This paper was presented at the Wayne Conference on Automatic Computing Machinery and Applications March 27-28, 1951, Detroit, Michigan by Joseph Rutledge and Harvey Rubinstein.
Matrix Algebra Programs for the UNIVAC

This paper is a report on the development of proposals, made by Dr. Mitchell at the Association for Computing Machinery convention at Rutgers in the Spring of last year, concerning a set of programs for UNIVAC for the execution of high order matrix calculations. The first part of the paper will review Dr. Mitchell's proposals and discuss the programs which have been developed. The second part will briefly discuss the application of these routines to the calculation of a problem in Economics.

General programs have been designed for the solution, in a reasonable length of time, of equations involving addition, multiplication and inversion of real matrices of arbitrary order, by a method which permits a considerable degree of control over the accuracy of the solution. These programs may be employed without manual intervention for matrices having up to about 100,000 elements, or about order 310 in terms of square matrices.

To digress for a moment, I would like to sketch briefly the essential features of UNIVAC. UNIVAC is a general purpose high speed electronic digital computer having an acoustic memory capacity of 1,000 twelve-character words. The characters may be alphabetic or numeric. If a word represents a numeric quantity, the first digit position is reserved for the sign. Both instructions and data are placed in this memory. The primary

method of transferring information into and out of the computer is via magnetic tapes at a rate of approximately 720 words per second, in groups of 60, called blocks. In addition these tapes can be used as an external memory of large capacity.

Obviously, it was not feasible nor particularly desirable to provide enough internal storage capacity to hold the high order matrices considered here. In these programs the method of partitioned matrices, which permits treating a portion of the matrix at a time, was adopted, for this reason and because it offers the advantage of economical control of the accuracy of an inversion by the elimination method. Thus, solution was resolved into two levels: 1) solution of a form of the given problem, in which the elements are themselves matrices, and 2) the solution of the resulting submatrix equations. Programs dealing with the first level are called high level routines, and those dealing with the second, low level routines. The submatrices are limited to about 100 elements.

The high level programs will solve the following equations directly:

1) \( D \pm A B = C \), where \( A, B, \) and \( D \) are conformable, but not necessarily square.

2) \( A B = C \)

3) \( I \pm A B = C \), where \( I \) is an identity matrix supplied by the program.

4) \( A A^* = C \), the transposition being performed by the program.

5) \( A F = C \), where \( F \) is a column matrix. This is treated as a special case because of its usefulness and because it requires fewer tape units.
6) \( A^{-1} = C \), where \( C \) is a non-singular square matrix.
7) \( A^{-1} = 0 \) where \( |A| 
eq 0 \), and \( A \) is a submatrix.
8) \( |A| = 0 \)

In the course of these high level calculations, only two low level routines are required: 1) to solve \( \delta \pm \eta \) and 2) to find \( A^{-1} \) where \( A \), \( \eta \), \( \delta \) are submatrices.

All matrices used as input to or resulting as output from these routines have the same standard form. This permits a sequence of matrix operations to be performed without additional editing of data; i.e., the output of any operation can be used as any one of the inputs to the next operation.

A ninth routine has been prepared to transform this standard form into a form suitable for printing.

It has not been our purpose to develop the smallest possible number of programs to perform these operations, but rather to develop a set of programs which carry out most efficiently the most widely used matrix operations.

While the programs deal directly only with matrices having real elements, operations on complex matrices may be performed by using a sequence of the above operations. Various other common matrix problems, such as determination of eigen-values, may also be handled. (See Hotelling, Berkeley Symposium on Mathematical Statistics and Probability, 1945-46).

The inversion is performed by the elimination method, as discussed by Dr. Mitchell in his paper. The submatrix method permits the improvement of the inverse of a large matrix by iteratively approximating the inverses of the diagonal submatrices.
This results in a considerable saving of time, if the inversion
is at all difficult. If a submatrix inversion fails to converge
under the iteration procedure, routine 7 is employed to repartition
the submatrix into $2 \times 2$ submatrices and invert the result
by a procedure similar to the high level program.

With the exception of this rather unlikely case, the only
manual operations required are 1) preparation of the input
matrix tapes; 2) mounting of the instruction and input matrix
tapes, and an output tape, and 3) typing two words into the
memory via the Supervisory control keyboard. The first of these
is a control word which selects the proper program and the second
identifies the output tape. While Computer failure should be
quite rare, any computation as long as these may be must be
guarded against it. In case it should occur, the starting pro-
cedure is repeated; with the addition of a "re-run indicator" in
the control word. The last word printed out at Supervisory
Control by the computer is typed in when requested by the com-
puter. A special routine determines the appropriate point
from which to repeat that part of the computation voided by the
failure. Automatically, the "re-run" routine reconstitutes
control of the computation. Thus, even power failure would
entail a loss of only about 3 minutes, including operator time.

The input matrix tapes are prepared by recording the
matrix elements, ordered by rows of submatrices. The submatrix
elements are in turn ordered by rows. Since UNIVAC is a fixed
decimal machine, a programmed floating decimal system is used.
Numbers are scaled to lie in the range $1 > |a| \geq 1$ and the ex-
pONENT recorded. Preceding each row of the submatrix is re-
corded the exponent of the largest element of that row. Each 
element has associated with it a single digit exponent and this 
"row exponent". The first word in each block is reserved for 
the indices of the submatrix.

Matrix multiplication as performed by routines one to five 
is somewhat unconventional. This applies to both high level and 
low level. In each of these routines an entire row of elements 
of the product matrix is found at a time, whereas usually the 
elements of the product matrix are found separately. This was 
made necessary by the fact that both the external and internal 
memories are serial media. In computing the product AB, the 
computation runs as follows: The first element of A is multi-
plied by the first row of B, and the resultant row is stored;
the 2nd element of A is multiplied by the 2nd row of B, and the 
result added to the previous result. The process is repeated 
until the last element in the first row of A is multiplied by 
the last row of B and added to the previous results. This pro-
duces the first row of the product matrix. This procedure is 
then repeated for each of the rows of the A matrix and in turn 
produces each row of the product matrix. Some interesting 
features of the low level multiplication routine are:
1) If every element of a matrix is zero, an indicator is set in 
the submatrix which is detected by the low level multiplication 
routine, and which causes it to bypass multiplication and set 
the same zero indicator for the product.
2) before performing any computation, the exponents are 
separated from the digits of each quantity, permitting compu-
tation to be carried thru with 11 digits. It might be
mentioned at this time that since the same multiplication
routine is used in performing all the submatrix multiplications,
a great deal of effort has been expended in coding this routine
to function as rapidly as possible. The high level routines
check the orders of super and submatrices to insure their
conformability before initiating computation. Each time the
high level routine directs the low level multiplication routine
to perform a multiplication, the indices of the product to be
formed are printed on the Supervisory Control printer, so that
the progress of the computation can be noted, and for the pur-
pose of re-runs, as mentioned above.

An interesting feature of the low-level inversion routine
is the fact that before each division of the elimination is
carried out, the largest element of the submatrix is found and
transferred to the main diagonal, where it is used as divisor.
Other than this, the method is essentially the standard Gauss
elimination method. The high level routine uses a similar
elimination method. At the start of the problem, the operator
is requested to type into the computer the largest exponent
permissible for the error matrix $e = I - \alpha^{-1}$, where $\alpha$ is a
submatrix, $\alpha^{-1}$, its inverse as determined by the low level
inversion routine, and $I$ is the identity matrix of proper
order. This tolerance defines the termination of the iteration
procedure. If the iteration fails to converge, the attending
mathematician must decide when the special inversion routine,
mentioned earlier, should be called in. This is done because
the criteria for recognising non-convergence are not well enough
known or simple enough to make their inclusion in the routine
worthwhile. The determinant is automatically computed during
the first triangularization and appears on the Supervisory
S2636L printer as soon as completed. If option 3 was
selected, requiring only the computation of the determinant,
the standard inversion routine is used, and stopped at this
point. In a normal inversion, the value of the determinant
gives some indication as to the validity of the result. Here
are some rough estimates of times required by UNIVAC to per-
form multiplications and inversions of matrices of certain
representative orders:

### Multiplication

<table>
<thead>
<tr>
<th>Order</th>
<th>Eq. x Eq.</th>
<th>20%</th>
<th>Eq. x 8 Eq.</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1 hour</td>
<td>46 minutes</td>
<td>1 minute</td>
<td>9.9 minutes</td>
</tr>
<tr>
<td>100</td>
<td>7 hours</td>
<td>5 hours</td>
<td>7.5 min.</td>
<td>6.1 minutes</td>
</tr>
<tr>
<td>200</td>
<td>75 hours</td>
<td>42 hours</td>
<td>30 minutes</td>
<td>25 minutes</td>
</tr>
<tr>
<td>500</td>
<td>187.5 hours</td>
<td>141 hours</td>
<td>1 hour</td>
<td>55 minutes</td>
</tr>
</tbody>
</table>

### Inversion

<table>
<thead>
<tr>
<th>Order</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1 hour</td>
</tr>
<tr>
<td>100</td>
<td>8 hours</td>
</tr>
<tr>
<td>200</td>
<td>37 hours</td>
</tr>
<tr>
<td>500</td>
<td>200 hours</td>
</tr>
</tbody>
</table>
These programs have been designed around matrices of order 100, with considerations of efficiency being ignored for orders much less than 50; since, for the lower order matrices, operator time becomes quite significant, although only on the order of minutes, and total time for the computation is quite small. The times given for improvement of inversion refer to one iteration cycle for each submatrix inverted. The 50 percent times refer to matrices having 50 percent zero elements, randomly distributed in the matrix.

It is of interest to note that the tape time, i.e., access time in the slow memory provided by the magnetic tapes, accounts for only about 2 percent of these figures. On the other hand, when access time in the high speed memory was reduced by careful coding of one section with that in mind, a saving of almost 20 percent was effected. Obviously, the use of tapes as slow memory, as in these routines, is no great hardship.

Some of these programs have been recently used at Eckert-Mauchly Computer Corporation to direct UNIVAC No. 1 in the execution of certain matrix computations submitted by Professor Leontief, of Harvard. Professor Leontief and his associates have been studying the structure of the American economy for several years. One of their problems consists in devising tables of input - output relations among industries in the U.S. Several years ago Dr. Mitchell of our laboratory, then at Harvard, inverted a matrix of order 38 for this group. The result was a table of input - output relations on a national basis for industry groups. Considering this table as a matrix,
and multiplying it by a vector which represents the output of
certain industries, one can obtain the input to these various
industry groups from the other industry groups on a national
basis. This study, though useful, did not provide them with a
fine enough analysis of the relations between the various in-
dustry groups in the U.S. Now they are interested in studying
the effect of changes in national industry output on local in-
dustry output. To determine this, it was found necessary to
transform the national input-output coefficients by multiplica-
tion by three matrices. With these results one now can determine
what effect a change in the output of a national industry, such
as the aircraft, would have on a local industry, such as the
electric power industry.

Before continuing the discussion of this problem, some
further discussion of UNIVAC is required.

No mention has been made of any programmed checks of the
computation. The explanation for this lies in the fact that
UNIVAC is a self-checking computer. By this is meant that all
arithmetic operations are automatically performed twice, in
parallel, using entirely different circuits, the results being
constantly checked against each other. The seven pulse code
that has been adopted for representing individual characters
is such that the number of binary ones used to represent any
character is odd. Based on this fact, the parity of the number
of 1's in each digit in the memory is checked every 5 seconds.
In addition, each transfer from one register to another within
the computer is similarly checked. In view of this, and since
there is no simple mathematical check on the multiplication of
two matrices, (although there are arithmetic checks which could have easily been coded), no programmed checks are made.

To check out the programs to insure that they would instruct the computer to calculate as intended, various small matrices were devised, each of which tested different subroutines in the Low-Level multiplication. Finally, a multiplication of two order 8 matrices of random elements was used as a concluding test, and the result verified against a hand computation of a number of elements in the matrix. The random elements were chosen as check on the possibility that something was overlooked in selecting the order 2 matrices.

To check the operation of the high level routines, additional coding was introduced which caused the indices of the submatrices being multiplied to be printed out on the Supervisory Control printer. Finally, the multiplication was carried out in two orders:

\[ \{A \cdot [B \cdot (C \cdot D)]\} \text{ and } \{(A \cdot B) \cdot C \} \cdot D\]

It is felt that this gives a sufficient check on the operation of the program. We now return to the problem:

The computation which was to be performed was as follows:

\[
\begin{bmatrix}
40 & \begin{bmatrix} 30 \\ 30 \\ 30 \\ 30 \\ 10 \\ 10 \\
\end{bmatrix}
\end{bmatrix}
\]
The elements of the three smaller matrices were presented on paper and the fourth with its elements on punched cards, one card per element. The former were recorded from the keyboard of the Unityper in our standard form, and the elements of the latter were transcribed onto tape by our card-to-tape-converter, and then the computer was used to transform the tape copy of the cards into a matrix of our standard form, this operation requiring approximately 1 minute of computer time. The computation of this problem was then commenced. Ten minutes later, the results were taken from the machine. Three runs of the #2 program mentioned earlier were accomplished in this time, the longest taking some 3 1/2 minutes. The result was then edited so that the result could be printed on UNIPRINTER. The output editing required less than 1 minute.

In conclusion, we would like to thank Albert Ponik and Stephen Wright for their work in the preparation of these programs. In particular, they were responsible for the low level inversion and multiplication routines.