HIGH ORDER MATRIX COMPUTATION
ON THE UNIVAC*

A year ago at the Wayne meeting in Detroit of this society, a paper was presented describing a set of matrix routines which had been programmed, at that time, for the UNIVAC. Since then a considerable amount of computation has been done with these routines, which is believed to be of sufficient interest to warrant the presentation of a summary, both of the routines and of the results of the calculations completed to date. These calculations have been concerned in the main with the inversion of large Lecatief matrices.

Figure 1 is a list of routines which are currently available. All of these routines are recorded on a single magnetic tape which has been entitled the "Matrix Math" tape. To perform one of these operations, the operator is required, upon signal from the computer, to enter into the memory the code corresponding to the desired routine as shown in the left hand column. The computer then automatically selects the proper blocks of coding from the Matrix Math tape to construct the desired program.

The first six routines are the computational routines, which perform the operation indicated in the second column. These can be applied to matrices, having up to 100,000 elements. In terms of square matrices, the upper limit is of the order 310. All of the arithmetic operations and storage are in floating decimal form. At least ten digits are retained throughout any computation. The result of any of these routines can be used as input to any of the other routines, since all input and output matrices have the same form. The methods employed will be discussed at greater length below.

*Presented at the Pittsburgh meeting of the Association for Computing Machinery, May 3, 1952, by H. Rubenstein and J. Rutledge
The remaining routines, with the exception of the "Modify" and "Norm" routines, are for data handling purposes. The format of the matrices, as designed for the computational routines, is well adapted for computer handling, but cannot be dealt with easily by humans. To overcome this difficulty routines code "9", "Input Edit", and "Conversion" were prepared.

The routine designated by the code "Modify", computes the inverse of a matrix which differs from a matrix whose inverse is known in only one row or column. This is accomplished by a sequence of multiplications in which several of the computational routines are controlled by the "Modify" routine.

The "Norm" routine calculates the indicated quantity, which is used in certain error and condition estimates.

The "Transpose" routine simply transposes a matrix, using the computational form of the matrix.

Code "9" designates a group of routines which, from the computational form of the matrix, prepares a tape for printing the matrix in floating decimal form. Work is now in progress to provide a fixed decimal form.

The "Input Edit" routine converts a matrix which has been recorded on tape in the form of a list of numbers to the computational form. This list may be prepared by unityping or produced from cards using the Conversion routine.

A trained unitypist can record elements consisting of 10 digits plus sign at a rate of better than 600 per hour.

The Conversion routine accepts a tape transcription of a deck of punched cards, each containing one element with its indices, checks for sequence and obvious mispunching, and then produces a list suitable for use by the "Input Edit" routine. In addition, the Conversion routine sums each row and column and records these sums on tape in a form suitable for printing.

The methods employed by the computational routines were selected on the basis that they would yield a high degree of accuracy in the results produced and, at the same time, be generally applicable. This meant that methods had to be chosen so that no advantage was taken of the special nature of certain matrices.
The adaptation of these methods to UNIVAC was dictated by the characteristics of the computer. The finite size of the high-speed memory coupled with the high-speed input-output devices which are associated with the computer were the primary considerations. The memory size made it apparent that an entire matrix of high order could not be handled internally at one time. This was not a serious disadvantage, however, because a scheme could be adopted where a portion of the matrix could be placed in the memory at one time, and the high-speed input-output equipment could be called upon to move small blocks of elements into and out of the internal memory without a significant loss of computing time as required. Hence, a scheme employing partitioned matrices was adopted. In addition, this method offered the advantage of an economical control of the accuracy of an inversion during the course of the calculation.

By a partitioned matrix, we meant a matrix which has been divided into a number of smaller matrices each having the same dimensions. The partitioned matrix may then be considered as a matrix having matrix elements, called submatrices.

Paralleling this division, the programming is separated into two levels of routines: low and high level. The high level routines carry out operations on matrices having submatrices as elements, using the low level routines as arithmetic subroutines to multiply and reciprocate submatrices.

To illustrate this technique, consider a matrix multiplication. Let A and B be two matrices which are conformable and are to be multiplied, and let their product be C. If A and B are partitioned into \( M \times M \) and \( M \times P \) submatrices of order \( n \times m \) and \( m \times p \) respectively, then their product C will contain \( N \times P \) submatrices of order \( n \times p \). Now, if the submatrices of A, B, and C are called \( A_{ij} \), \( B_{ij} \), and \( C_{ij} \) respectively, then

\[
C_{ij} = \sum_{k=1}^{M} A_{ik} B_{kj}.
\]

Note, again, that the quantities in this formula are matrices. To evaluate this formula two levels or types of operations are needed. The first, which is the high level operation, consists in selecting the required submatrix elements \( A_{ik} \) and \( B_{kj} \) and the second, which is the low level operation, consists of multiplying submatrix \( A_{ik} \) by submatrix \( B_{kj} \) to produce the elements of the matrix terms in the above sum.
The "high level" matrices have elements which do not have a commutative multiplication, and in at least the case of inversion, the numerical operations carried out are not those used in the case of unpartitioned matrices. Therefore, we should investigate the validity of operations on these "high level" matrices, and their equivalence to the corresponding operations on non-partitioned matrices. The demonstration of this equivalence for multiplication is basically simple, but notationally rather involved, and will not be presented here. The inversion operation will bear more detailed treatment. The inversion routines use the Gauss elimination procedure with minor modifications. This procedure consists of the application of a sequence of elementary row operations: multiplication of a row by a scalar and/or addition of two rows. Row operations can also be represented as pre-multiplication by elementary matrices, i.e., matrices obtained from the I matrix by a single row operation. In this form, the Gauss elimination method can be represented as follows:

Given a matrix \( A \), it is desired to find \( A^{-1} \) such that

\[
AT^{-1} = I
\]

Premultiplying both sides of the above by these,

\[
T^{-1}E_1AT^{-1} = T^{-1}E_1 \cdot I
\]

Therefore,

\[
A^{-1} = T^{-1}E_1
\]

and \( A^{-1} \) has been determined. It is assumed that the algorithm by which each successive

\[
F_k \cdot x
\]

is determined from

\[
K \sum_{i=1}^{\lambda} E_1A
\]

is known. If \( N \) is the order of the matrix \( A \), then \( N \) of the \( E_1 \) are obtained by reciprocation of an element of the

\[
T^{-1}E_1A
\]
Obviously, the element to be reciprocated should be as large as possible to minimize error; also, the reciprocation should be as accurate as possible. Since the low level routine operates entirely within the internal memory, it was practical to select, for each reciprocation, the largest remaining element of the matrix and move it to an appropriate location by including permutation matrices among the $E_i$. In the case of high level routine, a similar repositioning would require excessive time, therefore, it is not done. However, the inverse produced by the low level routine is checked by multiplication and improved by an iterative scheme as follows:

\[
E_0 = I - \lambda_0 A \\
\lambda_0^{-1} = \lambda_0^{-1} + E_0 \lambda_0^{-1}
\]

This is repeated until the maximum element of the error matrix is less than a preassigned tolerance. The iteration is an error-squaring process up to the limit imposed by rounding errors in the multiplication. For example, in the Leontief type matrices, the maximum element of $E$ can generally be made of order $10^{-3}$. This acts to offset the possibility of an accidental small divisor which might seriously disturb the accuracy. Since it need be carried out only $N$ times, the time required is small compared to that of an entire inversion or overall improvement of the final inverse. It is recognized that this method does not absolutely guarantee high accuracy in every inverse; however, that the programs are fairly successful in avoiding excessive roundoff in many practical cases can be seen from the results reported below.

Occasionally, in practice, it is desirable to alter slightly some of the elements of a matrix after its inversion has been performed, due to a change in the situation represented, or perhaps the discovery of an error in the original data. This can be done according to the equation:

\[
B = A + uv \\
B^{-1} = A^{-1} + A^{-1} uv A^{-1} \\
\frac{1 + v A^{-1}}{1 + \nu A^{-1}}
\]

where $u$ is a column, $v$ a row vector, $A$ the matrix whose
Inverse is known, B the matrix whose inverse is desired and differs from A only in one row or column. Either u or v contains only a single unit, serving to indicate which row or column is to be modified. Thus, about $3n^2 - n$ multiplications, plus some selection, are required to produce an inverse for changes in a single row or column of the original matrix.

The computation which has been performed using these routines will be discussed in three groups:

1. The sequence of multiplications, reported last year
2. The calculations concerned with the inversion of Leontief matrices of various orders
3. The inversion of several matrices of very poor condition.

Error estimates have been obtained for each calculation.

The multiplication sequence, which was discussed last year, arose in an application of the Leontief input-output theory to local situations. A product of four matrices was formed. The order of the matrices ranged from 10 x 10 to 30 x 40. The three multiplications required to form this product were performed in two sequences to provide a check, i.e., both $A \times (B \times (C \times D))$ and $((A \times B) \times C) \times D$ were formed. The maximum discrepancy between the elements of the two products was three in the tenth significant digit, the last carried. The time required for each sequence of multiplications was about ten minutes.

The second group accounts for the bulk of the computation time to date, and seems to be a principal application of high order matrix computation. A Leontief matrix is a matrix having positive elements of order unity in the main diagonal, and negative elements elsewhere, such that the sum of the elements in any row is positive. This means that the diagonal elements are quite dominant. All the eigenvalues are in the neighborhood of unity and conditioning is very good.

These matrices are the so-called inter-industry input-output matrices, derived from an input-output table which states the proportion of the output of industry $j$ consumed by industry $j$ with $i$ and $j$ ranging over whatever set of industries or industry groups is being considered. The economic interpretation of this problem has been widely discussed. It may be found in general terms in recent issues of "Fortune" or "Business Week".
We make use of two criteria which should be defined at this point. The first is an upper bound for the error in an approximate inverse as given by Hotelling, in terms of several norms. As defined previously, the norm of a matrix is the square root of the sum of the squares of the elements of the matrix. It can be shown that

\[ N(A^{-1} - C_0) \leq N(C_0) \frac{N(E_0)}{1-N(E_0)} \]

where \( A^{-1} \) is the true inverse, \( C_0 \) the approximation inverse being evaluated, and \( E = I - AxC_0 \). Hence, the right hand side may be used to obtain an upper bound on the error of a computed inverse. The second is Turing's N-Condition Number which is defined also in terms of norms, as

\[ C = \frac{1}{N(A) N(A^{-1})} \]

where \( n \) is the order of the matrix. Both quantities have been computed for most of the matrices which have been inverted. Derivations of these criteria are contained in References 1 and 2 respectively. The Leontief matrices which have been dealt with are illustrated in Fig. 5. For each matrix in the table an inverse was calculated, a multiplication was performed to obtain an error matrix \( E = I - AA^{-1} \), the norms of \( A \), \( A^{-1} \) and \( E \) were obtained, and tapes were prepared for printing \( A \), \( A^{-1} \), \( E \). In most cases, input editing was also required. It will be noted that all these matrices are quite similar, except as to order; this is to be expected, since all represent the American economy, with differing fineness of description. The numbers in parentheses are the orders of the respective matrices as used in computation. They were augmented to permit partitioning from the given orders by adjoining the required number of rows and columns of the I matrix or appropriate order.

The times given in the table are only for the inversions. The multiplication required in each case, slightly less time than the inversion. The error limit in the third column is on the norm of the difference between the approximate inverse obtained and the true inverse, and not the error in a single element; since this included from 1,600 to 35,100 elements in the various cases, the error in each element must be quite small. Note that the condition numbers are quite close to unity. It seems evident that the accumulation of round-off errors in the inversion process will not limit the use of Leontief matrices in Economic Analysis and Prediction.
The computation involved in the inversion of the order 190 matrix provides an interesting sample of recent UNIVAC operation. Time was obtained by the Air Comptroller's office on the Census UNIVAC for the purpose. During the period March 18 to 26, 1952, 100 hours of computer time were assigned to this problem. Of this time, 42.5 hours were required for the actual inversion, 32.1 hours for the multiplication, and 7.2 hours for miscellaneous processing, computation of norms, etc. A total of 81 hours of productive computation was obtained. As part of a training program, the Air Comptroller's office assigned programming personnel with little or no operating experience as operators for the entire period. During the inversion itself, when the computer required no attention except when the check circuits detected abnormal operation, 42.2 out of 47.5 hours, or 91.6% of the time, produced useful results. Included in this was a period of 10 hours 35 minutes during which the computer operated completely without attention. In this period it carried out about 75,000,000 operations without a failure or interruption of any kind. It should be noted that the check multiplication was undertaken not to verify the proper operation of the computer, but to assess the effect of round-off errors. The internal check system of UNIVAC reduces the probability of undetected error to a completely negligible level.

A feature of these programs which has been quite useful in practice is a set of rerun routines. These make it possible to re-enter any program after an interruption for any reason, with a minimum of re-computation. Re-entry points are provided at intervals of 15 to 30 seconds throughout the computational routines, so that despite the volatile nature of the acoustic memory, very little re-computation is required even in the event of a power failure.

Fig. 6 is a summary of the work which we have done with ill-conditioned matrices.

The first matrix in this table is derived from a Leontief matrix by subtracting 1 from an I matrix. The Leontief matrix was the one of order 45 listed in Fig. 5. The diagonal of the resulting matrix was not dominant, and all terms were of the same sign. So far as we know, this matrix has no practical significance; its inversion was an experimental computation. As can be seen from the figure, the condition of this matrix was much worse than that of its companion Leontief matrix, and consequently the errors in the computed inverse are higher. The maximum element of the inverse is about 500, so that the error limit of 10^{-3} indicates that the larger elements
of the inverse are accurate to at least five, and probably about seven significant digits. The slightly higher time was due to the rather extensive iteration required to improve the reciprocals of several diagonal submatrices. The maximum element in several error matrices was reduced by a factor of 100 by this process.

The remaining two matrices treated in Fig. 6 are members of an interesting group. Although these matrices are small, they are difficult to invert; the difficulty increasing rapidly with the order. Note the rather large condition numbers and small determinants.

The upper half of Fig. 7 gives the elements of the order 5 matrix of this group; the lower half, its inverse. The rule of formation for this group is obvious. These matrices look like a set of mathematical curiosities, but actually they arise from a practical problem in curve fitting. In this problem, it is desired to obtain the parameters of a polynomial having a specified moments and two specified roots. The elements of the matrix are the coefficients in the set of simultaneous equations, the solutions of which are the desired parameters. The matrix is a function of the number of specified moments, but not of the moments themselves; hence, since the inverse is obtained, a number of separate problems can be dealt with easily. The inverse given in Fig. 7 is the exact inverse, and contains no computational error. Note that the elements are all integers. This fact should make this group quite useful for experimental work. Despite the remarkably poor condition, we were able to obtain inverses correct to 5 significant digits for the order 5, and three significant digits for the order 6.
The work described and presented in this paper is the result of the joint effort of a number of people on the staff of the Eckert-Mauchly Division. In addition to the authors Dr. Herbert F. Mitchell, Jr., was responsible for the basic layout of the problem and at all times gave encouragement to development of the various phases of the matrix project; Albert B. Tenik, and Stephen E. Wright contributed the low level routine; Bernard N. Riskin, and Katherine Curtin worked in the processing routines. Donald Temme and James Kelley, who are members of the U.S. Air Force, did the major work on the "modify" routine.

Bibliography


2. Rounding-off Errors in Matrix Processes - A.M. Turning


4. Matrix Algebra Programs for the UNIVAC - H.Rubinstein and J.D. Rutledge

5. Solution of Matrix Equations of High Order - Dr. H.F. Mitchell
<table>
<thead>
<tr>
<th>CODE</th>
<th>OPERATION</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>± 1</td>
<td>D ± AB</td>
<td>A, B, D CONFORMABLE</td>
</tr>
<tr>
<td>± 2</td>
<td>± AB</td>
<td></td>
</tr>
<tr>
<td>± 3</td>
<td>G ± AF</td>
<td>F, G COLUMN VECTORS</td>
</tr>
<tr>
<td>± 4</td>
<td>I ± AB</td>
<td>A*B SQUARE, I IDENTITY MATRIX</td>
</tr>
<tr>
<td>± 5</td>
<td>AA^T</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>A⁻¹, D(A)</td>
<td></td>
</tr>
</tbody>
</table>

**FIG. 1a UNIVAC MATRIX LIBRARY PROGRAMS**

**PART I**
<table>
<thead>
<tr>
<th>CODE</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODIFY</td>
<td>GIVEN A AND A_-1, FIND B_-1</td>
</tr>
<tr>
<td></td>
<td>B DIFFERS FROM A IN ONLY ONE ROW</td>
</tr>
<tr>
<td>NORM</td>
<td>COMPUTES N(A) = (∑ a^2_{i,j})^{1/2}</td>
</tr>
<tr>
<td>TRANSPOSE</td>
<td>A^T</td>
</tr>
<tr>
<td>9,s,r</td>
<td>PREPARE FOR PRINTING</td>
</tr>
<tr>
<td>INPUT EDIT</td>
<td>CONVERT LIST TO COMPUTATIONAL FORM</td>
</tr>
<tr>
<td>CONVERSION</td>
<td>CONVERTS FROM PUNCHED CARD TO LIST FORM</td>
</tr>
</tbody>
</table>

**FIG. 1b UNIVAC MATRIX LIBRARY PROGRAMS**

**PART II**
\[ A \times B = C \]
\[
\begin{array}{c}
N \times M \\
M \times P \\
N \times P
\end{array}
\]

\[
C_{ij} = \sum_{k=1}^{M} A_{ik} \times B_{kj}
\]

- \( A_{ij} \) \text{ SUBMATRIX ELEMENT OF A} \\
- \( B_{ij} \) \text{ SUBMATRIX ELEMENT OF B} \\
- \( C_{ij} \) \text{ SUBMATRIX ELEMENT OF C}

\text{FIG. 2 - PARTITIONED MATRICES}
ITERATION SCHEME

\[ E_o = I - A_o^{-1} A \]
\[ A_1 = A_o + E_o A_o \]

MODIFICATION

\[ B = A + uv \]
\[ B^{-1} = A^{-1} + \frac{A^{-1}uv A^{-1}}{1 + vA^{-1} u} \]

FIG. 4
<table>
<thead>
<tr>
<th>ORDER</th>
<th>TIME OF INVERSION</th>
<th>ERROR LIMIT</th>
<th>DETERMINANT</th>
<th>TURING'S CONDITION NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>38(40)</td>
<td>45 Min</td>
<td>$2.80 \times 10^{-8}$</td>
<td>0.80349</td>
<td>1.12</td>
</tr>
<tr>
<td>40</td>
<td>45 Min</td>
<td>$2.45 \times 10^{-8}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>47(50)</td>
<td>58 Min</td>
<td>$3.21 \times 10^{-8}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44(45)</td>
<td>50 Min</td>
<td>$4.14 \times 10^{-8}$</td>
<td>3.027 x $10^{-3}$</td>
<td>1.58</td>
</tr>
<tr>
<td>190</td>
<td>42 1/2 HRS</td>
<td>$3.67 \times 10^{-7}$</td>
<td>5.257 x $10^{-6}$</td>
<td>1.52</td>
</tr>
</tbody>
</table>

**FIG. 5 LEONTIEFF MATRICES**
<table>
<thead>
<tr>
<th>ORDER</th>
<th>TIME OF INVERSION</th>
<th>ERROR LIMIT</th>
<th>DETERMINANT</th>
<th>TURING'S CONDITION NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>44(45)</td>
<td>65 Min</td>
<td>$1.91 \times 10^{-3}$</td>
<td>$4.733 \times 10^{-52}$</td>
<td>121.97</td>
</tr>
<tr>
<td>5(8)</td>
<td>2 Min</td>
<td>$1.374 \times 10^{-9}$</td>
<td>$1.2 \times 10^5$</td>
<td></td>
</tr>
<tr>
<td>6(8)</td>
<td>2 Min</td>
<td>$3.220 \times 10^{-16}$</td>
<td>$3.8 \times 10^6$</td>
<td></td>
</tr>
</tbody>
</table>

**FIG. 6 ILL CONDITIONED MATRICES**
\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\
1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\
1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\
1/5 & 1/6 & 1/7 & 1/8 & 1/9
\end{bmatrix}
\]

\[
\begin{bmatrix}
5 & 300 & -2,100 & 4,200 & -2,520 \\
-60 & -2,400 & 13,900 & -40,320 & 25,200 \\
210 & 6,300 & -52,920 & -117,600 & -75,600 \\
-280 & -6,720 & 58,800 & -134,400 & 88,200 \\
126 & 2,520 & -22,680 & 52,920 & -35,280
\end{bmatrix}
\]

FIG. 7 - AN ILL CONDITIONED MATRIX
AND ITS INVERSE