C

Center for Advanced Computation

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

CAC Document No. 235

AN EFFICIENT ALGORITHM FOR SOLVING THE LINEAR INPUT OUTPUT EQUATION WITH AN EXTENSION TO THE NONLINEAR INPUT OUTPUT MODEL

by

Doug Gilmore

August, 1977

CAC Document No. 235

An Efficient Algorithm for Solving the Linear Input Output Equation with an Extension to the Nonlinear Input Output Model

by

Doug Gilmore

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

August, 1977

This work was supported in part by the National Science Foundation under Contract US NSF C-1045, and was submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 1977.

ACKNOWLEDGEMENTS

I would like to express my appreciation first to Professor Hugh Folk for his approval of the work done on the subject of this thesis. At the beginning many thought that performing such a large computation on a small computer installation was absurd, and I appreciate that Professor Folk gave me the opportunity to demonstrate that the contrary was quite true. I would also like to thank Professor Achmed Sameh for his assistance during the development of the algorithms, and for his comments during the drafting of the thesis. I also thank my thesis advisor, Professor William Perkins for his comments on the drafts of the thesis. His comments have greatly improved the readability of the mathematical text. I also give special thanks to Steve Holmgren. He was unceasingly assistive during the coding of the algorithms. Though I value the assistance of all the above, I alone accept responsibility for any errors.

TABLE OF CONTENTS

INTRODUCTION

Due to recent economic developments such as the energy crisis and persistent cost push inflation there has been considerable interest of the structure of national economies, and how crisis effects economic activities. One method that is used to study economic systems is Input Output (IO) analysis which is particularly powerful in the study of interindustry activity. The purpose of this paper is to discuss the solution of the IO equation using several methods, the goal being to develop a method to solve a nonlinear Input Output model which will partly alleviate some of the restrictions of linear Input Output analysis.

Input Output analysis is an econometric method which attempts to explain all industrial activity by a simple cause effect relationship. The first critical assumption of IO analysis is that all goods in a product group are manufactured in an identical manner. From this point on when the term good is mentioned, it is to be taken as some representative good in one of the product groups. The amount of a good that society needs to produce is the amount to be supplied for final demand[1] plus the

----

[1] By final demand we mean several things: goods consumed by households and government, goods sold for export, and also goods used for investment. Most logically investment would be treated as an input to production, but investment can be a very nonlinear function of output. Thus in general economists find it easier to determine investment demands exogeneously. It is possible that the representation of investment as a nonlinear function of output could make the endogenous determination of investment demands more realistic.

amount used in the production of other goods. The second critical assumption of IO analysis is the following: when a good is used in the production of another good, the amount used in this activity is always in a fixed proportion to the total production of the other good. These proportionality constants are termed the technical coefficients, and since in general it is possible that all goods can be used directly in the production of all other goods, if we have an n good economy then there are n^2 technical coefficients. When the technical coefficients are arranged in a nxn matrix, this matrix is called the technical coefficients matrix or A matrix, though this should not be confused with the A matrix found in control systems literature. If our unit of value is dollars then the element a_{ij} represent the dollar value of the i^{th} good required in the direct production of one dollar of the j^{th} good. Thus the a_{ij}'s are positive fractions. The sum of the j^{th} column of the A matrix represents the fraction of the total cost of producing the good j that is embodied in the goods used in direct production of the j^{th} good. Then

$$v_j = 1 - \sum_{i=1}^{n} a_{ij} \tag{0.1}$$

represents the unit cost of production not embodied in the goods used in the direct production of the j^{th} good. V_j is termed the value added for good j and it consists of labor costs, interest, (on the capital used in the direct production of the j^{th} good,) direct business taxes, and profits.

If y_i is the amount of the i^{th} good sold to final demand, then the equation which represents the total production of the i^{th} good, that is, x_i, is

$$x_i = y_i + \sum_{j=1}^{n} a_{ij}x_j \ . \qquad (0.2)$$

The summation term in (0.2) is the amount of good i needed in the direct production of all goods. Proceeding in the same way for the other n-1 goods, the total demand for all goods is the solution of the matrix equation

$$Ax + y = x \quad \text{or} \quad (I - A)x = y \ . \qquad (0.3)$$

We reiterate the basic assumptions of IO analysis; 1) that all goods in the same product class are assumed to be made in the same way, 2) that the amount of an input good used in the direct production of another good is always in a fixed proportion.

The first question that needs to be answered is whether solutions to (0.3) do indeed exist. But since we also desire that solutions must correspond to actual economic behavior, the solution vectors should be positive[2] if the final demand vector is positive. Bellman[1, pp.296, Theorem 6] has shown that if the column sums of A are less that 1, then there is a unique positive

----

[2] A vector is said to be positive if all of its elements are positive.

solution vector for each positive right hand side. Furthermore, the eigenvalues of A lie inside the unit circle, and

$$(I - A)^{-1} = I + \sum_{i=1}^{\infty} A^{i} , \qquad (\emptyset.4)$$

with $\lim_{m\to\infty} A^{m} = \emptyset$, e.g. see Isaacson and Keller[15, pp.15, Theorem 5]. If k terms are used to approximate $(I - A)^{-1}$ then $(k-1)n^3$ multiplications[3] must be performed.

In the past, eg. Chenary and Clark[17], $(I-A)^{-1}$ was crudely approximated by this method. The objective of this paper is to discuss a method by which the IO equation can be solved with far more efficiency and accuracy. By a factorization method which will be discussed in Chapter 1, only $\frac{1}{3}n^3$ multiplications are required to completely factorize the matrix into a product of triangular matrices, which can then be solved for arbitrary right hand sides in n^2 multiplications. The first section of Chapter 1 will also analyze the numerical stability properties of factorization of Input Output matrices by elementary transformations, (sometimes denoted as LU decomposition,) and will reexamine the property of the A matrix so that (Ø.3) will admit only positive solutions for arbitrary positive final demand vectors. The second section in this chapter proposes a simple method which

----

[3] It is standard practice in numerical analysis to only count the number of multiplications or divisions in a computation since they are more time consuming than addition or subtraction operations.

will allow modifications of particular rows and columns of the A matrix that will not require the complete refactorization of the matrix. This method has a very useful property that allows the successive updating of the IO matrix without any algorithmic complexity. Procedures of this type are generally referred to as factorization modification, and several papers have been written on this subject, including Gill and Murray[13], Golub et al.[12], Sameh and Bezdek[2], and Noh and Sameh[3]. But with the exception of Gill and Murray[13], these only deal with factorization by orthogonal transformations, which will be more numerically stable for general matrices, but require more storage and computation than factorization by elementary transformations thus making them comparably more expensive. By examining the structure of IO matrices, we shall see that the use of elementary transformations in the factorization of IO matrices is quite appropriate. We do not mean to distract the reader away from orthogonal transformation factorization methods. As Bierman[5] points out, (this article is an excellent survey of numerical techniques in control and other applications,) in general these methods lend to many algorithmic advantages, and the numerical stability which results is important in control problems which can be ill-conditioned.

The next section discusses the updating of solutions of linear equations when only a few elements of several columns or rows are changed. We will denote this method by "solution perturbation" since the method depends on solutions to a nominal system of equations. This is a bit of a misnomer, since pertur-

bation implies small changes. However here the changes in elements of the matrix need not be small in any way. In many cases this method has a tremendous computational advantage over factorization modification but it does have its drawbacks. This method requires that columns of the nominal inverse, (that is of $(I - A)^{-1}$ before any elements have changed,) to be computed. In the case of modifying an entire column solution perturbation would require that the entire nominal inverse be computed. Since the computation of an inverse requires approximately three times the computation required to factorize a matrix, this method is not appropriate if many elements in a column are to be modified. Also the method has a disadvantage in that several successive updates of the solution due to new perturbations in the matrix elements are difficult to perform since there is no efficient way to compute the updated inverses.

In Chapter 3 a nonlinear IO model is proposed which can be solved quite efficiently with the eclectic use the algorithms described in the preceding chapters. This approach largely eliminates the linearity contraint of the standard IO model while the extra cost of solving the nonlinear model is quite small. Using solution perturbation the nonlinear equation that must be solved iteratively is reduced to smaller order. Once the residuals are sufficiently small factorization modification will be utilized to find the complete solution. These algorithms have been coded on a minicomputer system, and the computation of solutions of 350^{th}-order IO equations is quite feasible on such a computer system when these algorithms are used.

CHAPTER 1
FACTORIZATION MODIFICATIONS

SECTION 1.1
MATRIX DECOMPOSITION BY ELEMENTARY TRANSFORMATIONS, AND SIMPLIFICATIONS THEREOF ON IO MATRICES.

This section describes the LU decomposition algorithm. The presentation will also include a discussion of properties of IO matrices that will simplify the algorithms to be described in the next section.

As with all decomposition algorithms, in the decomposition of the matrix B, (in our case B= I- A, where I is the identity matrix, and A is the technical coefficient matrix defined in the introductory section, thus $b_{ij} < 0$ and $b_{ii} = 1 - a_{ii} > 0$,) a transformation Q is determined such that

$$QB = U \tag{1.1.1}$$

where U is a upper triangular matrix. To solve the system of equations

$$Bx = y \tag{1.1.2}$$

we premulitply both sides of (1.1.2) by Q, so that

$$QBx = Ux = Qy \tag{1.1.3}$$

and the equation Ux = Qy can be solved by back substitution. In the pure form of LU decomposition Q is the composition of only elementary transformations. An elementary transformation has a

simple matrix representation. Its diagonal elements are all one and it has only one nonzero off diagonal element which we shall call the multiplier element. If M_{ij} is an elementary transformation then it looks like:

$$\begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \cdots & & & & \\ & & & 1 & & & \\ & & & & \cdots & & \\ & & & m_{ij} & & & \\ & & & & & & 1 \end{bmatrix} \tag{1.1.4}$$

To simplify the notation, the subscripts below a symbol representing an elementary transformation will denote the position of the multiplier element in the elementary transformation matrix. Thus m_{ij} is in the position (i,j) of the matrix. It is easily verified the the inverse of an elementary transformation is merely the elementary transformation with the nonzero off diagonal element of opposite sign. The direct multiplication of elementary matrices,

$$N_j = M_{n,j}M_{n-1,j}\cdot\ldots\cdot M_{j+2,j}M_{j+1,j} \tag{1.1.5}$$

looks like:

$$\begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & m_{j+1,j} & 1 & & & \\ & & m_{j+2,j} & & \ddots & & \\ & & \vdots & & & \ddots & \\ & & m_{n,j} & & & & 1 \end{bmatrix} \qquad (1.1.6)$$

The LU decomposition algorithm determines the transformation

$$Q = N_{n-1}N_{n-2}\cdot\ldots\cdot N_2N_1 \ . \qquad (1.1.7)$$

If for each j we define

$$N_j(\bar{B}_j) = N_j(N_{j-1}N_{j-2}\cdots N_1B) \ , \qquad (1.1.8)$$

then the algorithm determines N_j such that it zeroes the elements of the j^{th} column of $\bar{B}_j$ below the diagonal element. Thus

$$m_{ij} = -\frac{\bar{b}_{ij}}{\bar{b}_{jj}} \qquad \text{for } j+1 \leq i \leq n \qquad (1.1.9)$$

where the m_{ij}'s are elements of (1.1.6), $\bar{B}_j = (\bar{b}_{ij})$, and $\bar{b}_{jj}$ is called a pivot element.

A reader familiar with decomposition by elementary transformations[4] probably notes that row permutation has been omitted. The reason for this will be clear shortly.

Proposition: If all the column sums of A are less than unity then the pivot elements are nonzero.

Proof: Clearly if the column sums of A are less than unity then the same must hold for the i^{th} leading principal minor of A. Thus by Bellman[1, pp.296, theorem 6] the principal minors of (I-A) are nonsingular. Stewart[10, pp.120, theorem 2.5] has shown that if the principal minors of a matrix are nonzero then if the matrix is decomposed by elementary transformations without pivoting, the pivot elements will be nonzero.

There is a fundamental result in Input Output analysis which describes the necessary and sufficient condition on A so that the IO equation admits only positive solutions vectors for positive final demand vectors. This is called the Simon-Hawkins condition[9]. The condition is that all the principal minors of (I - A) must be positive. Let us examine the computation of the solution of IO equation by factorization so that we can see when

----

[4] See Forsythe and Moler[16], or Isaacson and Keller[15] on introductory material on decomposition methods.

it is impossible for a solution to be a nonpositive vector when the final demand vector is positive. After applying the transformation Q to the final demand vector, since all the multiplier elements of Q are nonnegative, each element of Qy must be greater than or equal to the corresponding element of y. In the solution of Ux=Qy, since the off diagonal elements are all nonpositive, (thus in the back substitution all sums are on numbers of nonnegative sign,) the only way that a negative element can occur in the solution is if a diagonal element of U is negative. Thus if all the diagonal elements of U are positive, (which implies that all of the multiplier elements of the elementary transformations must be positive,) then it is impossible for the factorized equation to admit a nonpositive solution vector for a positive final demand vector. This is exactly how the Simon-Hawkins condition is checked: the determinant of the k^{th} principal minor of (I - A) is merely the product of the k topmost elements of the diagonal of U since $\det|Q|=1$.

In the definition of the transformation Q one order of applying the elementary transformations was given. Since elementary transformations are nearly identity matrices one would expect that these transformations will commute in certain cases, (clearly any permutation of the elementary transformations in N_j will yield the same transformation.) The elementary transformations can be commuted just as long as no element becomes nonzero that was intentionally zeroed by the application of a previous elementary transformation. Thus:

$$Q = M_{nn-1} \cdots M_{n2} \cdots M_{32} M_{n1} \cdots M_{21} \qquad (1.1.10)$$

$$= M_{nn-1} \cdots M_{43} M_{42} M_{41} M_{32} M_{31} M_{21} \qquad (1.1.11)$$

If we apply Q by (1.1.10) then after the first n-1 elementary transformations are applied, the partially decomposed matrix has the structure:

$$\begin{bmatrix} x & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x \end{bmatrix} . \qquad (1.1.12)$$

After the application of the next n-2 elementary transformations the matrix has the structure:

$$\begin{bmatrix} x & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x \\ 0 & 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x & x \end{bmatrix} . \qquad (1.1.13)$$

If we apply Q by (1.1.11) then after the application of the first

elementary transformation the partially decomposed matrix has the structure:

$$\begin{bmatrix} x & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x \\ x & x & x & x & x & x & x \\ x & x & x & x & x & x & x \\ x & x & x & x & x & x & x \\ x & x & x & x & x & x & x \\ x & x & x & x & x & x & x \end{bmatrix} . \qquad (1.1.14)$$

Then after the application of the next two elementary transformations the matrix has the structure:

$$\begin{bmatrix} x & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x \\ 0 & 0 & x & x & x & x & x \\ x & x & x & x & x & x & x \\ x & x & x & x & x & x & x \\ x & x & x & x & x & x & x \\ x & x & x & x & x & x & x \end{bmatrix} . \qquad (1.1.15)$$

Another ordering which will reduce page faults in virtual computers or IO requests in successively reading in and writing out parts of the matrix on machines with little main memory involves storing the matrix in blocks of rows. The topmost partially decomposed block is completely decomposed. Then using this completely decomposed block the remaining blocks are partially

decomposed using this block, that is, all the transformations that need this completely decomposed block will be applied to the lower blocks. Thus this block will no longer be needed. This will reduce page faults or IO requests by a factor equal to the number of rows that are stored in each block. All these methods are equivalent not only mathematically but numerically, that is, if one is careful about the ordering of the computations, all of the methods will achieve identical matrices in terms of the bit patterns.

The name of the method, LU decomposition, refers to the definition

$$LU = B \tag{1.1.16}$$

where L is lower triangular. By inspection

$$L = Q^{-1} \; . \tag{1.1.17}$$

This is easily shown by the direct multiplication of the inverses of the elementary transformations.

The algorithm performs approximately $n^3/3$ multiplications and requires only the original matrix as work space. The algorithm is the fastest and most compact of all decomposition algorithms, Moler[11].

SECTION 1.2

MULTIPLE ROW AND COLUMN MODIFICATIONS

In this section an algorithm will be discussed which will allow the modification of the factorization of a matrix when certain preselected rows and columns of the matrix are changed. A significant advantage of this algorithm is that each modification of the factorization adds no complexity to solving the new system of equations, since the modified factorization is solved in the same manner as the original system. Also the modified factorization is identical to the factorization resulting from completely factorizing the modified matrix. As we shall see the computation of the modified factorization and the computation of solutions to the modified system of equations can be performed simultaneously. This is useful when the computations are performed on a minicomputer. Experience has shown that the cost of reading in the matrices is the most significant cost in the solution of a large factorized system of equations. Thus streamlining the algorithms with respect to the number of times that the matrices must be read in from secondary storage is worthwhile. In Aoki[4] and Householder[5], LU decomposition is introduced in a manner that would lend itself to factorization modification methods, though there was no intention to discuss the subject.

The algorithm for factorization modification greatly simplifies if the rows to be modified are at the bottom of the matrix, and the columns are on the right hand border. Thus before performing the factorization we exchange the rows to be modified with the bottom rows of the matrix, and exchange the columns to

be modified with the columns on the right hand border. Let P, $P^{-1}=P^t$, be the transformation that exchanges the rows and columns. Clearly the matrix representation of P has n^2-n zero elements, the other n elements being unity. Rows of the transformation corresponding to coordinates that are not permuted have 1 on the diagonal. If coordinate i is to be exchanged with the j^{th} coordinate, then the elements in positions (i,j) and (j,i) would be one. Therefore for all i,j $p_{ij}=p_{ji}$ or $P=P^t$. Also it is easily verified that PP=I.

For example let there be two industries for which we desire column or row modifications, their positions being say 1 and 3. We wish to construct a transformation with the properties above which permutes these industries so that their rows are at the bottom and the columns are on the right hand border. P would then look like:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \qquad (1.2.1)$$

The above example verifies that P is symmetric. The permuted form of B will be denoted by

$$B = P^tBP = PBP . \qquad (1.2.2)$$

Let Q_p be the composition of M_{ij}'s for $1 \le j \le n-k$, $j+1 \le i \le n$. Q_p will partially decompose the matrix B. The elementary transformations are to applied as not to destroy zeros previously introduced. The partial decomposition will be denoted by

$$Q_p B = U \ . \tag{1.2.3}$$

In our 6x6 example above, pre- and post-multiplying the original matrix A by P will permute the rows and columns 1 and 3 to the bottom and right border of the matrix. Note that $P(I - A)P = (I - PAP)$. After Q_p is applied to our permuted (I-A) matrix, the resultant matrix U has the structure:

$$\begin{bmatrix} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x & x \end{bmatrix} \tag{1.2.4}$$

At this point any of the last k columns of B can be modified. Let us assume that we want to replace a column of B which corresponds to one of the last k columns of B by a column vector g. U is updated merely by replacing the corresponding column of U by $Q_p Pg$. Changing more such columns requires the identical procedure for each column. The transformation of g and the right

hand side vector by $Q_p P$ can be performed simultaneously. This feature in very useful when the computation is performed on a minicomputer system.

Also any of the last k rows of $\bar{B}$ can be modified. If a row of B which corresponds to one of the last k rows of $\bar{B}$ is to be replaced by the row vector h^t, then the corresponding elementary transformations in Q_p which zeroed out the first n-k elements of corresponding row of $\bar{U}$ are recomputed. Suppose that in our 6x6 example we desire to replace the first row of B by h^t. This would correspond to replacing the 5^{th} row of $\bar{B}$ by $(Ph)^t$. Thus we recompute the elementary transformations M_{51} through M_{54}. Before these elementary transformations are applied, the modified $\bar{U}$ has the structure:

$$\begin{bmatrix} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ x & x & x & x & x & x \\ 0 & 0 & 0 & 0 & x & x \end{bmatrix} \qquad (1.2.5)$$

Note that this procedure does not really violate the rule that no element that has been intentionally zeroed be set nonzero by a elementary transformation applied later. Since we are recomputing all the elementary transformations that zeroed out the elements in the row that is to be modified, it is as though these elementary transformations were never applied before the modification

was made.

For any number of row and/or column modifications, the modified factorization will be identical to the factorized matrix which would result from partially decomposing $\bar{B}$, where $\bar{B}$ is the matrix with the row and/or column modifications. In fact when the original rows and columns are replaced in the factorization, the new file is exactly bit comparable to the original. To achieve this the reader is warned that care must be taken in the ordering of operations in the original decomposition and the rows and columns modification algorithms.

To solve the original or a modified system of equations, the rest of the elementary transformations are applied that will reduce $\hat{U}$ to a triangular matrix. This second composition of elementary transformations, (the M_{ij}s' where $n-k+1 \leq i \leq n-1$ and $i+1 \leq j \leq n$,) are applied in an order that will not make any element nonzero which was intentionally set to zero by a previously applied elementary transformation. Denoting this transformation by Q_c and letting $\bar{U}$ be the completely decomposed matrix, the system is solved as follows.

$$Bx = y \tag{1.2.6}$$

$$PBPPx = \hat{B}Px = y \tag{1.2.7}$$

$$Q_cQ_p\hat{B}Px = Q_c\hat{U}Py = \bar{U}Px = Q_cQ_pPx \tag{1.2.8}$$

$$Px = \bar{U}^{-1}Q_cQ_pPy \tag{1.2.9}$$

$$x = P\bar{U}^{-1} Q_c Q_p P y \tag{1.2.10}$$

Here we have used the fact that PP = I and that $P^t = P$. The transformation U^{-1} is performed by backsubstitution on U. Note that if row and column modifications as above have been performed, the algorithm to solve the system does not change. The relevant multiplier elements in the elementary transformations of Q_p are changed only if row modifications were performed, while for column modifications only the relevant columns of $\bar{U}$ have different elements. If the decomposition is terminated so that the lower right hand kxk submatrix remains unfactorized, the modification of i rows and j columns requires no more than $\frac{1}{2}(i+j)n^2$ multiplications, while the solution of the new system requires $n^2 + \frac{1}{3}k^3$. If the factorization of the kxk submatrix is stored, the subsequent solutions require n^2 multiplications.

To give the reader a sense on how large k can be such that the completion of the factorization for each new solution becomes comparatively expensive let us set the number of multiplications required to solve the system equal to the number required to complete the factorization, that is $n^2 = \frac{1}{3}k^3$ or $k=(3n)^{\frac{3}{2}}$ For n=100, k=45, and for n=400, k=113, and for n=1000, k=208.

CHAPTER 2
A SENSITIVITY TRANSFORMATION FOR STUDYING THE PERTURBATIONS OF SOLUTIONS DUE TO THE PERTURBATION OF THE SYSTEM

SECTION 2.1
PERTURBATION OF A SINGLE ELEMENT

The first example to be investigated is the change of the solution due to the change of one element in the system of equations. This result was first deduced by Shermam and Morrison[8]. Though the result here is the same, the procedure and representation are different. At the end of this section we will point out how this is so.

Let $\tilde{B}$ be a nxn matrix that is identical to the matrix B except for the element (i,j) which is represented as:

$$\tilde{b}_{ij} = b_{ij} + \delta b_{ij} \; . \tag{2.1.1}$$

Let B be decomposed by a decomposition technique. Then the n columns of the inverse of B can be found by solving the decomposed system for the appropriate unit vector as a right hand side. Suppose that x is the solution of

$$Bx = y \tag{2.1.2}$$

and $\ddot{x}$ is the solution of

$$\tilde{B}\ddot{x} = (B+\delta B)\ddot{x} = y \; . \tag{2.1.3}$$

Premultiplying the above equation by B^{-1} we have

$$(I+B^{-1}\delta B)\ddot{x} = B^{-1}y = x \tag{2.1.4}$$

where δB is a square matrix of order n in which only one element is nonzero, that is, δb_{ij}. Written out explicitly $(I + B^{-1}\delta B)$, is of the form:

$$\begin{bmatrix} 1 & & & \bar{b}_{1i}\delta b_{ij} & & & \\ & \cdots & & \cdots & & & \\ & & 1 & \bar{b}_{j-1,i}\delta b_{ij} & & & \\ & & & 1+\bar{b}_{ji}\delta b_{ij} & & & \\ & & & b_{j+1,i}\delta b_{ij} & 1 & & \\ & & & \cdots & & \cdots & \\ & & & \bar{b}_{ni}\delta b_{ij} & & & 1 \end{bmatrix} \tag{2.1.5}$$

where $(\bar{b}_{kl}) = B^{-1}$. It follows directly that

$$\ddot{x}_j = \frac{x_j}{1+ \bar{b}_{ji}\delta b_{ij}} , \tag{2.1.6}$$

and for $k \neq j$

$$\ddot{x}_k = x_k - \bar{b}_{ki}\delta b_{ij}x_j = x_k - \frac{\bar{b}_{ki}\delta b_{ij}x_j}{1 + \bar{b}_{ji}\delta b_{ij}} . \tag{2.1.7}$$

This result can also be found by representing the perturbed system by

$$(B + \delta B)\ddot{x} = (I + \delta BB^{-1})B\ddot{x} = y . \tag{2.1.8}$$

If we denote $\bar{B}\ddot{x}=y$ as the perturbed system, then solving

$$(I + \delta BB^{-1})z=y \tag{2.1.9}$$

and then

$$B\ddot{x} = z \tag{2.1.10}$$

yields the solution to the perturbed system. The transforming matrix $(I + \delta BB^{-1})$ is of the form:

$$\begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ \delta b_{ij}\bar{b}_{j1} & \cdots & \delta b_{ij}\bar{b}_{ji-1} & 1+\delta b_{ij}\bar{b}_{ji} & \delta b_{ij}\bar{b}_{ji+1} & \cdots & \delta b_{ij}\bar{b}_{jn} \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \qquad (2.1.11)$$

Solving $(I + \delta BB^{-1})z = y$, for $k \neq i$

$$z_k \quad y_k , \tag{2.1.12}$$

while for the i^{th} element,

$$\begin{aligned} y_i &= \delta b_{ij}\bar{b}_{j1}y_1 + \delta b_{ij}\bar{b}_{j2}y_2 + \ldots + \delta b_{ij}\bar{b}_{ji-1}y_{i-1} + \\ &+ (1 + \delta b_{ij}\bar{b}_{ji})z_i + \delta b_{ij}\bar{b}_{ji+1}y_{i+1} + \ldots \\ &\ldots + \delta b_{ij}\bar{b}_{jn}y_n , \end{aligned} \tag{2.1.13}$$

or

$$\begin{aligned} y_i &= \delta b_{ij}\bar{b}_{j1}y_1 + \delta b_{ij}\bar{b}_{j2}y_2 + \ldots + \delta b_{ij}\bar{b}_{ji-1}y_{i-1} + \\ &+ \delta b_{ij}\bar{b}_{ji}y_i - \delta b_{ij}\bar{b}_{ji}y_i \\ &+ (1 + \delta b_{ij}\bar{b}_{ji})z_i + \delta b_{ij}\bar{b}_{ji+1}y_{i+1} + \ldots \\ &\ldots + \delta b_{ij}\bar{b}_{jn}y_n . \end{aligned} \tag{2.1.14}$$

Since

$$\sum_{k=1}^{n} \delta b_{ij}\bar{b}_{jk}y_k = \delta b_{ij}x_j , \tag{2.1.15}$$

(2.1.14) reduces to

$$y_i = \delta b_{ij} x_j + (1 + \delta b_{ij} b_{ji}) z_i - \delta b_{ij} \bar{b}_{ji} y_i \ . \qquad (2.1.16)$$

or

$$z_i = \frac{(1 + \delta b_{ij} \bar{b}_{ji}) y_i - \delta b_{ij} x_j}{1 + \delta b_{ij} \bar{b}_{ji}} \qquad (2.1.17)$$

$$= y_i - \frac{\delta b_{ij} x_j}{1 + \delta b_{ij} \bar{b}_{ji}} \ . \qquad (2.1.18)$$

Therefore z can be written as

$$z = y - \left(\frac{\delta b_{ij} x_j}{1 + \delta b_{ij} \bar{b}_{ji}}\right) e_i \qquad (2.1.19)$$

where e_i is the unit vector of the i^{th} component. Since $\ddot{x} = B^{-1} z$,

$$\ddot{x} = x - \frac{\delta b_{ij} x_j}{1 + \delta b_{ij} \bar{b}_{ji}} B^{-1} e_i \ . \qquad (2.1.20)$$

Explicitly for $k \neq j$

$$\ddot{x}_k = x_k - \frac{\bar{b}_{ki} \delta b_{ij} x_j}{1 + \delta b_{ij} \bar{b}_{ji}} \qquad (2.1.21)$$

and for the j^{th} element,

$$\ddot{x}_j = x_j - \frac{\bar{b}_{ji}\delta b_{ji} x_j}{1 + \delta b_{ij}\bar{b}_{ji}} = \frac{x_j}{1 + \delta b_{ij}\bar{b}_{ji}}\,, \qquad (2.1.22)$$

which corresponds to (2.1.6) and (2.1.7).

The next two sections will consider multiple row and column perturbations. We will see that the analysis of such perturbations will be quite straight forward. For column perturbations the perturbed system of equations will be represented as in (2.1.4), while for row peturbations the representation (2.1.8) is appropriate. In Sherman and Morrison[8] it is only shown that (2.1.6), and (2.1.7) are correct, though they do not show how they arrive at the results. The purpose of this presentation is to demonstrate a procedure which can be used to derive the solution for an arbitrary perturbation, rather than propose a solution and demonstrate its validity.

SECTION 2.2

COLUMN PERTURBATIONS

In this section the method just described is extended to columnwise perturbations. The first case is for perturbations in a single column with the scaling of the column, while the second is the extension to two columns.

Without loss of generality let us suppose that the first k elements of the j^{th} column are to be perturbed and that a constant multiple of the column is to be added to the entire column, that is, δB is an nxn matrix which has all zeroes for its elements except for the j^{th} column, where the j^{th} column is:

$$\left| \delta b_{1j}+cb_{ij} \quad \delta b_{2j}+cb_{2j} \quad \cdots \quad \delta b_{kj}+cb_{kj} \quad cb_{k+1j} \quad \cdots \quad cb_{nj} \right|^t . \tag{2.2.1}$$

Again the matrix $(I + B^{-1}\delta B)$ has the form of (2.1.5) except the j^{th} column is replaced by:

$$\left| \gamma_1 \quad \gamma_2 \quad \cdots \quad \gamma_{j-1} \quad (1+\gamma_j+c) \quad \gamma_{j+1} \cdots \quad \gamma_n \right|^t \tag{2.2.2}$$

where

$$\gamma_m = \sum_{p=1}^{k} b_{mp} \delta \bar{b}_{pj} \; , \tag{2.2.3}$$

and $(\bar{b}_{ij}) = B^{-1}$. The solution is found oy inspection as in

Chapter 2.1:

$$\ddot{x}_j = \frac{x_j}{1 + \alpha_j + c}, \tag{2.2.4}$$

and for $i \neq j$,

$$\ddot{x}_i = x_i - \alpha_i \ddot{x}_j \tag{2.2.5}$$

where $\ddot{x}$ is the solution of $(B + \delta B)\ddot{x}=y$, and x is the solution of the unperturbed system $Bx=y$.

Now suppose that two columns are to be perturbed, say columns j_1 and j_2. For the simplicity of demonstration let us suppose that only the first k_1 and k_2 elements of each respective columns are perturbed. So δB has only 2 nonzero columns, and the $j_1{}^{th}$ is of the form

$$\left| \delta b_{1j_1} \quad \delta b_{2j_1} \quad \ldots \quad \delta b_{k_1 j_1} \quad 0 \quad \ldots \quad 0 \right|^t \tag{2.2.6}$$

and the j_2 column is:

$$\left| \delta b_{1j_2} \quad \delta b_{2j_2} \quad \ldots \quad \delta b_{k_2 j_2} \quad 0 \quad \ldots \quad 0 \right|^t . \tag{2.2.7}$$

Now two columns of $(I + B^{-1}\delta B)$ are not unit vectors, namely columns j_1 and j_2. The $j_1{}^{th}$ column will be denoted by

$$\left| \psi_1 \quad \psi_2 \quad \cdots \quad \psi_{j_1-1} \quad (1+\psi_{j_1}) \quad \psi_{j_1+1} \cdots \quad \psi_n \right|^t \tag{2.2.8}$$

where

$$\psi_i = \sum_{p=1}^{k_1} \bar{b}_{ip} \delta b_{pj_1} \, , \tag{2.2.9}$$

and the j_2th column as,

$$\left| \phi_1 \quad \phi_2 \quad \cdots \quad \phi_{j_1-1} \quad (1+\phi_{j_1}) \quad \phi_{j_1+1} \cdots \quad \phi_n \right|^t \tag{2.2.10}$$

where

$$\phi_i = \sum_{p=1}^{k_2} \bar{b}_{ip} \delta b_{pj_2} \, . \tag{2.2.11}$$

Again by inspection the j_1^{th} and j_2^{th} elements of $\hat{x}$ are found by solving

$$\left| \begin{array}{cc} 1+\psi_{j_1} & \phi_{j_1} \\ \psi_{j_2} & 1+\phi_{j_2} \end{array} \right| \left| \begin{array}{c} \ddot{x}_{j_1} \\ \ddot{x}_{j_2} \end{array} \right| = \left| \begin{array}{c} x_{j_1} \\ x_{j_2} \end{array} \right| . \tag{2.2.12}$$

Then for $k \neq j_1, j_2$:

$$\ddot{x}_k = x_k - \not{\phi}_k \ddot{x}_{j_1} - \phi_k \ddot{x}_{j_2} . \tag{2.2.13}$$

If all the elements of a column are to be perturbed then it is necessary to have the entire inverse computed. In such a case it would be wiser to use the factorization modification algorithm. In many cases though, especially in Input Output analysis, only a few elements are modified, and the rest are just scaled. In such cases the above method becomes very appealing since the simpler closed form equations provide greater insight into the system.

SECTION 2.3

ROW PERTURBATIONS

This section nearly duplicates the presentation of the previous section except that the perturbation vectors are rows instead of columns.

Without loss of generality let us suppose that the first k elements of the j^{th} row are to be perturbed and that a constant multiple of the row is to be added to the entire row. Then δB is an nxn matrix which has all zeroes for its elements except for the j^{th} row. The j^{th} row is:

$$\left| \delta b_{j1}+cb_{ji} \quad \delta b_{j2}+cb_{j2} \quad \cdots \quad \delta b_{jk}+cb_{jk} \quad cb_{jk+1} \quad \cdots \quad cb_{jn} \right| .$$

(2.3.1)

Again the matrix $(I + \delta BB^{-1})$ is of the same form as (2.1.11) of Chapter 2.1, but the j^{th} row is replaced by:

$$\left| \gamma_1 \quad \gamma_2 \quad \cdots \quad \gamma_{j-1} \quad (1+\gamma_j+c) \quad \gamma_{j+1} \cdots \quad \gamma_n \right| , \qquad (2.3.2)$$

where

$$\gamma_m = \sum_{p=1}^{k} \delta b_{jp} \bar{b}_{pm} , \qquad (2.3.3)$$

and $(\bar{b}_{ij}) = B^{-1}$. By inspection the solution of $(I + \delta BB^{-1})z = y$ is, for $i \neq j$:

$$z_i = y_i \tag{2.3.4}$$

while

$$\sum_{m\neq j}^{k} \alpha_m y_m + (1 + \alpha_j + c) z_j = y_j \tag{2.3.5}$$

$$= \sum_{m\neq j}^{k} \alpha_m y_m + \alpha_j y_j - \alpha_j y_j + (1 + \alpha_j + c) z_j \tag{2.3.6}$$

$$= \sum_{p=1}^{k} \delta b_{jp} x_p - \alpha_j y_j + (1 + \alpha_j + c) z_j \tag{2.3.7}$$

and therefore

$$z_j = \frac{(1 + \alpha_j) y_j - \sum_{p=1}^{k} \delta b_{jp} x_p}{(1 + \alpha_j + c)} \tag{2.3.8}$$

where x is the solution of the unperturbed system $Bx=y$. If we denote $z_j - y_j$ as t then

$$z = y + te_j \tag{2.3.9}$$

where e_j is the unit vector of the j^{th} component. Denoting the solution of the perturbed system as $\ddot{x}$, as in Chapter 2.1,

$$\ddot{x} = B^{-1}z = B^{-1}(y + te_j) = x + tB^{-1}e_j \; . \qquad (2.3.10)$$

Now suppose that two rows are to be perturbed, say rows j_1 and j_2. Again purely for the simplicity of demonstration suppose that only the first k_1 and k_2 elements of each respective rows are perturbed. So δB has only 2 nonzero rows, the $j_1{}^{th}$ of which is of the form:

$$\left| \delta b_{j_1 1} \quad \delta b_{j_1 2} \quad \cdots \quad \delta b_{j_1 k_1} \quad 0 \quad \cdots \quad 0 \right| , \qquad (2.3.11)$$

and the j_2 row is:

$$\left| \delta b_{j_2 1} \quad \delta b_{j_2 2} \quad \cdots \quad \delta b_{j_2 k_2} \quad 0 \quad \cdots \quad 0 \right| . \qquad (2.3.12)$$

Now two rows of ($I + \delta BB^{-1}$) are not unit vectors, namely rows j_1 and j_2. The $j_1{}^{th}$ row will be denoted by

$$\left| \gamma_1 \quad \gamma_2 \quad \cdots \quad \gamma_{j_1-1} \quad (1+\gamma_{j_1}) \quad \gamma_{j_1+1} \cdots \quad \gamma_n \right|$$

$$(2.3.13)$$

where

$$\gamma_i = \sum_{p=1}^{k_1} \delta b_{j_1 p} \bar{b}_{pi} \; , \qquad (2.3.14)$$

and the j_2^{th} row as,

$$\left| \phi_1 \quad \phi_2 \quad \cdots \quad \phi_{j_1-1} \quad (1+\phi_{j_1}) \quad \phi_{j_1+1} \cdots \quad \phi_n \right| \tag{2.3.15}$$

where

$$\phi_i = \sum_{p=1}^{k_2} \sigma b_{j_2 p} \bar{b}_{pi} \ . \tag{2.3.16}$$

Solving $(I + \sigma BB^{-1})z = y$, for $i \neq j_{1}, j_2$:

$$z_j = y_j \tag{2.3.17}$$

while for $i = j_1, j_2$:

$$\begin{vmatrix} 1+\phi_{j_1} & \phi_{j_2} \\ \phi_{j_1} & 1+\phi_{j_2} \end{vmatrix} \begin{vmatrix} z_{j_1} \\ z_{j_2} \end{vmatrix} = \begin{vmatrix} \ddot{y}_{j_1} \\ \ddot{y}_{j_2} \end{vmatrix} \tag{2.3.18}$$

where

$$\ddot{y}_{j_1} = (1 + \phi_{j_1}) y_{j_1} + \phi_{j_2} y_{j_2} - \sum_{p=1}^{k_{j_1}} \sigma b_{j_1 p} x_p \tag{2.3.19}$$

and

$$\ddot{y}_{j_2} = (1 + \phi_{j_2})y_{j_2} + \phi_{j_1}y_{j_1} - \sum_{p=1}^{k_{j_2}} \delta b_{j_2 p}x_p \ . \qquad (2.3.20)$$

Redefining t to be $z_{j_1} - y_{j_1}$ and defining u to be $z_{j_2} - y_{j_2}$, then the perturbed solution is:

$$\ddot{x} = B^{-1}z = B^{-1}y + tB^{-1}e_{j_1} + uB^{-1}e_{j_2} \ . \qquad (2.3.21)$$

If an entire row is perturbed then only its associated column of the inverse is needed in the computation of the perturbed solution vector. The computation requires 2n+1 multiplications, (column perturbation would require about n^2.)

The above presentation lends particular insight into the perturbation of row elements in linear equations. When several rows are perturbed then the perturbation of the solution is a linear combination of the associated columns of the nominal inverse. Therefore we can make a rough guess on whether solutions are sensitive to perturbations of several rows of the IO matrix by inspecting the associated columns of the nominal inverse.

CHAPTER 3
A NONLINEAR INPUT OUTPUT MODEL

As we have seen both factorization modification and solution perturbation have comparative advantages in the solution of modified linear equations. Factorization modification has the advantage of simplicity of data storage, that is, when a row or column is changed, then only that row or column of the modified factorization is changed. Also the repetitive modification of the linear equations is very stable numerically since the refactorized matrix is machine identical[5] to the factorized matrix when the modified system of linear equations is completely decomposed from scratch. The disadvantage of factorization modification is that n^2 multiplications must be performed, and n^2 matrix elements must be read in from secondary storage. In some cases solution perturbation requires much less computation than factorization modification to achieve the same result. The disadvantage of solution perturbation is that if many elements in different rows are to be changed, as in the case of modifying all the elements in a single column, then much or all of the nominal inverse must be computed. Also repetitively computing new solutions due to changes in the elements of the matrix is difficult by this method. In this section we will utilize the comparative advantages of both methods in solving a nonlinear Input Output model.

----

[5] "Machine identical" means that the bit patterns of the two matrices are identical.

There has been very little work presented in the literature about nonlinear IO models, most of it only in the way of existence proofs, Sandberg[6] and Lahiri[7], and very little empirical application. This may be due to the difficulty of obtaining data on the nonlinearity of the transactions, for just obtaining the transactions matrix data is a difficult task in itself.[6] But the nonlinearity of the transaction elements may not be due so much to the nonlinearity of the input requirements per unit output for each firm. Rather the nonlinearity may be due to the average input requirements per unit output, that is the a_{ij}'s, shift from the production techniques of firms that cannot increase output towards the production techniques of those firms that can. A good example of this type of nonlinearity is the transactions to the oil industry if oil demand would change. If the demand for oil by consumers and industry would drop drastically then the transactions for drilling equipment and real estate would drop disporportionally to the drop in total oil output since oil wells would last for a longer period of time. Given more time there would be fewer chances that wells that are drilled end up to be dry, etc. It is likely that at lower oil demand more oil drilling would be internally financed, and thus the amount of interest charges would be proportionally less.

----

6 The transactions matrix is AX where X is a diagonalization of the total demand vector. The transactions matrix is the interindustry data that governments actually collect, the A matrix is then derived from the transactions matrix after the total demand vector is computed by summing the rows of the transaction matrix and adding the final demand vector to the resultant vector.

Without scarcity driving up prices firms are less willing to take chances, they may curtail research and development activity. Thus the composition of value added changes. As we shall see in the example below, if the demand for oil would rise the a_{ij}'s for the oil industry would shift toward the production techniques of the exploratory firms. New production techniques such as extraction of oil from shale would become more profitable, and average production shifts towards this technique. By classification of firms into two groups, those that can and cannot increase output, we have an elementary procedure by which we can represent the nonlinearity of transactions.

Let j be the index corresponding to the oil industry. Suppose that oil total output is represented by

$$x_j = x^n_j + x^o_j \ , \tag{4.1.1}$$

where x_j, x^n_j, x^o_j are total demand for oil, total demand for oil supplied by new oil, and total demand for oil supplied by old oil respectively. We assume that only the producers of new oil can expand the output of oil to a level greater than x^o_j. The transaction from the i^{th} industry to the oil industry is

$$t_{ij} = a_{ij}x_j = a^o_{ij}x^o_j + a^n_{ij}(x_j - x^o_j) \ . \tag{4.1.2}$$

Therefore

$$a_{ij}(x_j) = \frac{t_{ij}}{x_j} = \frac{(a^o_{ij} - a^n_{ij})x^o_{ij}}{x^j} + a^n_{ij} \qquad (4.1.3)$$

for $x_j \geq x^o_j$. (4.1.3) exemplifies the shift in the technical coefficients mentioned in the preceding discussion.

Though oversimplified, (but the linear IO model is even more simplified,) this example does illustrate that empirical work in this area may be more practical than was previously thought. It can also open up the possibility of treating investment as a direct input to production instead of treating it as final demand. Investment cannot be realistically treated as a linear function of total output since if existing capital is used at full capacity the only way output can increase is by investment. Thus investment would make up an considerable portion of costs. Alternatively if capital is not being used at full capacity then there will be little investment till excess capacity has depreciated away. Observation of plant capacity levels, and the catagorization of firms in a industry may give us the essential information needed to produce a viable nonlinear IO model.

The nonlinear IO model in the following exposition has an restriction in the form of the nonlinearities. It is not the most general model that could be presented, though the formulation does encompass the situations that we would most likely encounter in the real world. We assume that the nonlinearities of the tranasctions are only a function of the buying industries, that is

$$t_{ij} = t_{ij}(x_j) \; . \tag{4.1.4}$$

This is a crucial assumption in this algorithm for it tremendously reduces the order of the computation. Though this is a restriction to the analysis, it is difficult to visualize an example where a transaction element is a direct function of total demand other than the buying industry. Even in the literature though this assumption is seldom made or utilized, the examples presented usually are of this form. Also we have assumed a functional form for the nonlinearities. Though this assumption is not crucial to the exposition, and it can easily be relaxed, we shall see that this representation is computationally advantageous.

A truncated power series will be used as the functional form of the nonlinearities. The use of only three terms is only for the simplicity of demonstration.

$$t_{ij}(x_j) = \bar{a}_{ij}(x_j)x_j = \bar{a}_{ij}[\alpha_{ij} + \beta_{ij}(\frac{x_j}{\bar{x}_j}) + \gamma_{ij}(\frac{x_j}{\bar{x}_j})^2]x_j \tag{4.1.5}$$

where

$$1 = \alpha_{ij} + \beta_{ij} + \gamma_{ij} \; . \tag{4.1.6}$$

Note that $\bar{a}_{ij}=a_{ij}(\bar{x}_j)$. We denote the matrix $\bar{A}$ and vectors $\bar{x}$ and $\bar{y}$ as the nominal technical coefficients matrix, and nominal total and final demand vectors respectively. If $\alpha_{ij}=1$, and $\beta_{ij}=\gamma_{ij}=0$

for some $a_{ij}(x_j)$, then the function is constant, as in the linear IO model. Varying the values of β_{ij} and γ_{ij} from zero introduces nonlinearity in the technical coefficient. But as long as constraint (4.1.6) holds then the nominal output vector $\bar{x}$ is still the solution of the nonlinear IO model when the final demand vector is $\bar{y}$.

To introduce the concept of the model, let us assume that only the j^{th} industry has nonconstant technical coefficients. These technical coefficients are only a function of the total output of the j^{th} industry, that is, x_j. Let $\ddot{y}$ be a vector different from $\bar{y}$ and let $\ddot{x}^1$ be the solution of

$$(I - \bar{A})\ddot{x}^1 = \ddot{y} \; . \tag{4.1.7}$$

Clearly there is little chance that $\ddot{x}^1$ is the solution of the nonlinear equation

$$(I - A(\ddot{x}))\ddot{x} = \ddot{y} \; . \tag{4.1.8}$$

Let

$$\Delta A(x) = A(x) - A \; . \tag{4.1.9}$$

Note that $\Delta A(x)$ is nonzero only for the j^{th} column. By solution perturbation

$$(I - A(\ddot{x}))\ddot{x} = (I - A)(I - (I - A)^{-1}\Delta A(\ddot{x}))\ddot{x} = \ddot{y} \tag{4.1.10}$$

or

$$(I - (I - A)^{-1}\Delta A(\ddot{x}))\ddot{x} = \ddot{x}^{1} \tag{4.1.11}$$

where $\ddot{x}^{1}$ is the solution of (4.1.7).

The j^{th} row of 4.1.11 is

$$[1 - \sum_{i=1}^{n} \bar{b}_{ji}\delta\, a_{ij}(\ddot{x}_j)]\ddot{x}_j = \ddot{x}_j^{1} \tag{4.1.12}$$

where

$$(\bar{b}_{ij}) = (I - A)^{-1} , \tag{4.1.13}$$

and

$$(\delta a_{ij}(x_j)) = \Delta A(x) . \tag{4.1.14}$$

Since the $\ddot{x}_j$ is the only unknown in (4.1.12) we need only solve this scalar nonlinear equation for $\ddot{x}_j$. If the nonlinearities are represented as a truncated power series as in (4.1.5), then (4.1.12) reduces to

$$[1 - \theta((\frac{x_j}{\bar{x}_j}) - 1) - \sigma((\frac{x_j}{\bar{x}_j})^2 - 1)]\ddot{x}_j = \ddot{x}_j^1 \qquad (4.1.15)$$

where

$$\sigma = \sum_{i=1}^{n} \bar{b}_{ji}\bar{a}_{ij}\beta_{ij} \text{ , and} \qquad (4.1.16)$$

$$\theta = \sum_{i=1}^{n} \bar{b}_{ji}\bar{a}_{ij}\gamma_{ij} \text{ .} \qquad (4.1.17)$$

Then (4.1.15) can be solved by a numerical method such as Newton's method.

The advantage of the functional representation (4.1.5) is that the effects of the nonlinearities of all the elements of the j^{th} column of $A(x)$ are expressed in σ and θ. If more terms are added to (4.1.5) then only similar terms are then added to (4.1.15). Note that only the j^{th} row of the inverse needs to be computed to solve for $\ddot{x}_j$, but once we have solved for $\hat{x}_j$ we now know all the values of the technical coefficients in the j^{th} column. Using factorization modification we can compute the entire solution without the computing the inverse, as we would have had to do if we use solution perturbation only. During the factorization modification we can simultaneously compute the solution $\ddot{x}$, thus saving an extra pass through the matrix.[3]

----

[3] This is only important of course if the computer installation does not have enough main memory to hold the entire matrix.

The extension to case of nonlinearities in more than one column of A is quite straightforward. If k columns of A have nonlinearities, then the k corresponding rows of $(I - \bar{A})^{-1}$ must be computed and a k^{th} order nonlinear matrix equation corresponding to (4.1.15) is solved.

This algorithm is a tremendous savings over the usual Newton's method solution of this problem, which would be

$$x^{k+1} = x^k - (I - \frac{\partial}{\partial x}T)^{-1}[x^k - A(x^k)x^k - y] \tag{4.1.18}$$

where x^k is the approximate solution at the k^{th} iteration, and T is a vector valued function whose j^{th} element is

$$t_j = \sum_{i=1}^{n} a_{ij}(x_j)x_j \ . \tag{4.1.19}$$

For each iteration (4.1.18) requires approximately $n^3/3 + 2n^2 + ikn$ multiplications where n is the order, i+1 is the number of terms in the truncated power series, and k is the number of nonlinear columns. The method proposed here requires only about $k^3/3 + (i+1)k^2$ multiplications. If n=100, k=5, and i=3, then an iteration of (4.1.18) requires about 353,000 multiplications, while the method presented here requires about 162 multiplications. Thus the proposed method has a computational savings by a factor of 2,180!

Most importantly this method gives us a very simple means to compute the "marginal inverse", that is compute the incremen-

tal total output due to an increment in final demand. The marginal inverse is the inverse of the Jacobian matrix evaluated at a solution. Sandberg[6,Theorem 1] has shown that the inverse of the Jacobian evaluated at a solution will approximate the perturbation of solutions around the solution due to perturbations of final demand. If the vector c is the perturbation in final demand, and the vector x^p is the actual perturbation of total demand then

$$x^p = J^{-1}c + \Delta(c) \tag{4.1.20}$$

where

$$\frac{||\Delta(c)||}{||c||} \to 0 \quad \text{as} \quad ||c|| \to 0 . \tag{4.1.21}$$

The norm operator is the Euclidean. In our example in which only column j is nonlinear, if $(j_{ik}) = J$, then:

$$j_{ik} = -\bar{a}_{ik}, \text{ for } k \neq i,j \tag{4.1.22}$$

$$j_{kk} = 1 - \bar{a}_{kk}, \text{ for } k \neq j \tag{4.1.23}$$

and

$$j_{ij} = -\bar{a}_{ij} - (x_j)\frac{d}{dx_j}a(x_j) \text{ for } i \neq j, \tag{4.1.24}$$

$$j_{jj} = 1 - \bar{a}_{jj} - (x_j)\frac{d}{dx_j}a(x_j) \ . \tag{4.1.25}$$

Since the Jacobian matrix is identical to the IO matrix except for the j^{th} column we can use factorization modification to replace the j^{th} column of the nominal IO matrix by the the associated column of the Jacobian matrix. In doing so we have effectively computed the factorization of the Jacobian.

At this point it is worthwhile to digress to a fundamental result of IO analysis. In the linear IO model price is computed by the identity

$$(I - A^t)p = v \quad \text{or} \quad p = A^t p + v \ . \tag{4.1.26}$$

The j^{th} row of (4.1.26) reads: the price of good j, that is, p_j equals the average unit costs of inputs, that is, $(A^t p)_j$ plus the cost of value added, that is, v_j. In this equation profit rates must be assumed, and the costs are average, not marginal. Assuming nonincreasing returns[8] suppose that we are given a final demand vector y and the corresponding total demand vector x which is the solution of the nonlinear IO model. Then we can find a price vector p such that it is profit maximizing for each industry to produce the total demand vector x when the final

----

[8] Concavity of the technical coefficients functions and value added functions would imply nonincreasing returns.

demand vector is y. The equation which computes this price level is:

$$J^t p = \hat{v} \ . \tag{4.1.27}$$

where the i^{th} element of $\hat{v}$ is:

$$v(x_i) + (x_i)\frac{\partial}{\partial x_i} v(x_i) \ . \tag{4.1.28}$$

We have allowed for nonlinearities in value added, and as for the transaction elements, the nonlinearities are only a function of that particular industry. Of course this definition of value added excludes profits. We can easily verify that the price vector which is the solution of (4.1.27) is the price vector that makes x the profit maximizing output vector. For the j^{th} industry the per unit profit is

$$p_j - \sum_{i=1}^{n} a_{ij}(x_j) - v(x_j) \ . \tag{4.1.29}$$

Therefore total profits are:

$$\pi_j = x_j [p_j - \sum_{i=1}^{n} a_{ij}(x_j) - v(x_j)] \ . \tag{4.1.30}$$

Taking the derivative with respect to x_j we have

$$\frac{d}{dx_j}\pi_j = 0 = p_j - \sum_{i=1}^{n} p_i a_{ij}(x_j) + v(x_j)$$

$$- x_j \left[\sum_{i=1}^{n} p_i \frac{d}{dx_j} a_{ij}(x_j) + \frac{d}{dx_j} v(x_j) \right] \tag{4.1.31}$$

or

$$p_j - \sum_{i=1}^{n} p_i \left[a_{ij}(x_j) + (x_j)\frac{d}{dx_j} a_{ij}(x_j) \right] = v(x_j) + (x_j)\frac{d}{dx_j} v(x_j) \; . \tag{4.1.32}$$

Notice that (4.1.24) and (4.1.25) are the general form of the elements of the Jacobian matrix J. Also the right hand side of (4.1.32) corresponds to (4.1.26). By differentiating each of the industry's profits function with respect to that industry's total output we see that (4.1.26) is the first order condition for profit maximization.

DISCUSSION AND CONCLUSIONS

A summary ot the relative advantages and disadvantages of solving modified systems of equations by factorization modifca- tion using elementary transformations and by solution perturba- tion was presented in the beginning ot Chapter 3. The conclusion of the discussion was that neither method had a clearcut advan- tage over the other. When the two methods are used in conjunc- tion with each other, as in the nonlinear IO model, the resulting algorithm can be very efficient.

The efficient solution of IO equations has been largely ignored by economists. This is evident by the fact that up to now only large batch computer systems were used to solve IO equa- tions. Using the algorithms discussed in this paper a 368th ord- er IO matrix was factorized on a minicomputer installation.[9] The solution computation of the solution for an arbitrary final demand vector required 8 seconds user time and 12 seconds system time.[10]

----

[9] The computer was a Digital Electronics Corporation PDP11/50. The computer was operating under the UNIX operating system developed by Bell Laboratories. The algorithms were coded in C, the principle language in which much of the UNIX system was coded.

[10] The Unix operating system indicates two types of program timings, one for user time, which is the time required to perform the actual computation in the user's program area. The other is denoted as system time which is the computation required by the operating system to perform principly the input output functions. Since a 368th order matrix requires approximately 1 megabyte of storage, the computation of physical block addresses of the data consumed the largest portion of the total computation. System time would be significantly reduced by performing raw, ie., pure direct memory access (DMA) input output.

Double precision arithmetic was used by both the decomposition and solution programs. As a test of the numerical stability of the algorithms on actual data, the Bureau of Economic Analysis (BEA) 368th order IO matrix was decomposed, and then a system of equations utilizing this matrix was solved. The residuals were of the order of 10^{-14} when double precision arithmetic was used. Thus the use of elementary transformations has not been detrimental to numerical stability.

As a test of the nonlinear IO model, nonconstant technical coefficients were introduced into five columns of the 1967 368th order BEA IO matrix. The iterative solution perturbation algorithm required 3.3 seconds user and 6.1 seconds system time, while the factorization algorithm which computes the entire solution vector required 31 seconds user and 27 seconds system. Since the solution perturbation algorithm computes the total output for the industries which have nonlinear input technical coeffficients, we can check the computation of the factorization algorithm by comparing the elements in the solution vector to those computed by the solution perturbation algorithm. The residuals were of the order of 10^{-14}.

For a small set of nonlinear industries, the computation involved in the solution of a nonlinear IO model is about two or three times the computation involved in solving a factorized system of equations. In terms of computation, there is little that bars the incorporation of the algorithms in empirical IO

research. It may be futile to consider the construction of a empirical nonlinear IO model with all of the columns being nonlinear, at least this is so for the present. Many times researchers who utilize IO models focus most of their attention upon one or two, or a small group of industries. The effects of alternative technical coefficients of a small group of industries are often analyzed. The framework presented could easily be molded to such applications. Finally it is suggested that interindustry data collection should gear some of its efforts toward the determination of capacity levels of the individual firms. Doing so will allow the construction of the nonlinear investment functions needed to make the endogenous determination of investment levels in IO models possible.

REFERENCES

[1] R. Bellman, Introduction to Matrix Analysis. New York: McGraw-Hill, 1970, 2nd ed.

[2] A. Sameh and R. Bezdek, "Methods for Increasing the Computational Efficiency of Input-Output and Related Large Scale Matrix Operations," Center for Advanced Computations Document No. 66, Univ. of Illinois, May 1973.

[3] K. Noh and A. Sameh, "Computational Techniques for Input-Output Econometric Models," Center for Advanced Computations Document No. 134, Univ. of Illinois, Sept 1974.

[4] M. Aoki, Introduction to Optimization Techniques. New York: Macmillan, 1971.

[5] A. Householder, The Theory of Matrices in Numerical Analysis. New York: Blaisdell, 1964.

[6] I. Sandberg, "A Nonlinear Multisectored Input-Output Model of a Multisectored Economy," Econometrica, Vol. 41, No.6, pp. 1167-1182, Nov. 1973.

[7] S. Lahari, "Input-Output Analysis with Scale-Dependent Coefficients," Econometrica, Vol. 44, No. 5, pp 947-961, Sept. 1976.

[8] J. Sherman and W. Morrison, "Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix," Annals of Mathematical Statistics, No.1, Vol. 21, pp 124-126, March 1950.

[9] D. Hawkins and H. Simon, "Note: Some Conditions of Macroeconomic Stability," Econometrica, Vol. 17, No. 3, pp. 245-248, July 1949.

[10] G. W. Stewart, Introduction to Matrix Computation. New York: Academic, 1973.

[11] C. Moler "Matrix computations with Fortran and Paging," Communications of the Association of Computing Machinery, Vol. 15, pp. 268-270, April 1972.

[12] G. H. Golub, P. E. Gill, W. Murray, and M. A. Saunders, "Methods for Modifying Matrix Factorization," Stanford Univ, Rep. STAN-CS-72-322, 1972.

[13] P. E. Gill and W. Murray, "A Numerically Stable Form of the Simplex Algorithm," Linear Algebra and Its Applications., Vol. 7, pp. 99-138, 1973.

[14] G. J. Bierman "Additional Comments on "Multistage Least-Squares Parameter Estimation," IEEE Trans. Automat. Contr. Vol. AC21, pp. 883-885, Dec. 1976.

[15] E. Isaacson and H. Keller, Analysis of Numerical Methods. New York: Wiley and Sons, 1966.

[16] G. Forsythe and C. Moler, Computer Solutions of Linear Algebraic Systems, Englewood Cliffs, N.J.: Prentice Hall, 1967.

[17] H. Chenary and P. Clark, Interindustry Economics, New York: Wiley, 1959.