBAR*R(K,J)
  160 R(K,J)=TEMP
  190 CONTINUE
  200 CONTINUE
C
C     COMPUTE RHS=(ATILDE)T*B
      DO 220 J=1,N
      TEMP=0.0D0
      DO 210 I=1,N
      S=A(I,J)
      IF (I.EQ.IROW) S=ROW(J)
  210 TEMP=TEMP+S*B(I)
  220 X(J)=TEMP
C
C     REDUCTION OF R2 TO UNIT UPPER TRIANGULAR MATRIX RTILDE
      DK=1.0D0
      DO 400 I=1,N
      IP1=I+1
      UI=R(I,I)
      IF (DABS(UI).LT.EPS) GO TO 325
      IF (I.EQ.N) GO TO 322
      DO 320 J=IP1,N
  320 R(I,J)=R(I,J)/UI
  322 CONTINUE
      R(I,I)=1.0D0
      D(I)=UI*UI*D(I)
      GO TO 330
  325 R(I,I)=0.0D0
  330 CONTINUE
      VI=ROW(I)
      IF (DABS(VI).GE.EPS) GO TO 335
      ROW(I)=0.0D0
      GO TO 390
  335 CONTINUE
      IF (DABS(UI).GE.EPS) GO TO 350
      IF (I.EQ.N) GO TO 342
      DO 340 J=IP1,N
      TEMP=-R(I,J)
      R(I,J)=ROW(J)
  340 ROW(J)=TEMP
  342 CONTINUE
      TEMP=D(I)
      D(I)=DK
      DK=TEMP
      IF (I.EQ.N) GO TO 347
      DO 345 J=IP1,N
  345 R(I,J)=R(I,J)/VI
  347 CONTINUE
      D(I)=VI*VI*D(I)
      R(I,I)=1.0D0
      GO TO 390
  350 CONTINUE
```

```
      ALPHA=D(I)
      BETA=DK
      D(I)=ALPHA+BETA*VI*VI
      DK=ALPHA*BETA/D(I)
      CBAR=ALPHA/D(I)
      SBAR=BETA*VI/D(I)
      R(I,I)=1.0D0
      ROW(I)=0.0D0
      IF (I.EQ.N) GO TO 390
      DO 360 J=IP1,N
      TEMP=ROW(J)-VI*R(I,J)
      R(I,J)=CBAR*R(I,J)+SBAR*ROW(J)
  360 ROW(J)=TEMP
  390 CONTINUE
  400 CONTINUE
C
C     FORWARD AND BACKWARD SUBSTITUTION
      DO 430 J=1,N
      JM1=J-1
      TEMP=0.0D0
      IF (JM1.LT.1) GO TO 430
      DO 420 I=1,JM1
  420 TEMP=TEMP+R(I,J)*V(I)
  430 V(J)=X(J)-TEMP
      DO 440 I=1,N
  440 V(I)=V(I)/D(I)
      DO 460 I=1,N
      J=N-I+1
      JP1=J+1
      TEMP=0.0D0
      IF (JP1.GT.N) GO TO 460
      DO 450 K=JP1,N
  450 TEMP=TEMP+R(J,K)*X(K)
  460 X(J)=V(J)-TEMP
      RETURN
      END
      SUBROUTINE EMTMOD(NN,N,A,X,D,R,EMT,ITH,JTH,V)
C
C     THIS SUBROUTINE UPDATES THE SOLUTION X OF A*X=B WHEN AN ELEMENT
C     OF A IS CHANGED, PROVIDED THAT THE FACTORS R, D AND THE SOLUTION
C     X, OBTAINED FROM 'GIVGEN', ARE KNOWN.  V IS A TEMPORARY STORAGE
C     VECTOR.
C
C     INPUT
C        NN    DECLARED DIMENSION OF MATRIX A
C        N     ORDER OF MATRIX A
C        A     MATRIX OF ORDER N
C        X     SOLUTION VECTOR OBTAINED FROM 'GIVGEN'
C        D     DIAGONAL VECTOR OBTAINED FROM 'GIVGEN'
C        R     UNIT UPPER TRIANGULAR MATRIX OBTAINED FROM 'GIVGEN'
C        EMT   NEW ELEMENT
C        ITH,JTH   THE (ITH,JTH)-ELEMENT OF A IS CHANGED
C
C     OUTPUT
C        X     UPDATED SOLUTION VECTOR
C
      REAL*8 A(NN,N),X(N),D(N),R(NN,N),V(N),SIGMA,TAU,EMT,TEMP
C
C     SOLVE RT*D*R*V=AT*E(I)
      DO 30 J=1,N
```

```
      JM1=J-1
      TEMP=0.0D0
      IF (J.EQ.1) GO TO 30
      DO 20 I=1,JM1
   20 TEMP=TEMP+R(I,J)*V(I)
   30 V(J)=A(ITH,J)-TEMP
      DO 40 I=1,N
   40 V(I)=V(I)/D(I)
      DO 60 I=1,N
      J=N-I+1
      JP1=J+1
      TEMP=0.0D0
      IF (J.EQ.N) GO TO 60
      DO 50 K=JP1,N
   50 TEMP=TEMP+R(J,K)*V(K)
   60 V(J)=V(J)-TEMP
C
C     COMPUTE SIGMA AND TAU
      SIGMA=EMT-A(ITH,JTH)
      TAU=1.0D0+SIGMA*V(JTH)
      TAU=SIGMA/TAU
C
C     COMPUTE THE UPDATED SOLUTION
      TEMP=TAU*X(JTH)
      DO 70 I=1,N
   70 X(I)=X(I)-TEMP*V(I)
      RETURN
      END
```

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUC-CAC-DN-74-134	2.	3. Recipient's Accession No.
4. Title and Subtitle COMPUTATIONAL TECHNIQUES FOR INPUT-OUTPUT ECONOMETRIC MODELS			5. Report Date September, 1974
			6.
7. Author(s) Killion Noh and Ahmed Sameh			8. Performing Organization Rept. No. CAC 134
9. Performing Organization Name and Address Center for Advanced Computation University of Illinois at Urbana-Champaign Urbana, Illinois 61801			10. Project/Task/Work Unit No.
			11. Contract/Grant No. NSF GI-35179X
12. Sponsoring Organization Name and Address National Science Foundation 1800 G Street Washington, D. C. 20301			13. Type of Report & Period Covered Research
			14.

15. Supplementary Notes

16. Abstracts

In an input-output econometric model we are often concerned with solving the system of n equations (I - A) x = y, repeatedly for various changes in the elements of A. This system of equations expresses gross output requirements (x) as a function of final demand (y) and the technological structure of the economy (A); changes in the elements of A can come about for a variety of reasons. In this paper we present techniques for solving such large systems of equations, and for updating the solution to account for changes in A. The methods presented effect substantial savings in computing time and storage requirements over those conventionally employed.

17. Key Words and Document Analysis. 17a. Descriptors

Computational Techniques
Econometric Model
Input/Output
Output requirements
Column modification.
Row modification
Element modification

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement No restriction on distribution. Available from National Technical Information Service, Springfield, Virginia 22151	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 21
	20. Security Class (This Page UNCLASSIFIED	22. Price