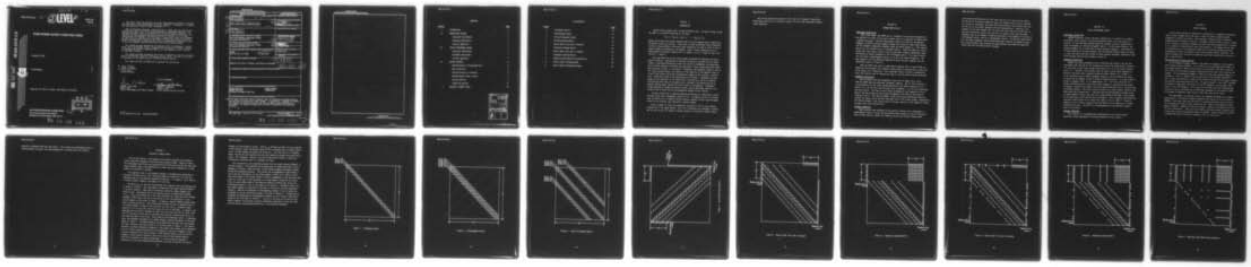
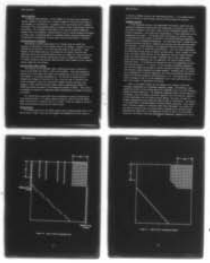
AD-A066 810 AIR FORCE WEAPONS LAB KIRTLAND AFB N MEX F/G 12/1
SOME SPARSE MATRIX COMPUTER CODES.(U)
JAN 79 S HENDRIX

UNCLASSIFIED AFWL-TR-78-216 SBIE-AD-E200 263 NL

| OF |
AD
A066810





END
DATE
FILMED
5-79
DDC

SBIE

AFWL-TR-78-216

(2) LEVEL III

AFWL-TR-78-216

ADE200263

DDC FILE COPY

AD A066810

SOME SPARSE MATRIX COMPUTER CODES

January 1979

Final Report

Approved for public release; distribution unlimited.

AIR FORCE WEAPONS LABORATORY
Air Force Systems Command
Kirtland Air Force Base, NM 87117

AFWL-TR-78-216

This final report was prepared by the Air Force Weapons Laboratory, Kirtland AFB, New Mexico, under AFOSR funds, Job Order 2304Y101. Lieutenant S. P. Hendrix was the Laboratory Project Officer-in-Charge.

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been authored by an employee of the US Government. Accordingly, the US Government retains a nonexclusive royalty-free license to publish or reproduce the material contained herein, or allow others to do so, for the US Government purposes.

This report has been reviewed by the Office of Information (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be be available to the general public, including foreign nations.

This report has been reviewed and is approved for publication.

Steven P. Hendrix
STEVEN P. HENDRIX
2d Lt, USAF
Project Officer

FOR THE COMMANDER

George L. Williams
GEORGE L. WILLIAMS
Major, USAF
Chief, Environment and Effects Branch

Thomas W. Ciambrone
THOMAS W. CIAMBRONE
Colonel, USAF
Chief, Applied Physics Division

DO NOT RETURN THIS COPY. RETAIN OR DESTROY.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFWL-TR-78-216	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): SOME SPARSE MATRIX COMPUTER CODES	5. TYPE OF REPORT & PERIOD COVERED: Final Report,
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): 2Lt S. Hendrix	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Weapons Laboratory (DYVM) Kirtland AFB, NM 87117	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 2304 Y1
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Weapons Laboratory (DYVM) Kirtland AFB, NM 87117	12. REPORT DATE: Jan[illegible]979
	13. NUMBER OF PAGES: 24
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): AFOSR Bolling AFB, DC 20332	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

SBIE

Approved for public release; distribution unlimited.

AD-E200 263

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Sparse Matrices
Banded Matrices
Blocked-Tridiagonal Matrices
Linear Systems
Eigenvalues

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report discusses several computer codes for manipulating sparse matrices and solving the associated linear systems. In particular, a package for banded or diagonally structured matrices and one for general sparse matrices are discussed. The general package also includes an eigenvalue and eigenvector routine.

A

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 013 150

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CONTENTS

ILLUSTRATIONS

SECTION I

INTRODUCTION

Sparse linear systems occur in many different areas. By sparse linear system we refer to a system of equations of the form:

$$A_{I1} X_1 + A_{I2} X_2 + A_{I3} X_3 + \ldots + A_{IN} X_N = B_I$$

where the As and Bs are known constants and Xs are the unknown and at least 85 to 90 percent of the A_{IJ} are zero. Since such a linear system must have N equations to have a unique solution, the term linear system will be used to refer to such a square system of N equations in N unknows. Such a linear system may also be represented by the matrix equation AX = B, where A is an N x N matrix, and X and B are N x 1 vectors.

Two algorithms for solving such systems are Gaussian-Elimination and Gauss-Seidel iteration. Explanations of both may be found in any good text on linear algebra. Gaussian-Elimination is done by a series of elementary row operations, each of which produces a different but equivalent linear system (equivalent means that the same X vector is a solution). Elementary row operations affect a row of the linear system, where a row consists of one equation in the equation form, or a row of the A matrix plus the corresponding element of the B vector, in matrix form. Three basic types of row operations are allowable - exchanging two rows (order of the equations is not significant), multiplying a row by a constant (multiplying both sides of an equation by a constant), and adding one row to another (adding equal quantities to both sides of an equation). Elementary column operations, though not normally useful, can be performed similarly on a column of the A matrix and the corresponding element of the X vector.

The main difficulty in applying Gaussian-Elimination to large sparse linear systems arises from a phenomenon called fill-in. In adding one row to a second row, the result will in general contain nonzero elements at every position where either of the original rows contained a nonzero element.

After a number of these row operations, the matrix fills-in with nonzero elements and is no longer sparse. This fill-in destroys any advantage a routine may be able to gain (in speed or reduced storage) if the matrix remains sparse.

Each of the routines discussed in this report is intended to manipulate large sparse matrices of a particular type or to solve the associated system of linear equations.

SECTION II

NARROW-BAND SOLVERS

FUNCTIONAL DESCRIPTION

TRI2, TRIM, PENT2, and PENTM are all intended to solve matrix equations of the form AX = B, where X and B are n-vectors and A is a sparse N x N matrix with a special structure. TRI2 and TRIM accept a matrix in which all the nonzero elements lie within the main diagonal (upper left corner to lower right corner) and the diagonals immediately adjacent above and below (figure 1). This type of matrix will be referred to as a tridiagonal matrix. PENT2 and PENTM are similar except that they accept a matrix with one additional diagonal immediately above and below the nonzero diagonals of the tridiagonal matrix. This type of matrix will be called a pentadiagonal matrix, as it has five nonzero diagonals: the main diagonal, the two diagonals immediately above the main diagonal, the two immediately below it (figure 2).

TRI2 and PENT2 each solve a single matrix equation (AX = B) while TRIM and PENTM each solve several such equations concurrently by changing the X vector and the B vector to N x M matrices. This technique allows a saving in computation, since the A matrix need be reduced only once in solving several matrix equations.

PARAMETER DESCRIPTION

TRI2 and PENT2 need four parameters, which will be names N,A,X, and B. TRIM and PENTM need one additional parameter, named M. For all four routines, N is the dimension of the linear system described by the matrix equation; the A matrix is N x N, and the X and B vectors have N elements. A is an array representing the A matrix. The rows of the A array correspond to the diagonals of the A matrix; and the columns of the A array correspond to the rows of the A matrix, (figures 1 and 2). For TRI2 and PENT2, X and B are singly-dimensioned arrays to contain the result vector and the right-hand side of the linear system, respectively, For TRIM and PENTM, X and B each contain M vectors, where M is the number of linear systems to be solved concurrently.

INTERNAL OPERATION

These routines take advantage of the special structure of a tridiagonal or pentadiagonal matrix to solve the associated linear system as quickly as possible. They can each entirely ignore the elements outside the original nonzero band,

since Gaussian-Elimination will not cause any fill-in on this type of matrix. At the end of the forward reduction step, all elements below the main diagonal will be zeros; therefore, they need not be calculated or even stored--they can be left as is, and simply ignored in the back-substitution. Similarly, the elements on the main diagonal will be ones, and since they will be divisors in the back substitution phase, they may also be ignored, thus, only the upper diagonals need to be changed in reducing the matrix--one diagonal in the case of a tridiagonal matrix; two for a pentadiagonal matrix. During back substitution, only the upper diagonals need be considered again reducing the amount of work. On a given problem, these routines are approximately two orders of magnitude faster than the Yale Sparse Matrix Package.

SECTION III

TRIPLY-TRIDIAGONAL SOLVER

FUNCTIONAL DESCRIPTION

GAUSDL iteratively solves the matrix equation AX = B, where A is a sparse N x N matrix with a special structure (described below), and X and B are N x 1 vectors. It uses the Gauss-Seidel algorithm, adapted to solve a system with a block-tridiagonal or "triply-tridiagonal" matrix. In this type of matrix, the nonzero elements lie on three sets of three adjacent diagonals. The middle set of three consists of the main diagonal and the adjacent diagonal above and below it, while the other two sets consist of three adjacent diagonals centered at some fixed distance from the main diagonal (figure 3).

PARAMETER DESCRIPTION

GAUSDL uses seven parameters--four to describe the A matrix, two for the right-hand side, and one for the convergance criteria for Gauss-Seidel iteration. N is the order of the linear system (the A matrix is N x N; the X and B vectors are N x 1). The GAP is the distance from the center set of three diagonals to either of the outer sets, measured center-to-center, NP2GP2 is N + 2*GAP + 2. It was originally made a formal parameter because it was to dimension an array, but might now be changed so that it would be computed locally. A is the A matrix, stored in diagonal form. It is dimensioned 9 x N. The first subscript of an element designates which of the nine nonzero diagonals, counted from the lowest to highest, contains the element, while the second subscript tells in which column the element appears. The X and B arrays contain the X and B vectors (solution and right-hand side, respectively). They must each be equivalenced to another array of length NP2GP2 in such a way that X(1) is equivalent to the (GAP + 2)nd element of the long array, and B must be set up similarly. This is because the subroutine subscripts these arrays in the range from -(GAP + 1) to N + GAP + 1. When all elements of the X vector change by less than DELTA in one iteration, no further iterations are made. Thus DELTA is an approximation of the maximum allowable error in the result.

INTERNAL OPERATION

This routine is a straightforward implementation of the Gauss-Seidel algorithm, taking advantage of the known structure of the A matrix.

SECTION IV

MATRIX PACKAGES

One of these packages is designed for manipulation of diagonally structured sparse linear systems, while the other is made for general sparse systems, assumming no specific structure. Both packages include routines to add two packed matrices, multiply a matrix by a constant, multiply times a vector, extract a row, column, or individual element, and solve the matrix equation AX = B, where A is a packed N x N matrix and X and B and N-vectors. The diagonal solver uses an algorithm which is arithmetically equivalent to Gaussian-Elimination after rearranging the rows; the general solver uses the Gauss-Seidel algorithm. The general package also includes an eigenvalue and-vector routine and a routine to pack a sparse matrix.

REPRESENTATION OF PACKED MATRICES

In the diagonal package, IDIAGS is the number of diagonals in the A matrix which contain nonzero elements. A is stored in an array dimensioned IDIAGS x N. The second subscript refers to the column of the A matrix; the first subscript tells which of the nonzero diagonals (counting from the lower left) contains the element. The corresponding element in IPOS tells where the diagonal appears in the matrix. That is, IPOS(I) gives the position of the diagonal contained in A(I,*), where the position is given by the directed distance from the main diagonal, the positive sense being down and left. The main diagonal is the zeroth diagonal, the element in the lower left corner is the Nth diagonal, and the upper right corner is the -Nth diagonal. The A array and the IPOS array may be dimensioned oversize in the first subscript; an additional parameter to the routine, IDIMA, is provided to dimension these arrays to that they need not be tight.

The requirements for packing a matrix are somewhat different for a general sparse matrix. The nonzero elements themselves are placed in an array called PACKED, in the order they would be found if scanning the matrix row by row from top to bottom, and left to right within each row. NNZR is the number of these nonzero elements. IROWS(I) contains the index in PACKED of the beginning of the Ith row of the matrix. ICOLS parallels PACKED, with each element containing the column index of the corresponding element of PACKED. IBIG(I) contains the index in PACKED and ICOLS of the pivot element in row I of the matrix.

MATRIX ADDITION

In adding sparse matrices, a given element of the result may correspond to nonzero elements in both matrices, a nonzero element of one but not the other, or zero elements in both matrices. An element of the result matrix will in general be nonzero if the corresponding element in either or both of the original matrices was nonzero. For the general sparse matrix, this must be checked element-by-element, while the diagonally structured matrix can be checked diagonal-by-diagonal. In either case, a new packed matrix is assembled without unpacking either matrix. The matrices to be added are called A and B, and the result matrix is called R.

MULTIPLICATION BY A CONSTANT

Multiplying a packed sparse matrix by a constant does not affect the structure of the matrix, so the pointers need not be changed. Likewise, it is not affected by the structure, so it does not need access to the pointers. Since elements of the packed array may lie outside the real matrix, they may have undefined values, and multiplying these values by a number will cause an error in some cases, so the multiplying routine must be able to determine which elements lie within the matrix and then may simply multiply those elements by the constant, without regard for any pointers.

MULTIPLICATION TIMES A VECTOR

The routines in both packages input a packed sparse matrix along with a vector called VECTOR and return a result called RESULT. Since zero elements of the matrix will not affect the result, the routines deal only with the elements which appear in the packed representation. The routine for sparse matrices computes each element of the result in turn, since the rows of the matrix are stored contiguously, while the routine for diagonal matrices traverses each diagonal in turn, multiplying each element times the appropriate element of VECTOR and adding the product to the appropriate element of RESULT. Each routine is designed to traverse its respective packed representation as simply and quickly as possible.

This routine may be used under some circumstances to avoid multiplying two large sparse matrices. If a matrix equation contains a term of the form (AB)X, where A and B are N x N matrices and X is a vector, it may be changed to the form A(BX) and evaluated by applying this routine twice.

PACKING ROUTINE

The packing routine for the general sparse matrix package accepts rows of the matrix one at a time, strips out zero elements and compresses the row, places it

in the array PACKED, and sets the appropriate pointers. It also makes preparations for solving a linear system, such as marking columns for pivoting.

EIGENVALUE ROUTINE

In the eigenvalue routine, the eigenvectors are actually found first and eigenvalues determined by multiplying the eigenvector by the matrix and taking the ratio of elements of the result vector to the corresponding elements of the original eigenvector. Any vector may be expressed as a linear combination of the eigenvectors of a matrix. The routine starts by choosing an arbitrary vector. It then multiplies this vector by the matrix, thus multiplying each eigenvector component by its corresponding eigenvalue. The resulting vector is normalized and again multiplied by the matrix, repeating this process for some specified number of iterations. In each iteration, the eigenvector with the largest eigenvalue (in absolute value) is effectively multiplied by 1, while all other eigenvectors are multiplied by some factor of smaller absolute value. After a number of iterations, the result vector consists of the eigenvector with the largest absolute value, plus some relatively very small components due to other eigenvectors. Having found one eigenvector and its eigenvalue, the routine chooses another arbitrary vector. It eliminates any component of the first eigenvector by effectively multiplying by $A - \lambda I$, where λ is the previously determined eigenvalue. The iterative scheme then proceeds as before, eventually causing the eigenvector with the next largest eigenvalue to become dominant. Roundoff error can reintroduce a component of a previous eigenvalue, so it must be eliminated occassionally during the iteration process by multiplying by $A - \lambda I$. This process can, in theory, eventually find all of the eigenvalues, but is most useful where only a few of the largest (in magnitude) are desired.

There is no apparent reason for this method to be unstable, given that extraneous eigenvectors are eliminated frequently enough. Unfortunately, some instability exists in this routine which will sometimes cause it to fail miserably, beyond the second eigenvalue. The determinant apparently does not hold the key to convergance, since the routine can either succeed or fail for positive, negative, or zero determinants and for absolute values greater, less than, or equal to 1. Diagonal dominance was also tested and seems to bear no relationship. The routine is not guaranteed to converge beyond the second largest eigenvalue. Since termination is based on number of iterations rather than convergance, the routine will not enter an infinite loop, but results must be checked by the calling routine. Experience has shown that if one eigenvalue is badly in error, all later eigenvalues will be invalid, since the corresponding eigenvector component will not be

correctly eliminated from the test vector. This project was terminated prior to finding either the reason for nonconvergance or a suitable test for a matrix.

SECTION V

SOLVING A LINEAR SYSTEM

The solving routine in the general sparse matrix package uses the Gauss-Seidel algorithm, adapted to this particular packing scheme. Several criteria for convergance were tried and discarded while writing this routine but none were found suitable, hence, the termination on either the error tolerance or a fixed number of iterations is allowed.

The algorithm used in the diagonal package is arithmetically equivalent to Gaussian-Elimination with a full pivoting scheme to minimize fill-in, but was developed and will be explained somewhat differently. In figure 4, cells with an X contain nonzero elements, and empty cells are zero.

The matrix A_{IJ} is the first partitioned into reducible rows and reducing rows as shown in figure 4. (For the present, assume that the lowest nonzero diagonal is entirely nonzero.) Rows which lie entirely above the lowest diagonal are reducing rows. Multiplying the first reducing row by an appropriate constant $\left(-A_{1,1} / A_{M+1,1} \right)$ and adding it to the first reducible row clears the first element (changes it to a zero), but due to the nature of this row operation, it also causes elements to the right to change. In general, nonzero elements appear where zeros occurred previously. Since the second reducing row has no nonzeros left of the second element, it may be used similarly to clear the $A_{1,2}$ element without causing fill-in to the left, but also causes fill-in to the right. In general, the reducing rows are used in sequence to clear the first row from left to right, with the I^{th} row used to clear the A_{1I} position while possibly causing some fill-in to the right but never to the left. Since there are N columns but only N - M reducing rows, the rightmost M elements will remain nonzero, leaving the matrix in the form shown in figure 5. The same procedure can be applied to the 2nd, 3rd, 4th through Mth rows, leaving an equivalent matrix of the form shown in figure 6. This matrix contains a uniquely determined submatrix in the upper right corner, which represents an M x M linear system consisting of this submatrix, the last M elements of the X vector (related to the last M columns of the original matrix), and the first M element of the B vector (related to the first M rows of the original matrix). Since this subsystem is uniquely determined, it can be solved (by Gaussian Elimination or any other appropriate procedure), yielding values for the last M

elements of the original X vector. With X_{N-M} through X_N known, the last equation of the original linear system now contains only one unknown with a nonzero coefficient, so that unknown can readily be solved for. With the last M + 1 elements known, the next-to-last equation has only one remaining unknown, which may now be found. This procedure, similar to the back-substitution process in Gaussian-Elimination, continues until all unknowns are found.

This discussion of the algorithm assumes that the lowest nonzero diagonal is entirely nonzero. The algorithm can be generalized to handle some zeros on the diagonal as follows. Rows with zeros on the lowest diagonal are simply skipped during the elimination process. This leaves the corresponding column without a row to use for elimination, so the column remains filled, as in figure 7 and 8. Each row skipped in this manner remains linearly independent of the reducible rows since it is not used in the reduction, so it may become a reducible row itself. Thus, for each column added to the reduced system by skipping rows during the elimination process, a row is also added, as in figure 9. (Diagonals other than the lowest diagonal are omitted for the sake of clarity in this and subsequent figures.) Rows may be exchanged if corresponding elements of the B vector are exchanged to give an equivalent system in the form of figure 10. Columns may be exchanged by exchanging the corresponding elements of the X vector, leaving the system in the form shown in figure 11. Just as in the case of no zeros on the lowest diagonal, we now have a uniquely determined subsystem to start the final solution process and back-substitution as in the previous paragraph.

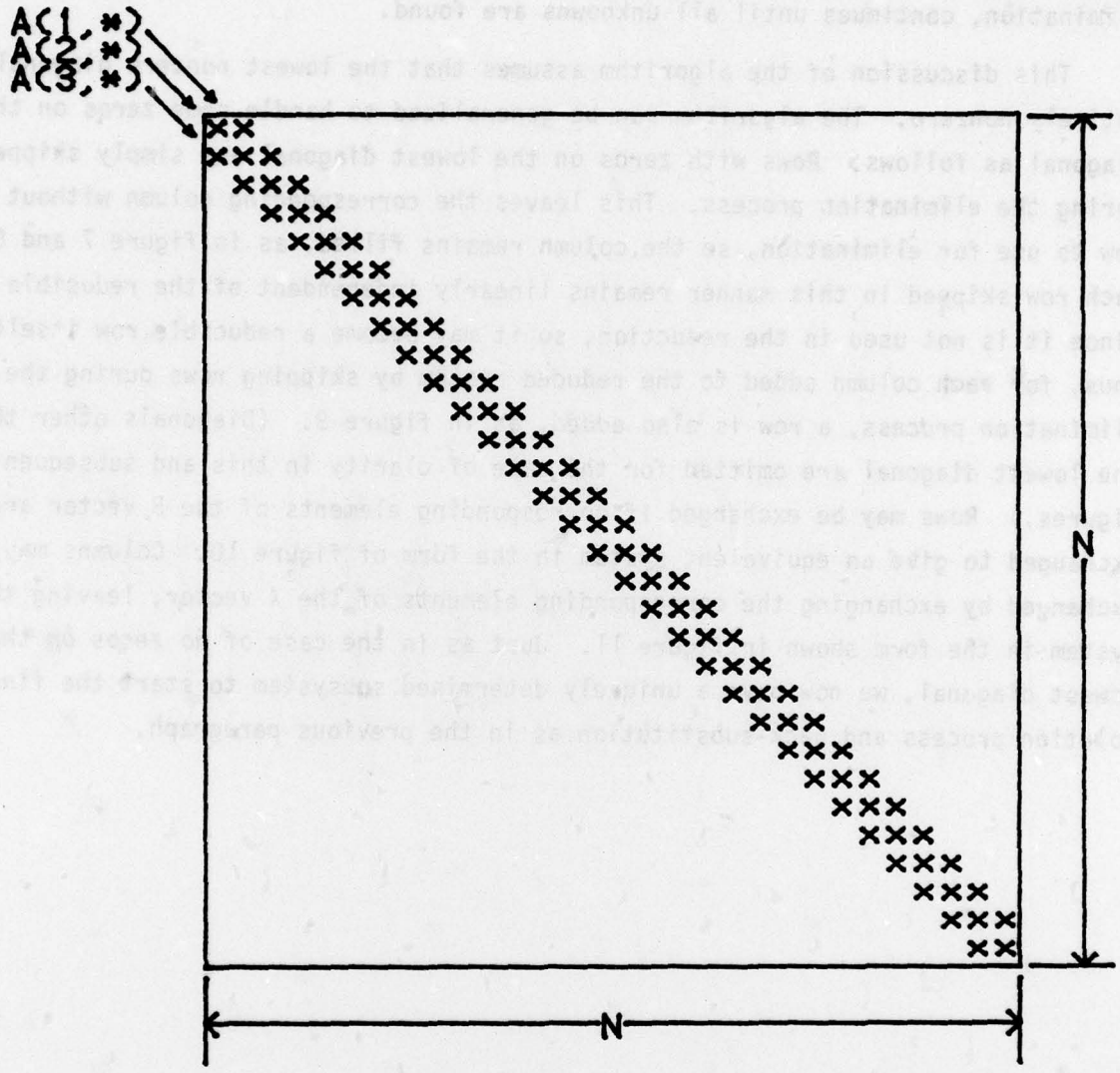


Figure 1. Tridiagonal Matrix

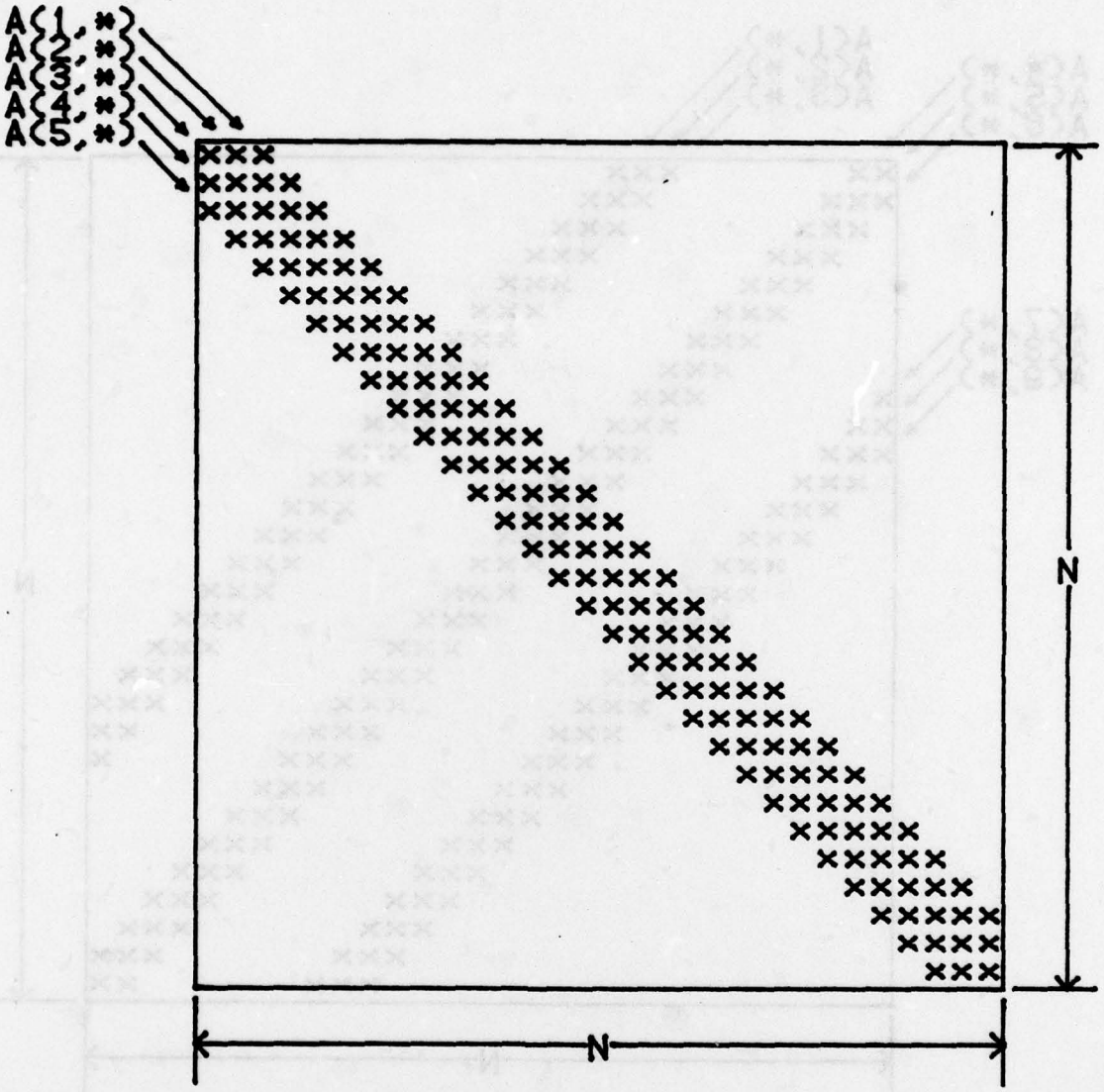


Figure 2. Pentadiagonal Matrix

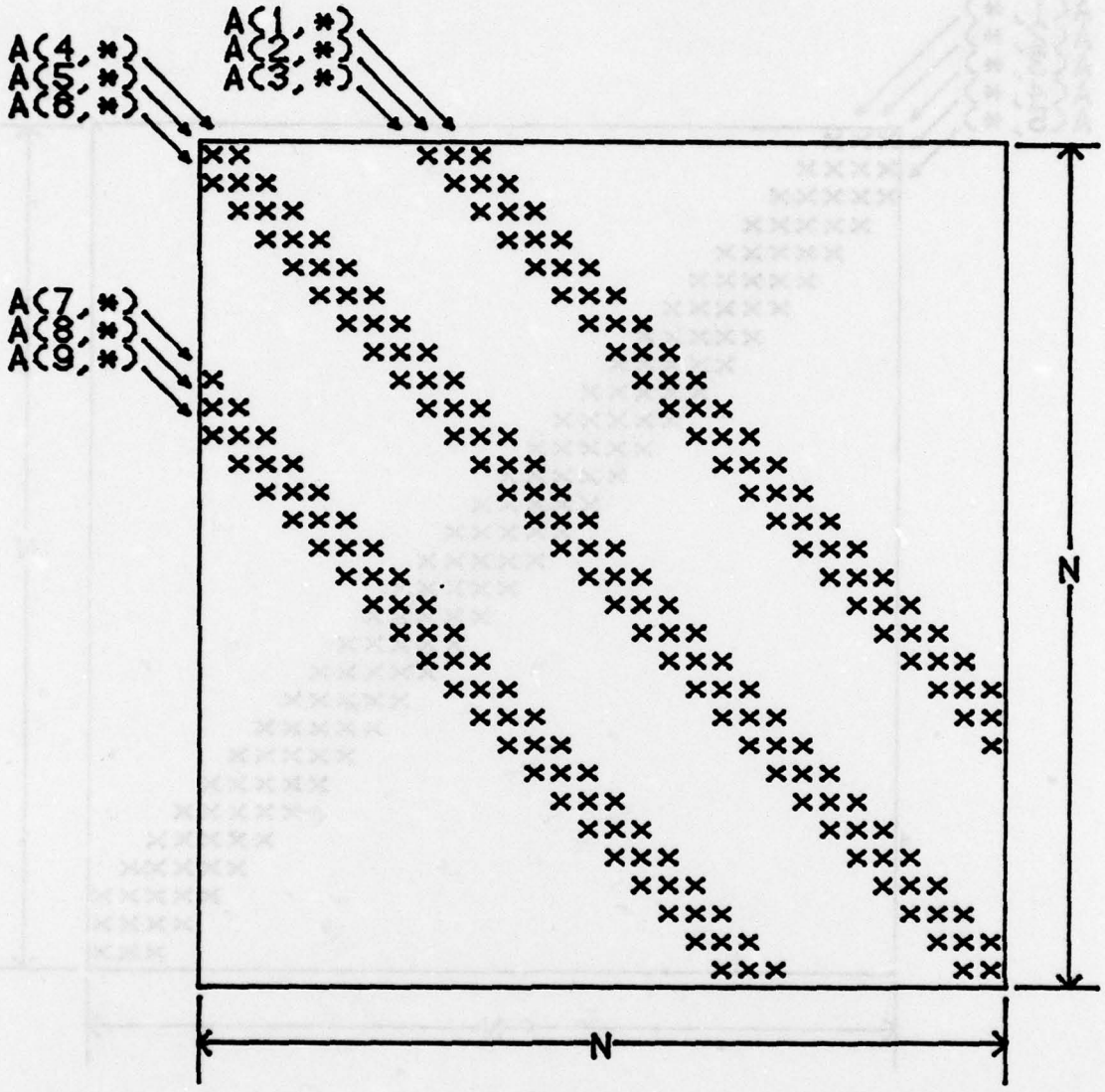


Figure 3. Triply-Tridiagonal Matrix

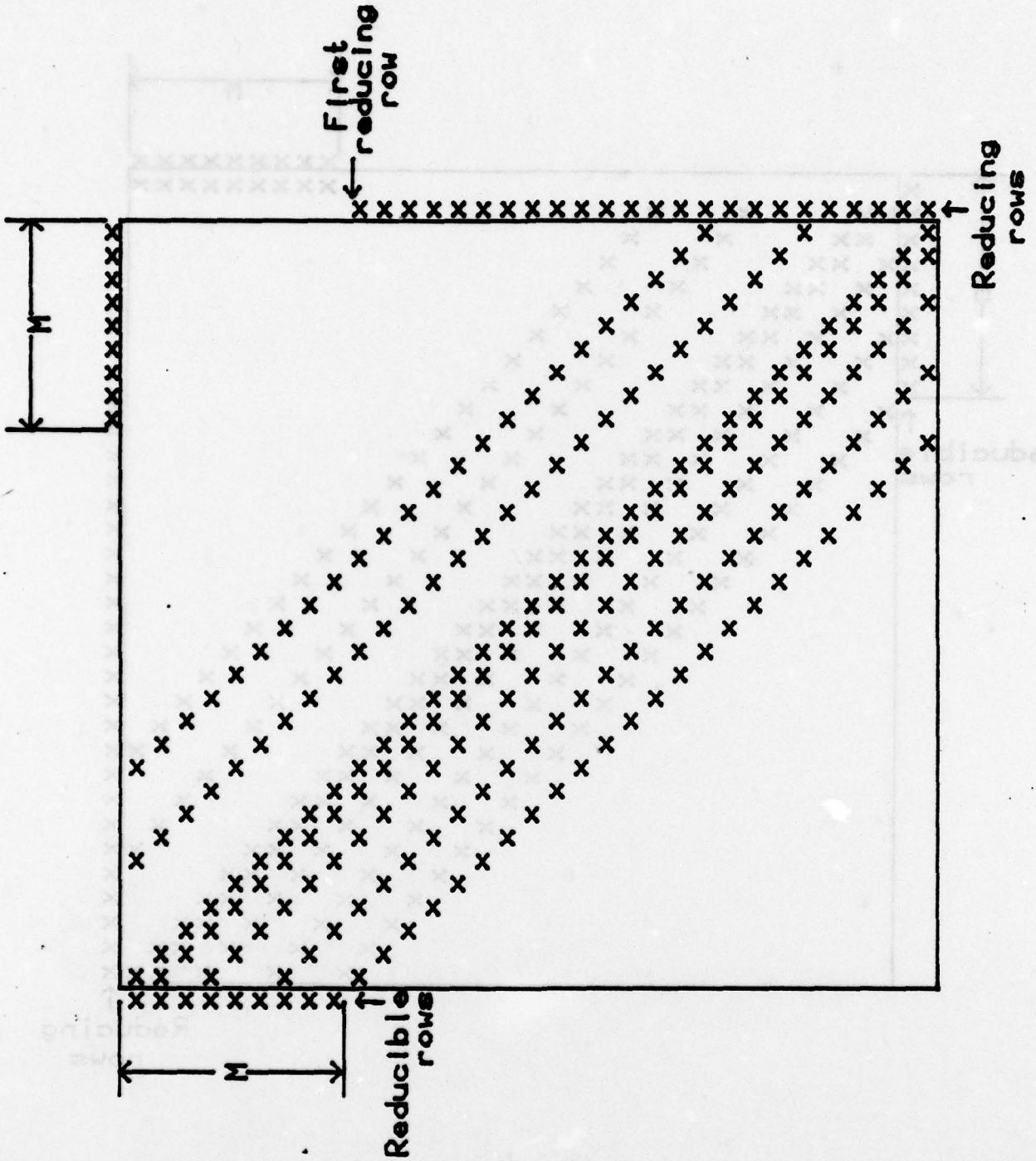


Figure 4. Partitioned Sparse Matrix

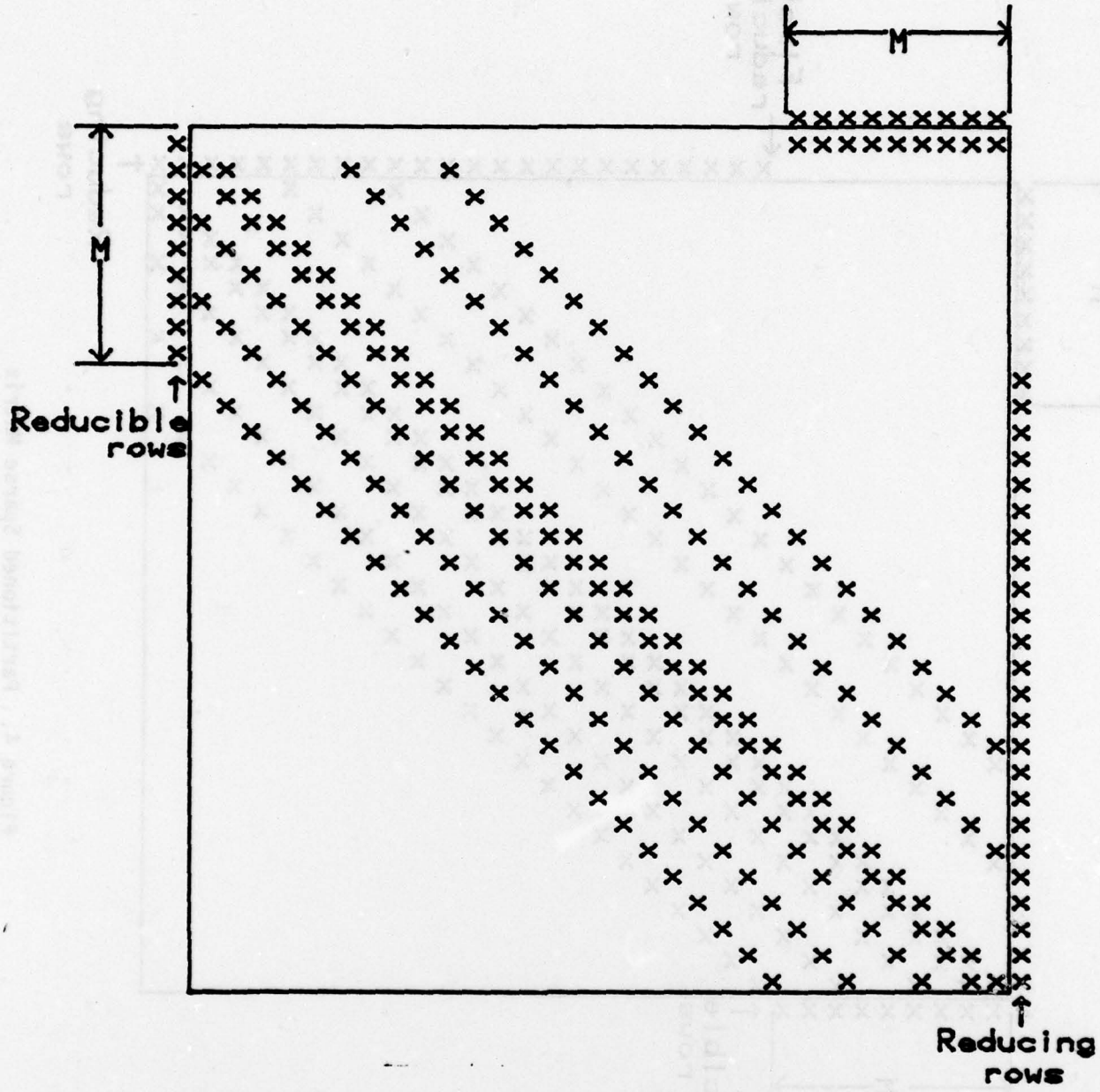


Figure 5. Matrix After First Row is Reduced

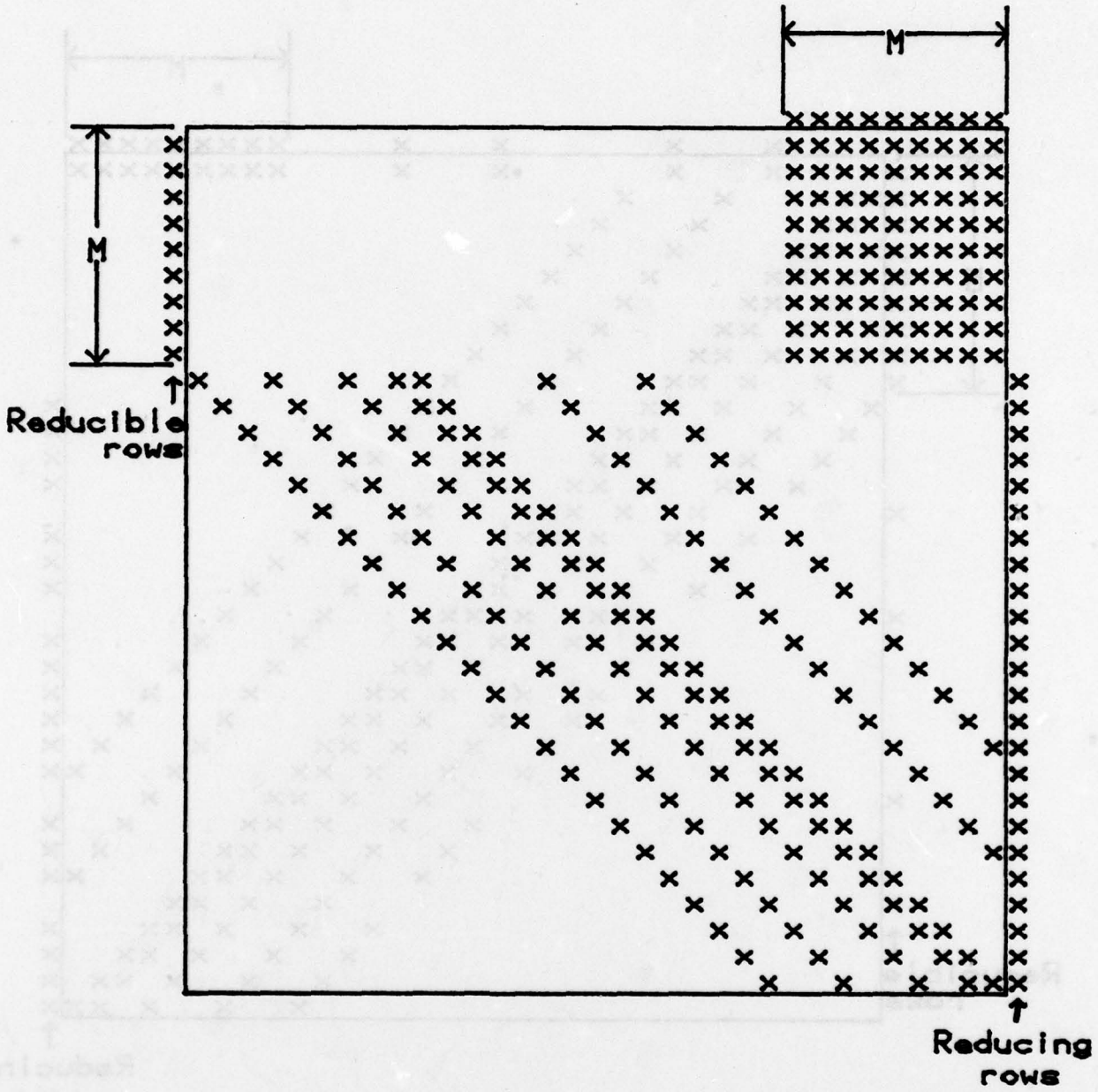


Figure 6. Completely Reduced Matrix

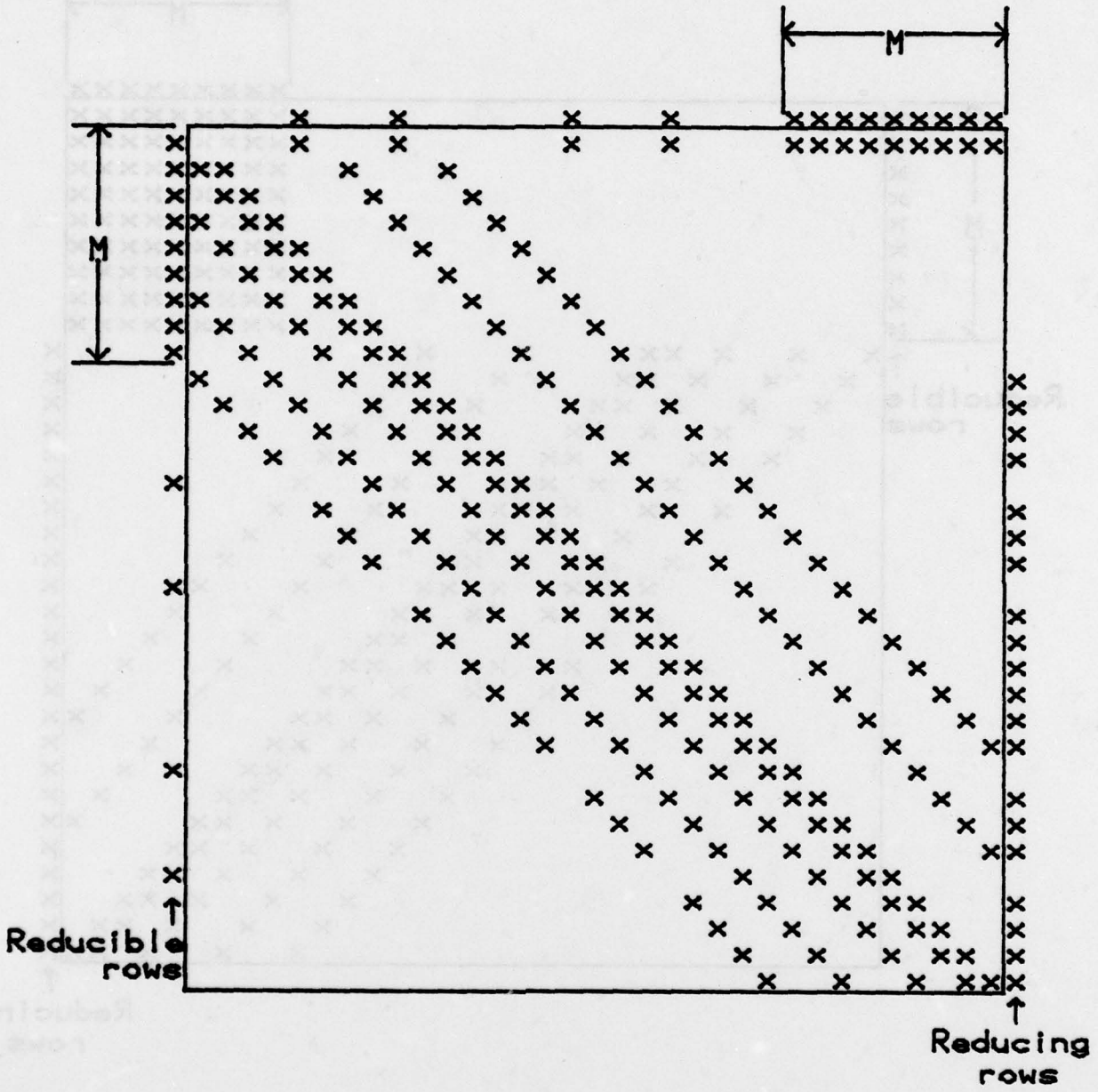


Figure 7. Matrix After First Row is Reduced

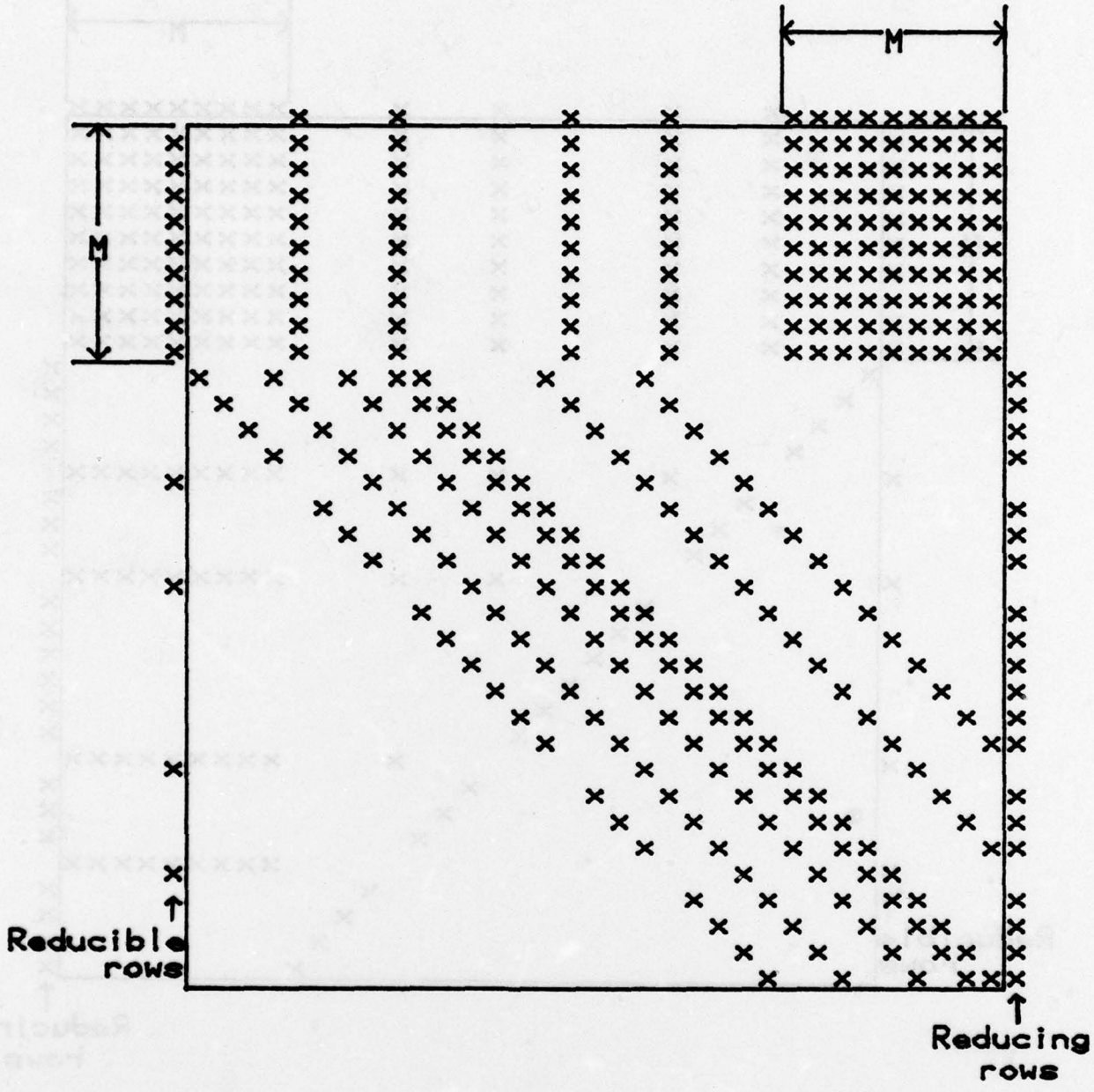


Figure 8. Completely Reduced Matrix

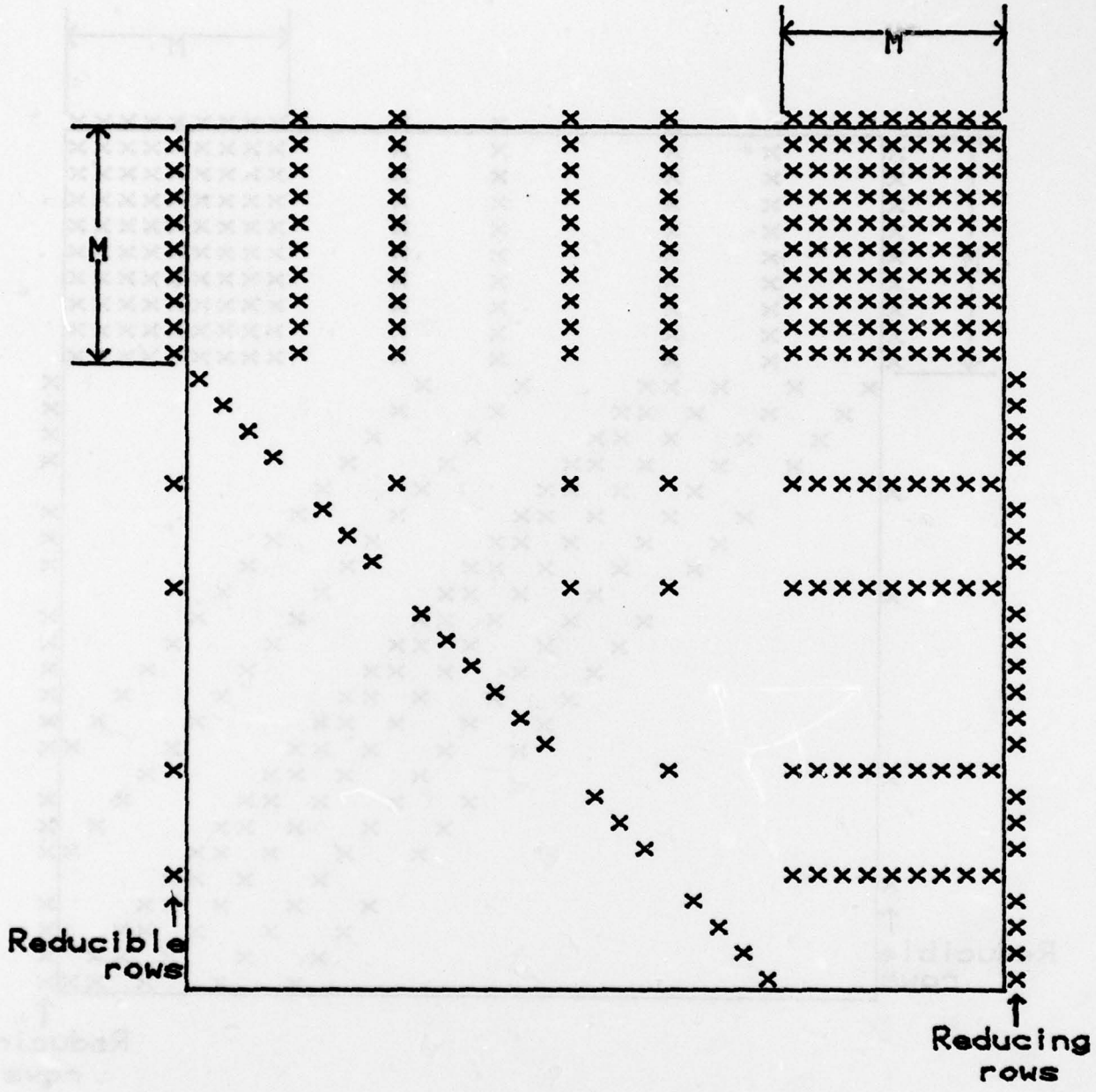


Figure 9. Reducible Rows Added During Reduction

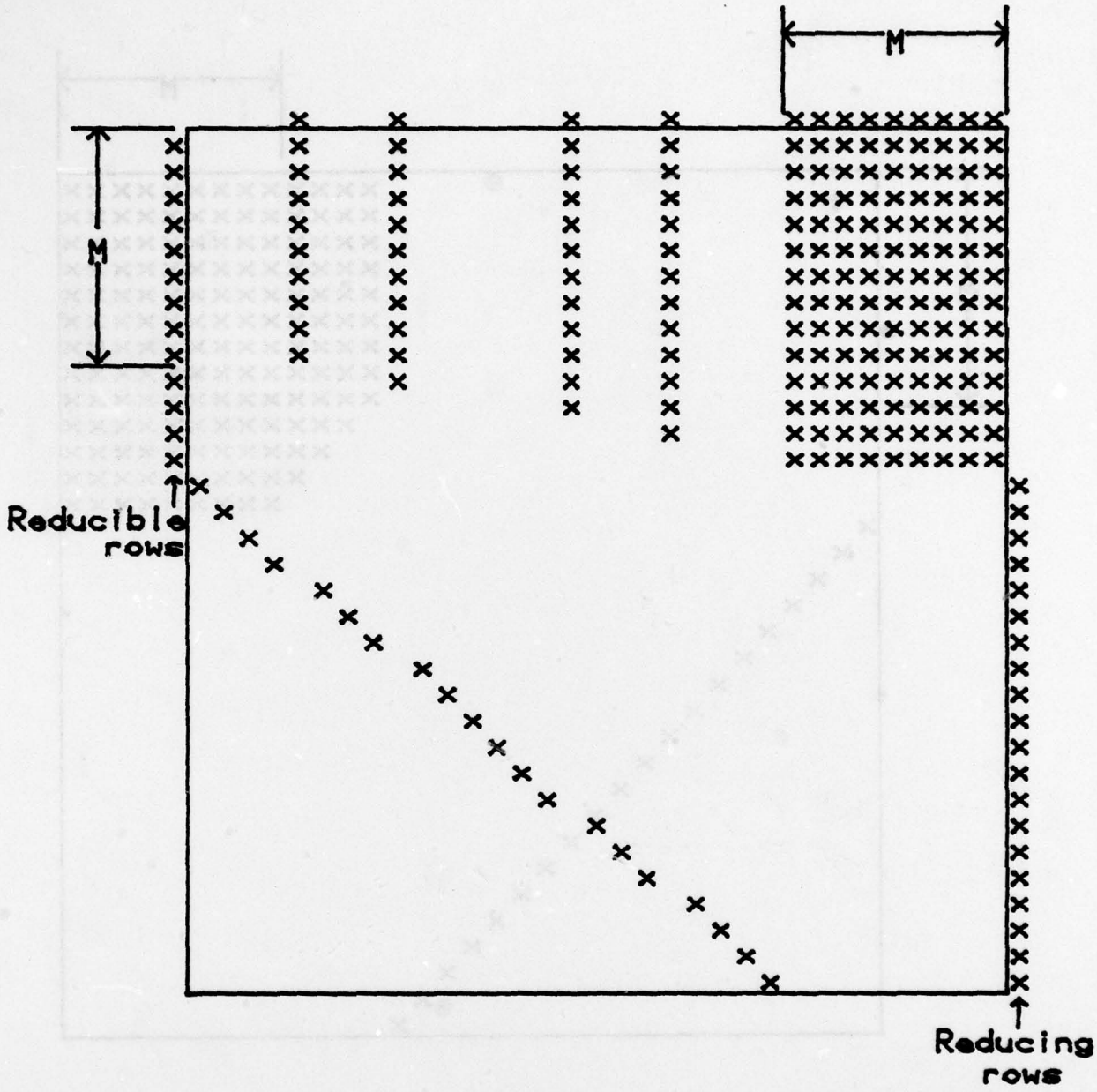


Figure 10. Matrix After Exchanging Rows

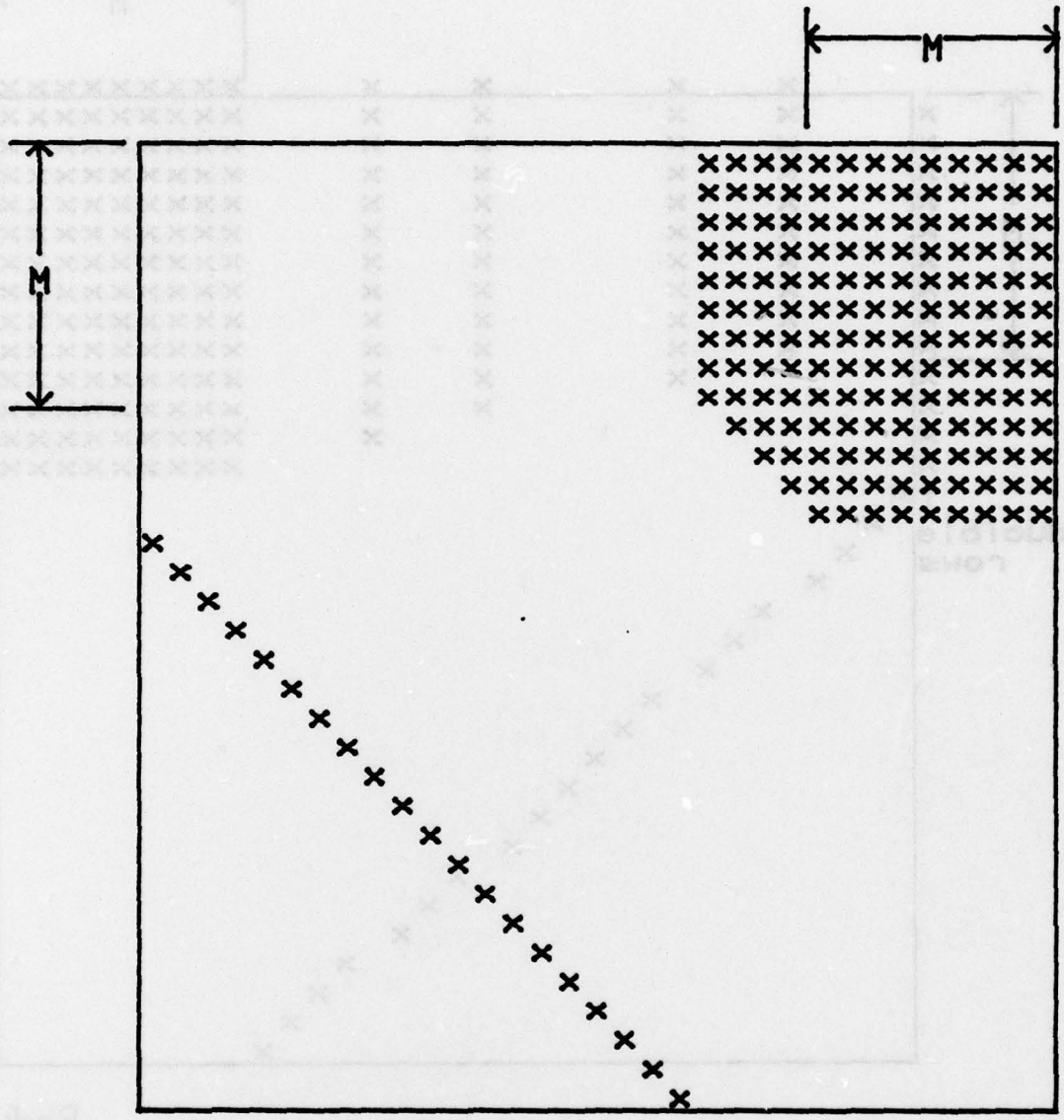


Figure 11. Matrix After Exchanging Columns