X-921-76-160
Preprint

NASA TM X- 71181

# RECURSIVE PARTITIONED INVERSION OF LARGE (1500 x 1500) SYMMETRIC MATRICES

BARBARA H. PUTNEY
JOSEPH E. BROWND
RICHARD A. GOMEZ

JULY 1976

GSFC
GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

X-921-76-160
Preprint

# RECURSIVE PARTITIONED INVERSION OF LARGE (1500 X 1500) SYMMETRIC MATRICES

Barbara H. Putney
Joseph E. Brownd
Richard A. Gomez

July 1976

GODDARD SPACE FLIGHT CENTER
Greenbelt, Maryland

# RECURSIVE PARTITIONED INVERSION OF LARGE (1500 × 1500) SYMMETRIC MATRICES

Barbara H. Putney
Goddard Space Flight Center

## ABSTRACT

A recursive algorithm was designed to invert large, dense, symmetric, positive-definite matrices using small amounts of computer core, i.e., a small fraction of the core needed to store the complete matrix. The algorithm described in this report is a generalized Gaussian elimination technique. Other algorithms are also discussed for the Cholesky decomposition and step-inversion techniques.

The purpose of the inversion algorithm is to solve large linear systems of normal equations generated by working geodetic problems. The algorithm was incorporated into a computer program called SOLVE (Reference 1 and 2). In the past the SOLVE program has been used in obtaining solutions published as the Goddard Earth Models (References 3 through 6). The latest of the Goddard Earth Model (GEM) solutions, GEM 8, contained approximately 1156 modeled variables.

## TABLE OF CONTENTS

## LIST OF ILLUSTRATIONS

# RECURSIVE PARTITIONED INVERSION OF LARGE $(1500 \times 1500)$ SYMMETRIC MATRICES

## SECTION 1 - INTRODUCTION

This report is concerned with the solution of linear systems of least-squares normal equations. The problem, described in matrix notation as

$$Ax = y$$

is to solve the vector, $x$, given the vector $y$ and a normal matrix, $A$, (positive-definite and symmetric).

The matrix inversion algorithm described in this report was designed for inverting large matrices using a fraction of the computer storage space (core) needed to store the complete matrix. The algorithm was designed to partition the matrices into small segments which fit into the available computer core.

Section 2 describes a generalized form of Gaussian elimination. This algorithm is presently used in the SOLVE 2 (Reference 1) program at Goddard Space Flight Center (GSFC) and is being programmed into several other geodetic computer programs as well. The algorithm is programmed in a package which includes a subroutine, PINV, which is the main driver. The PINV package is presented in the Appendix.

Section 3 describes two alternative algorithms using the same basic partitioning technique described in Section 1. These algorithms are Cholesky decomposition and a step-inversion method.

Section 4 deals with numerical problems in obtaining a matrix inverse and presents a simple method of obtaining a set of "condition numbers" which indicate numerical loss of precision for each row of the inverse matrix. Also described is a simple algorithm to iterate a solution vector in order to minimize numerical truncation error.

Test results using the PINV inversion package are presented in Section 5.

## SECTION 2 - A GENERALIZED GAUSSIAN ELIMINATION ALGORITHM

This section presents an algorithm for inverting symmetric, positive-definite matrices using a generalized form of Gaussian elimination. Normally in the elimination process, one row is eliminated at a time. The generalized algorithm allows for a set of rows, called a segment, to be eliminated each time. The size of the segment is selected to fit into available computer core, while the remainder of the matrix is stored on external storage devices. The algorithm presented in this section is coded in FORTRAN and listed in the Appendix.

The equation

$$Bx = y \tag{2-1}$$

can readily be partitioned to take the form

$$\begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \tag{2-2}$$

From the symmetry properties of B ,

$$B_{11} = B_{11}^T \tag{2-3}$$

$$B_{22} = B_{22}^T \tag{2-4}$$

$$B_{21} = B_{12}^T \tag{2-5}$$

The size of $B_{11}$ can be selected so that $B_{11}$ and $B_{12}$ contain no more elements than the computer storage will allow. This usually means that $B_{22}$ will not fit into the same storage area and, therefore, will need to be recursively partitioned in the same manner as B .

If B is nonsingular, a matrix, V , exists such that

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \tag{2-6}$$

and

$$\begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} = \begin{pmatrix} I & O \\ O & I \end{pmatrix} \tag{2-7}$$

In order to compute the submatrices of V , intermediate matrix sets are defined by

$$C_{11} = B_{11}^{-1} \tag{2-8}$$

$$C_{12} = C_{11} B_{12} \tag{2-9}$$

$$C_{21} = C_{12}^{T} \tag{2-10}$$

$$C_{22} = B_{22} - C_{21} B_{12} \tag{2-11}$$

The solutions for the submatrices of V are then

$$V_{22} = C_{22}^{-1} \tag{2-12}$$

$$V_{12} = -C_{12} V_{22} \tag{2-13}$$

$$V_{21} = V_{12}^{T} \tag{2-14}$$

$$V_{11} = C_{11} - V_{12} C_{21} \tag{2-15}$$

Solutions for the vector x are

$$x_2 = C_{22}^{-1} \left[ y_2 - C_{21} y_1 \right] \tag{2-16}$$

$$x_1 = C_{11} y_1 - C_{12} x_2 \tag{2-17}$$

A more direct method of computing the solution vector x would be to use the entire inverse matrix, i.e.,

$$x = Vy \tag{2-18}$$

However, the advantage of using the generalized Gaussian method (Equations (2-16) and (2-17)) is that the solution can be obtained without the complete inverse matrix.

Given below is a recursive algorithm for this method in which the matrix B is stored on an external scratch device and submatrices read into the internal core storage area as needed. Only the upper triangular portion of B elements (on and above the diagonals) need be stored because of the symmetric property of B . Three scratch units, denoted as units 1, 2, and 3, are used in the algorithm. The steps in the algorithm are as follows:

1. Read as many rows of B on unit 1 as possible to fill the available core storage area. The submatrices $B_{11}$ and $B_{12}$ are formed from these rows. $B_{21}$ is also initiated into core by using the symmetry property $B_{21} = B_{12}^T$ (see step 2 below).

2. Compute

$$C_{11} = B_{11}^{-1}$$

$$C_{12} = C_{11} B_{12}$$

$$y_2' = y_2 - C_{21} y_1$$

and

$$y_1' = C_{11} y_1$$

simultaneously replacing $B_{11}$ , $B_{12}$ , $y_1$ , $y_2$ in core with the newly computed variables, $C_{11}$ , $C_{12}$ , $y_1'$ , $y_2'$ .

3. Store the $C_{11}$ and $C_{12}$ submatrices on a scratch storage device (i.e., unit 3).

4. Read an additional row of the matrix B from scratch unit 1 which belongs to the submatrix $B_{22}$ on unit 1. Compute one row of the

$C_{22}$ submatrix (Equation (2-11)) and write the row on unit 2. For the ith row of $C_{22}$, the equation for the jth element is

$$C_{22}(i,j) = B_{22}(i,j) - \sum_k C_{21}(i,k) \cdot B_{12}(k,j)$$

Repeat step 4 until all rows of the $B_{22}$ matrix have been processed.

5. Rewind scratch units 1 and 2. The identification numbers for the two scratch units are switched so that the $C_{22}$ matrix can be recursively partitioned in the same manner (i.e., steps 1 through 4 are repeated) until the matrix written on scratch unit 2 is small enough to fit into the available core area. (Figure 2-1 illustrates the information on the external scratch units for steps 1 through 5.)

6. Read $C_{22}$ from unit 2 and store in core. Compute $V_{22} = C_{22}^{-1}$ and write the results back on unit 1. Rewind unit 2. Compute $x_2 = V_{22}\, y_2'$.

7. Read the last $C_{11}$, $C_{12}$ submatrix sets written on unit 3 and store in core. Read through unit 1; operating with one row of $V_{22}$ at a time, compute $V_{12} = -C_{12}\, V_{22}$.

8. Compute $V_{11}$, where

$$V_{11} = C_{11} - V_{12}\, C_{21}$$

and add the $V_{11}$ and $V_{12}$ submatrices with the $V_{22}$ matrix on unit 1. Rewind unit 1. If the complete matrix B has not been inverted, return to step 7. (Figure 2-2 illustrates the data stored on the scratch units during steps 6 through 8.)

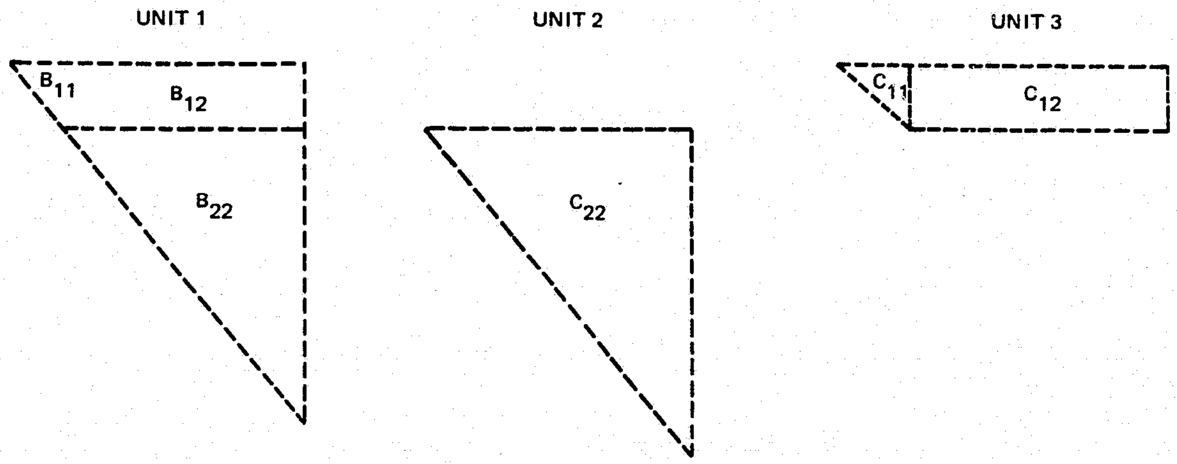


Figure 2-1. Information Stored on Scratch Units in Steps 1 Through 5 for the Recursive Gaussian Method

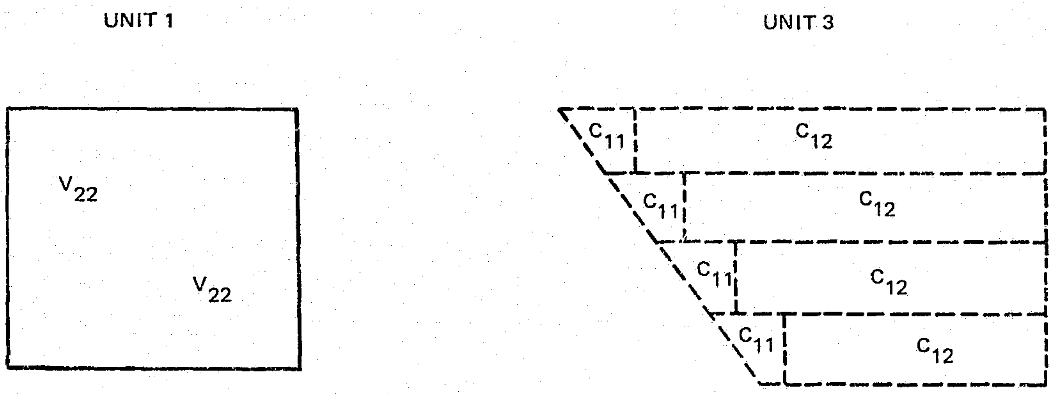


Figure 2-2. Information Stored on Scratch Units in Steps 6 Through 8 for the Recursive Gaussian Method

## SECTION 3 - CONSIDERATION OF OTHER INVERSION METHODS

The basic idea of segmenting the rows of the matrix to be inverted can be applied to other matrix inversion methods. This section describes two alternate inversion algorithms.

### 3.1 CHOLESKY DECOMPOSITION BY PARTITIONING

Equation (3-1) can be reformed by assuming the existence of a matrix, C , such that

$$C^T C = B \tag{3-1}$$

and

$$C^T C\, x = y \tag{3-2}$$

where the matrix C has elements whose values are zero below the diagonal. Matrix C is the decomposition of B . The vector w is then defined such that

$$w = Cx \tag{3-3}$$

and, therefore,

$$C^T w = y \tag{3-4}$$

Since the elements of $C^T$ are zero above the diagonal, w can be solved using forward substitution. Once w is obtained, x can be solved using back substitution.

Since C has values of zero below the diagonal,

$$c_{11} c_{1j} = b_{j1} \tag{3-5}$$

B is symmetric; therefore,

$$b_{j1} = b_{1j} \tag{3-6}$$

$$c_{11} c_{1j} = b_{1j} \tag{3-7}$$

and

$$c_{11}^2 = b_{11} \tag{3-8}$$

Therefore, in order to decompose the first row of B , the elements of C are

$$c_{1j} = \frac{b_{1j}}{\sqrt{b_{11}}} \tag{3-9}$$

The matrices C and B can be partitioned so that

$$\begin{pmatrix} C_{11}^T & O \\ C_{12}^T & C_{22}^T \end{pmatrix} \begin{pmatrix} C_{11} & C_{12} \\ O & C_{22} \end{pmatrix} = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \tag{3-10}$$

From this equation,

$$C_{12}^T C_{12} + C_{22}^T C_{22} = B_{22} \tag{3-11}$$

By partitioning so that $C_{11}$ and $C_{12}$ are the first row of the matrix $C$, and by rearranging Equation (3-11) so that

$$C_{22}^T C_{22} = B_{22} - C_{12}^T C_{12} \tag{3-12}$$

$C_{22}$ can be computed by decomposing the new matrix $B'$, where

$$B' = B_{22} - C_{12}^T C_{12} \tag{3-13}$$

Using the above recursive technique, $C$ can be computed one row at a time. Each time a row of $C$ is computed, it can simply replace the same row of $B$ in the computer storage. If not enough storage is available for the complete matrix, then only those rows which will fit into the area need be operated on. Once those rows are decomposed, Equation (3-13) is used to obtain a reduced matrix $B'$, which can then be decomposed.

An inverse matrix can be obtained by setting $y$ in Equation (3-2) to be a column of the identity matrix and then solving for one row at a time. The Cholesky method is best used for obtaining solutions which do not require inverse matrices.

## 3.2 RECURSIVE STEP METHOD

The recursive step method uses the same formulas as the generalized Gaussian inversion process but differs in the manner in which the formulas are employed. This method is designed to work more effectively using the lower triangular matrix, whereas the Gaussian elimination algorithm uses the upper triangle. The inverse of one submatrix is generated on a scratch unit and the other submatrices stored in core to be used to complete the inverse. The procedure is recursive in that the complete inverse of a smaller matrix is the inverse of the submatrix of a larger matrix. The procedure is repeated in order to obtain the complete inverse of the larger matrix. The steps in the algorithm are as follows:

1. Read from a scratch unit 1 as many rows of the matrix $B$ that can be stored into the available machine core. This unit will then contain the lower triangular elements of the matrix $B$, designated submatrix $B_{11}$.

2. Invert the submatrix $B_{11}$ and write the resultant inverse $\left(C_{11} = B_{11}^{-1}\right)$ on scratch unit 2. Rewind unit 2.

3. Read from scratch unit 1 as many additional rows as possible and form the submatrices $B_{21}$ and $B_{22}$ from these rows.

4. Read one row at a time of $C_{11}$ from unit B and compute

$$C_{12} = C_{11} B_{12}$$

   inserting the $C_{12}$ matrix in the computer core. Rewind unit 2.

5. Compute

$$C_{22} = B_{22} - B_{21} C_{12}$$

$$V_{22} = C_{22}^{-1}$$

$$V_{21} = -V_{22} C_{21}$$

These submatrices can be computed completely in the computer core area by replacing the area used for $B_{22}$ by $C_{22}$ and finally by $V_{22}$ ; the $B_{21}$ submatrix is then replaced by $V_{21}$ .

6. Read through unit 2 again one row at a time; compute

$$V_{11} = C_{11} - C_{12} V_{21}$$

and write the results on scratch unit 3. Rewind unit 2.

7. Write the submatrices $V_{21}$ , $V_{22}$ on unit 3 so that the complete inverse is on unit 3. Rewind unit 3. If the inverse of the complete matrix B has not been computed, switch units 2 and 3 so that $C_{11}$ is now the matrix V ; return to step 3. (Figure 3-1 illustrates data on the scratch units as they are used in the above steps.)

8. Compute the solution vector by reading the inverse matrix V on unit 3. Solve $x = Vy$ .

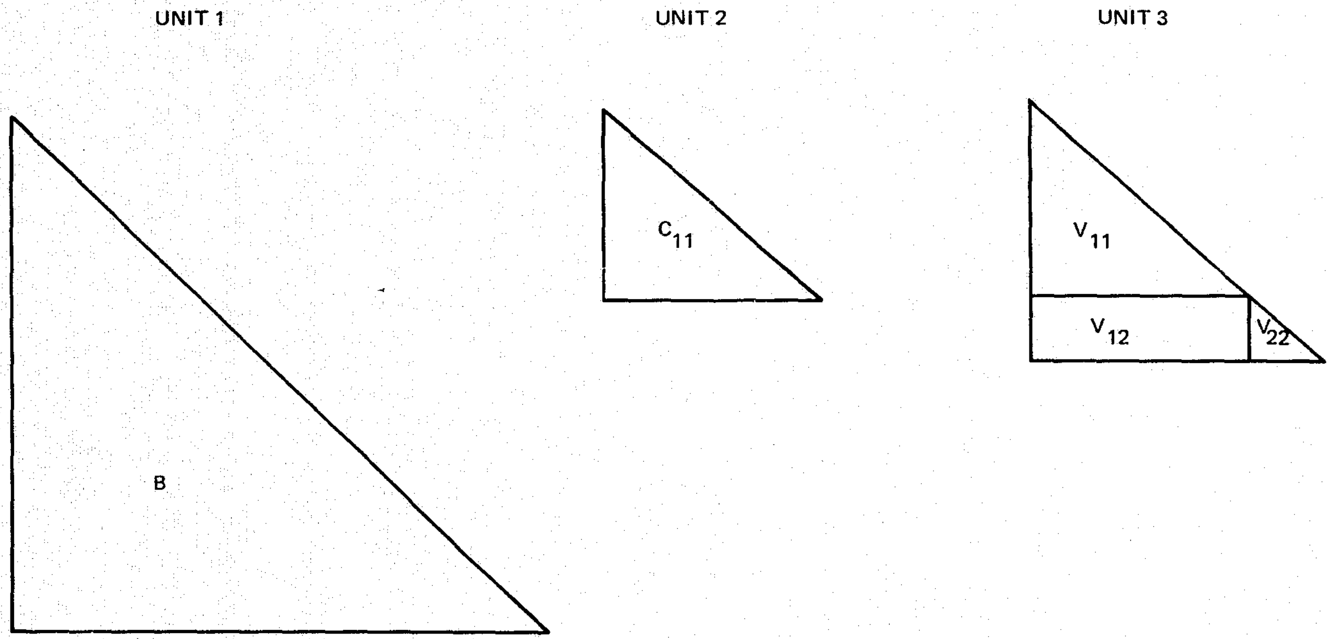


Figure 3-1. Information Contained on Scratch Units Used in the Recursive Step Method

## SECTION 4 - ERROR ANALYSIS

The source of error considered in this section is the numerical round-off error which occurs in floating point arithmetic when obtaining the solution to the linear system

$$Bx = y \tag{4-1}$$

If the assumption is made that the inverse matrix V can be obtained with the same numerical accuracy as the original matrix B, then the round-off error would occur in the matrix product

$$x = Vy \tag{4-2}$$

The numerical round-off error should occur primarily in the process of subtraction; i.e., if two six-digit numbers, a and b, are subtracted and yield a four-digit difference, then the answer would be good to two less decimal places. For a series summation, such as

$$x_i = \sum_j v_{ij}\, y_j \tag{4-3}$$

a condition number, $C_i$, is defined as

$$C_i = \frac{E\left(v_{ij}\, y_j\right)_{max}}{E\,(x_i)} \tag{4-4}$$

where $E\,(v_{ij}\, y_j)_{max}$ is the expected value of the largest term in the summation, and $E\,(x_i)$ is the expected value of the solution. The logarithm of the condition number should then yield the number of decimal places lost in the matrix product.

There is a round-off error, however, that is accumulated when the V matrix is computed. Therefore, the condition number noted above indicates the best possible answer, i.e., when V is accurate to as many decimal places as the original matrix B .

In order to obtain the condition number, $C_i$, the expected value of a random element of a vector is defined as the square root of the variance, i.e.,

$$E\,(x_i) = \left[\frac{\sum_{i=1}^{n} X_j}{n}\right]^{1/2} \qquad (4\text{-}5)$$

The above definition is useful only if it is reasonable to expect any one element of a vector to be the same magnitude as another. We can define such a vector as being "normaly distributed".

Given a vector X and a normaly distributed vector y the maximum expectation of the vector product, as used in equation (4-4), would be

$$E\,(X_i\,Y_i\,)_{max} = E\,(X_i\,)_{max} \cdot E\,(Y_i\,) \qquad (4\text{-}6)$$

The above equation cannot be applied directly to equation 4-4 because the vector is not, in general least squares problems, normaly distributed.

The expected magnitude of the sum of elements of a vector can be defined as

$$E\,(\sum_{i=1}^{n} X_i) = \left[\sum X_j^2\right]^{1/2} \qquad (4\text{-}7)$$

This definition is consistant with statistical error estimates of a sum of observations.

Equation 4-7 can then be used to define the expected magnitude of a vector product given the vector x and a normaly distributed vector y

$$E\left(\sum_{i=1}^{n} X_i Y_i\right) = E\left(\sum_{i=1}^{n} X_i\right) \cdot E(y_i) \qquad (4\text{-}8)$$

In order to evaluate the expected values in Equation (4-4), some insight into the least-squares process is needed. The matrix B is obtained from a set of observation equations in the matrix notation

$$Ax = r \qquad (4\text{-}9)$$

In the equation noted above, the vector r contains a set of weighted residuals whose expected values are

$$E(r_i) = 1 \text{ ,}$$

i.e., $\sigma_r = \pm 1$ for weighted observations and therefore r is considered normaly distributed. In the least-squares process,

$$B = A^T A \qquad (4\text{-}10)$$

and

$$y = A^T r \qquad (4\text{-}11)$$

From Equation (4-11), the ith element of the vector y is

$$y_i = \sum_j a_{ij} r_j \qquad (4\text{-}12)$$

and using definition 4-8 the expected value of $y_i$ is

$$E(y_i) = E\left(\sum_j a_{ij}\right) \cdot E(r_i) \tag{4-13}$$

using definition 4-7 it is evaluated as

$$E(y_i) = \left(\sum_j a_{ij}^2\right)^{1/2} \tag{4-14}$$

From Equation (4-10), the right hand term is

$$\left(\sum_j a_{ij}^2\right)^{1/2} = (b_{ii})^{1/2} \tag{4-15}$$

and Equation (4-14) becomes

$$E(y_i) = (b_{ii})^{1/2} \tag{4-16}$$

From Equation (4-16),

$$\frac{E(y_i)}{(b_{ii})^{1/2}} = \frac{E(y_j)}{(b_{jj})^{1/2}} \tag{4-17}$$

for all values of i and j. If the matrix D is defined such that

$$d_{ij} = \frac{b_{ij}}{(b_{ii})^{1/2}\,(b_{jj})^{1/2}} \tag{4-18}$$

then Equation (4-1) would become

$$D \begin{pmatrix} x_1 \, (b_{11})^{1/2} \\ \cdot \\ \cdot \\ \cdot \\ x_n \, (b_{nn})^{1/2} \end{pmatrix} = \begin{pmatrix} \dfrac{y_1}{(b_{11})^{1/2}} \\ \cdot \\ \cdot \\ \cdot \\ \dfrac{y_n}{(b_{nn})^{1/2}} \end{pmatrix} \qquad (4\text{-}19)$$

From the positive definite properties of the matrix B and using Equation (6-18),

$$\begin{aligned} |d_{ij}| &< 1 \text{ when } i \neq j \\ &= 1 \text{ when } i = j \end{aligned}$$

The maximum expected value of the ith row of D , $E\left(\sum_j d_{ij}\right)_{max}$ , is the diagonal term; the minimal expected value of $x_i$ is then

$$(b_{ii})^{1/2} \, E\,(x_i) = \frac{E\,(y_i)}{(b_{ii})^{1/2}}$$

or

$$E\,(x_i) = \frac{E\,(y_i)}{(b_{ii})} \qquad (4\text{-}20)$$

Using the definition 4-6 we can calculate the second expected value required in Equation (4-4), since $\left(\frac{y_i}{b_{ii}^{1/2}}\right)$ is normaly distributed.

$$E\left(v_{ij}\, y_j\right)_{max} = E\left(v_{ij} \cdot b_{jj}^{1/2}\right)_{max} \cdot E\left(\frac{y_i}{b_{ii}^{1/2}}\right)$$

$$E\left(v_{ij}\, y_j\right)_{max} = E\left(v_{ij} \cdot b_{ii}^{1/2} \cdot b_{jj}^{1/2}\right)_{max} \cdot \frac{E\,(y_i)}{b_{ii}}$$

The maximum terms of the $v_{ij} \cdot b_{ii}^{1/2}\, b_{jj}^{1/2}$ are when $i = j$ so that

$$E\left(v_{ij}\, y_j\right)_{max} = v_{ii}\, E\,(y_i) \tag{4-21}$$

The condition number is then derived using Equations (4-4), (4-20), and (4-21) as

$$C_i = b_{ii} \cdot v_{ii} \tag{4-22}$$

The equation noted above indicates that the loss of significant numbers of digits in the solution can be computed for each element in the solution vector using the product of the diagonal elements in the original and inverse matrices. The error analysis holds only when the solution vector has been iterated such that Equation (4-1) is maintained to the precision of the original matrices.

A method of ensuring the accuracy indicated by $C_i$ is to iterate the solution until the original Equation (4-1) is satisfied. Let

$$x_i = x_{oi} + \Delta x_i \tag{4-23}$$

where $x_{oi}$ is the intermediate solution, and $x_i$ is the complete solution which satisfies Equation (6-1). The vector $\Delta y$ can be computed such that

$$\Delta y = y - Bx_o \tag{4-24}$$

and $x_o$ is then recomputed so that

$$x_o = x_o + V\Delta y \tag{4-25}$$

After enough iterations $x_o$ approaches the solution vector $x$. Convergence occurs when the ratio

$$r_i = \left|\frac{\Delta x_i}{E(x_i)}\right| \tag{4-26}$$

is less than $10^{-d}$, where $d$ is the number of digits accuracy in the matrices $B$ and $y$.

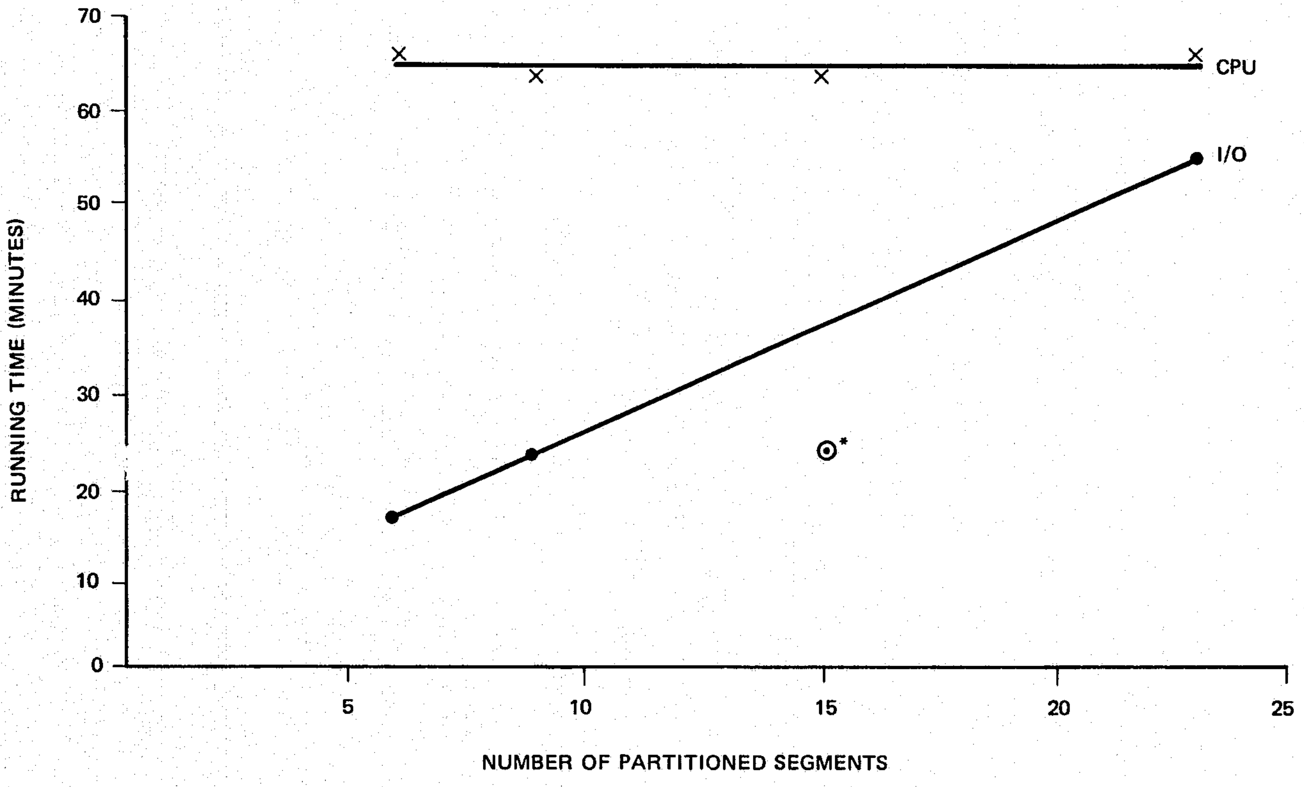
## SECTION 5 - RESULTS

Extensive tests were made of the generalized Gaussian elimination algorithm. The primary concerns were to determine optimum ratios of core requirements and computing time. There are many factors that can affect the computing time which are functions of the machine that is being used. However, there are some variables that can be extrapolated from one machine to another which are functions of the algorithm. These variables are discussed in this section.

The primary test was to invert a matrix dimensioned at (1500 x 1500) using five different partitioning levels. The test was controlled in that the solution vector was known a priori. The test showed that numerical stability is not affected by the number of partitioning levels. The results also showed that the central processing unit (CPU) time was only moderately affected by the number of levels of partitioning, whereas the input/output operating (I/O) time was linearly affected. Figure 5-1 illustrates the results as obtained on the IBM S/360-95 computer. Figure 5-2 illustrates the change in the I/O slope when different sized matrices are inverted.

In Figure 5-3, the I/O time is plotted versus the matrix dimension. Core is given in units of 1024, K bytes. The plots were obtained using the various partitioning levels. The plot illustrates the problems of inverting very large matrices, using small amounts of computer core. The I/O time in the plots is proportional to the dimension to the fourth power.

Figure 5-4 illustrates the CPU time required for inverting matrices of various dimensions. In this test the CPU time is porportional to the matrix dimension to the third power. Since the CPU time is only moderately affected by the number of partitioned segments (Figure 5-1), the CPU time can be easily established for the machine being used.



*THE 15-SEGMENT CASE USED A TAPE ON ONE OF THE SCRATCH UNITS WITH A LARGER BUFFER SIZE THAN THE DISK UNIT CASES (i.e., 32K VERSUS 7K).

Figure 5-1. Running Time Versus Number of Partitioned Segments for a Matrix of Dimension 1500 x 1500 (IBM S/360-95)

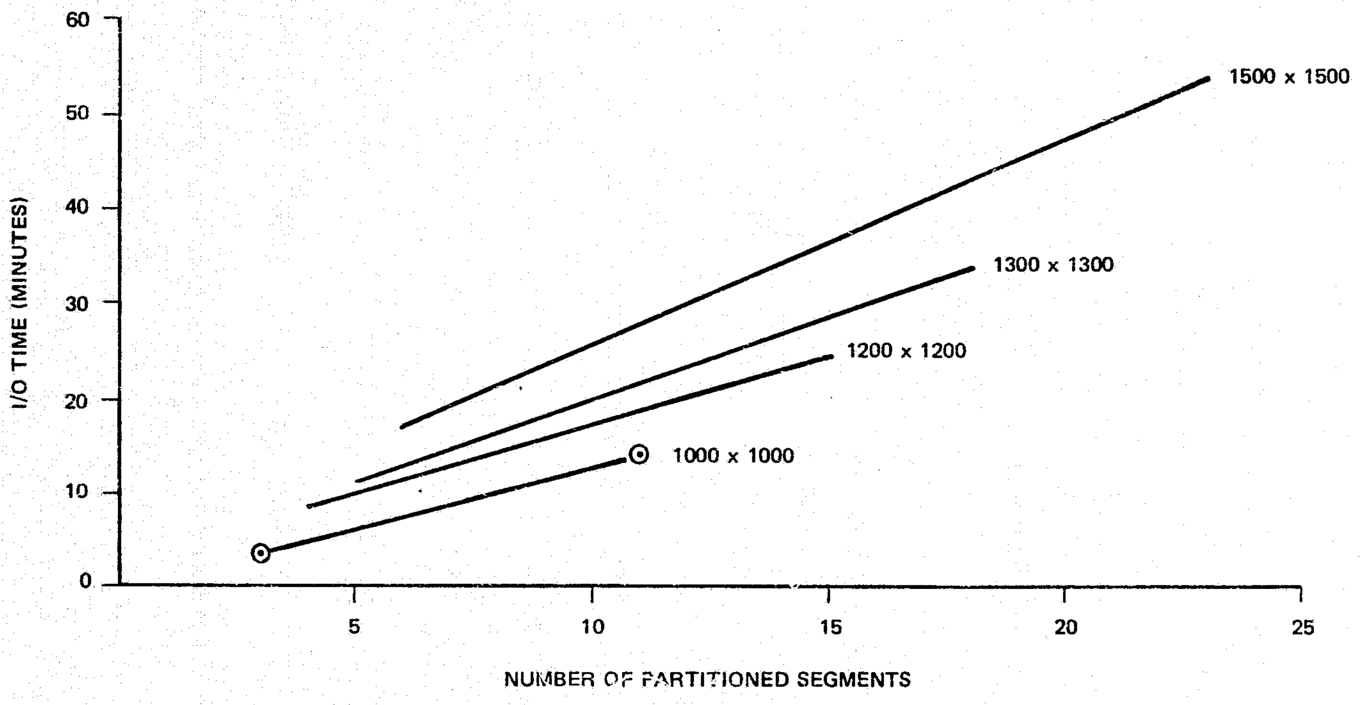
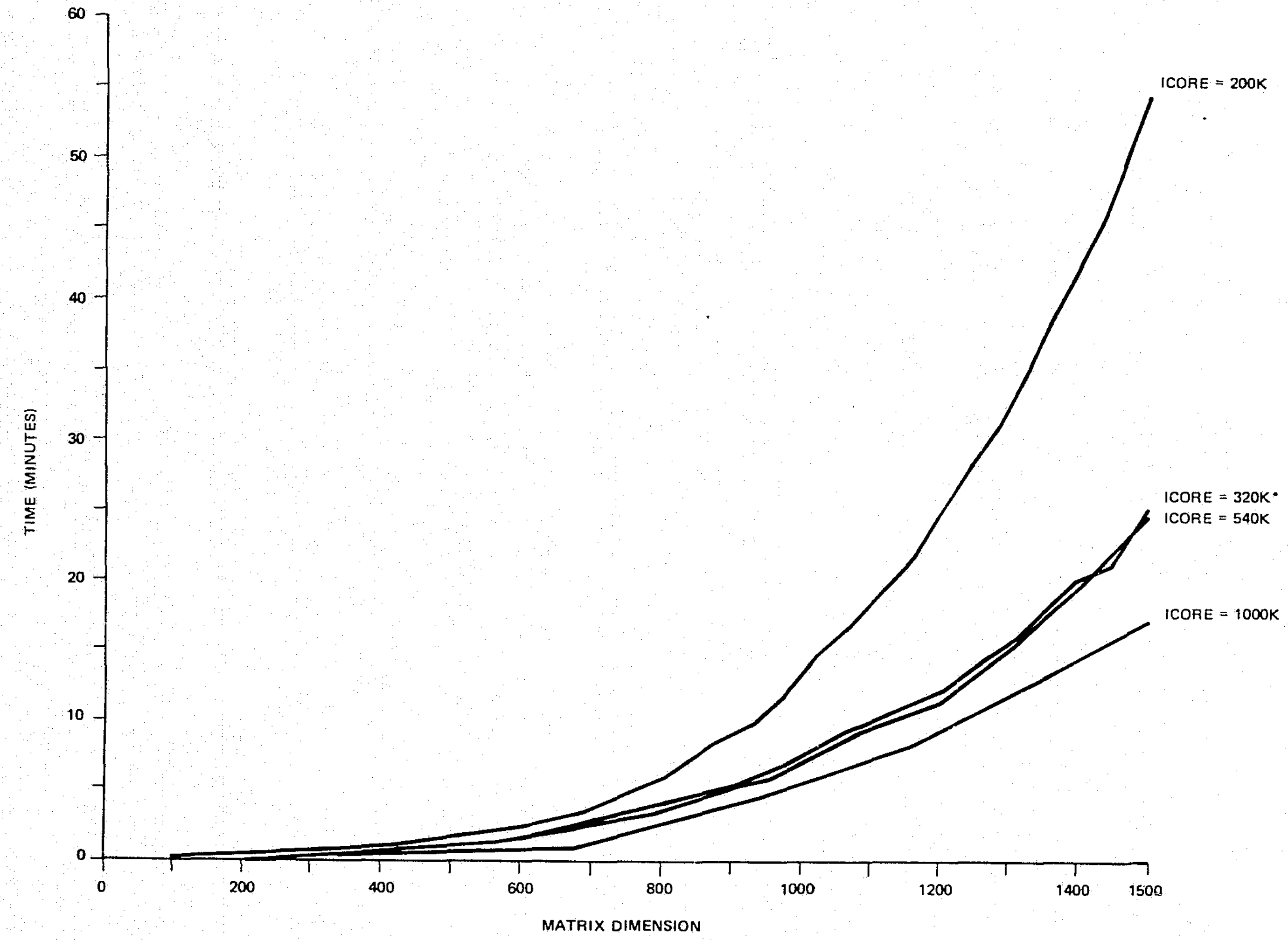


Figure 5-2. I/O Running Time Versus Number of Partitioned Segments for a Matrix of Various Dimensions (IBM S/360-95)



*THIS CASE USED A LARGER BUFFER ON ONE OF THE SCRATCH UNITS (i.e., 32K INSTEAD OF 7.4K).

Figure 5-3. I/O Time Required for Inverting Matrices Up to 1500 x 1500

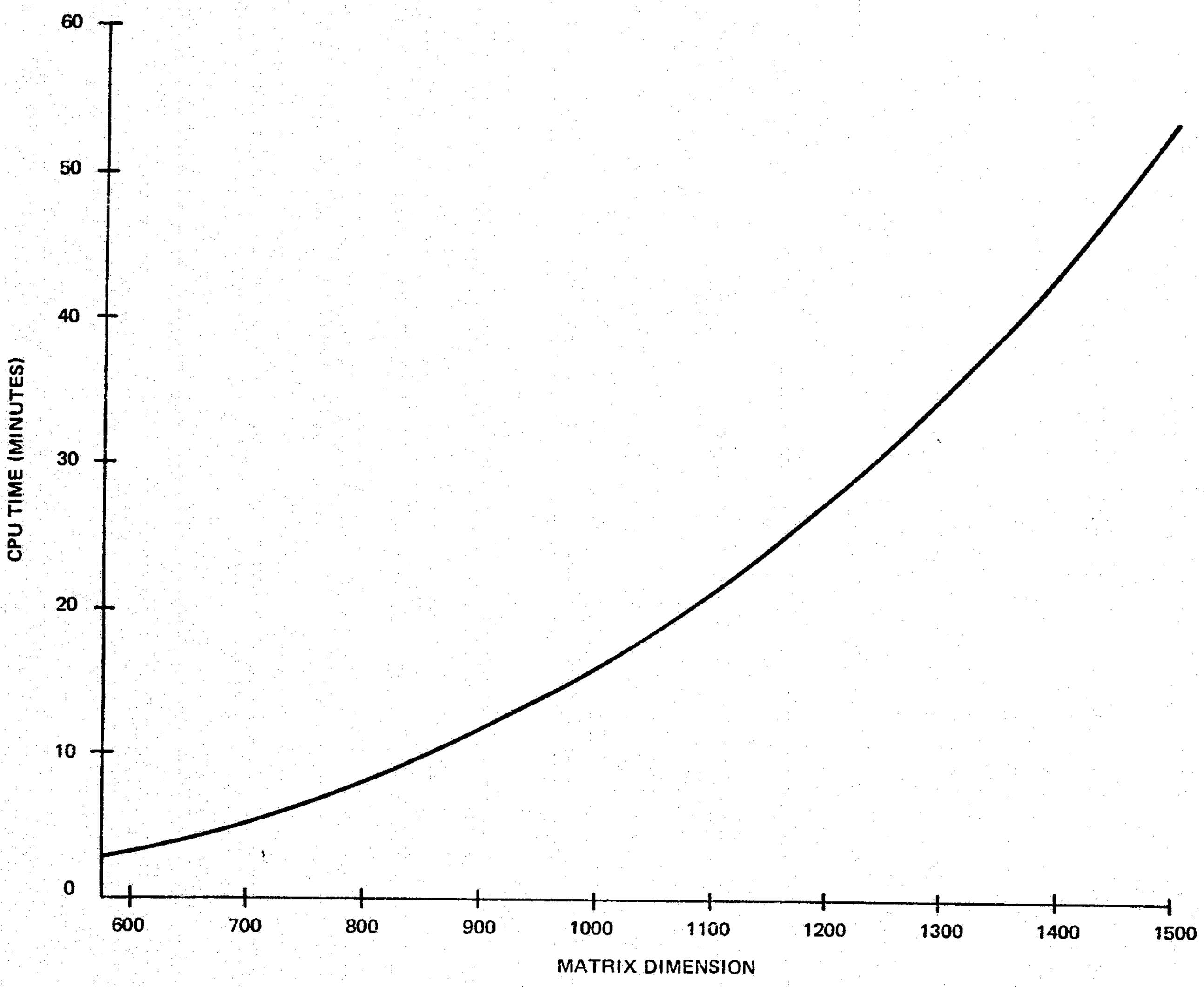


Figure 5-4. CPU Running Time in Minutes Using the Generalized Elimination Algorithm on the IBM S/360-95

Comparisons of running were made for computing the complete inverse and for stopping after the forward-elimination procedure and obtaining a solution. Table 5-1 gives results of a regression analysis of the resultant CPU and I/O curves. The parameters in the table are for inverting a matrix using 100K core. CPU time was fit to the curve

$$t_{CPU} = a_1 N + a_2 N^2 + a_3 N^3 \tag{5-1}$$

and the I/O time

$$t_{I/O} = a_1 N + a_2 N^2 + a_4 N^4 \tag{5-2}$$

The CPU comparisons show that the $a_3$ coefficient is reduced by approximately one-third when the solution is obtained without obtaining the complete inverse. Likewise, the $a_4$ coefficient is reduced by approximately one-fourth.

Table 5-1. Coefficients for Computing Running Time on the IBM S/360-95

| Coefficient | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
|---|---|---|---|---|
| CPU (complete inverse) | $2.02 \times 10^{-4}$ | $1.51 \times 10^{-7}$ | $1.58 \times 10^{-8}$ | - |
| CPU (solution vector only) | $4.76 \times 10^{-4}$ | $2.27 \times 10^{-7}$ | $0.52 \times 10^{-8}$ | - |
| I/O (complete inverse) | $-1.81 \times 10^{-4}$ | $3.85 \times 10^{-6}$ | - | $1.86 \times 10^{-11}$ |
| I/O (solution only) | $0.53 \times 10^{-4}$ | $3.36 \times 10^{-7}$ | - | $0.47 \times 10^{-11}$ |

## REFERENCES

1. Computer Sciences Corporation, CSC/TM-76/6095, SOLVE II Program Description and User's Guide, R. Gomez, J. Brownd, May 1976

2. --, 6006-00400-01TM, Methods of Successive Matrix Partitioning for Computing Inverses of Symmetric, Positive-Definite Matrices, J. Brownd, March 1975

3. National Aeronautics and Space Administration, Goddard Space Flight Center, TMX 65970, Gravitational Field Models for the Earth (GEM 1 and GEM 2), F. Lerch, C. Wagner, D. Smith, M. Sandson, J. Brownd, and J. Richardson, May 1972

4. --, TMX 66207, Gravitational Field Models GEM 3 and 4, F. Lerch C. Wagner, B. Putney, M. Sandson, J. Brownd, J. Richardson and W. Taylor, November 1972

5. --, X-921-74-145, Goddard Earth Models (5 and 6), F. J. Lerch, C. A. Wagner, J. A. Richardson, and J. E. Brownd, December 1974

6. --, X-921-76-20, Improvement in the Geopotential Derived From Satellite and Surface Data (GEM 7 and 8), C. A. Wagner, F. J. Lerch, J. C. Brownd and J. A. Richardson, January 1976

## APPENDIX

Presented in this appendix is a FORTRAN listing of the PINV package. PINV is the driver routine for the generalized Gaussian elimination algorithm.

```
      SUBROUTINE PINV(A,Y,X,B,L,M,ID,NN,N)
      REAL*8 Y(N),X(N)
      DIMENSION A(NN),B(1),ID(1),L(1),M(1)
C
C     THIS IS THE MAIN DRIVE SUBROUTINE USING  RECURSIVE PARTITIONING
C     AS A GENERALIZED GAUSSIAN ELEMINATION TO INVERT A LEAST SQUARES
C     MATRIX.
C     THE SOLUTION FOR X IS OBTAINED IN THE LINEAR SYSTEM  AX=Y .
C     ALSO THE INVERSE MATRIX IS OBTAINED AS A COMPLETE SQUARE MATRIX.
C
C     THE PROGRAM EXPECTS THE MATRIX A TO BE AVALABLE ON UNIT 40. EACH
C     RECORD CONTAINS 1 ROW OF A,STARTING WITH THE DIAGIONAL ELEMENT.
C
C     THE INVERSE MATRIX IS WRITTEN ON UNIT 41 IN A SQUARE FORMAT WHERE
C     EACH ROW IS CONTAINED ON 1 RECORD.
C
C     SCRATCH UNITS ARE  40,41,42,43,44  (I1 - I4 )
C
C     A ARRAY IS USED TO STORE PARTITIONED SEGMENTS OF THE MATRIX A
C     NN IS THE AMOUNT OF STORAGE AVALABLE
C     B ARRAY IS USED AS A TEMPORARY STORAGE , ID ARRAY ALSO
C     X - SOLUTION VECTOR , Y - RIGHT HAND SIDE
C     N - NUMBER OF ROWS IN MATRIX A
C
C
C     INITALIZE AND CLEAR ARRAYS
C
      DO 1 I=1,N
      B(I)=0.
    1 X(I)=Y(I)
      I1=40
      I2=41
      I3=43
      I4=44
C
C     COMPUTE NUMBER OF PARTITIONED SEGMENTS NEEDED (IP),
C     NUMBER OF COLUMS IN FIRST ROW (M) , AND NUMBER OF ROWS (L)
C     IN EACH SEGMENT
C
      M(1)=N
      CO=NN*2
      DO 2 I=1,N
      MI=M(I)*(M(I)+1)/2
      IF(MI.LE.NN)GO TO 21
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      BO=M(I)*2+1
      BO=BO/2
      L(I)=BO -SQRT(BO*BO-CO)
    2 M(I+1)=M(I)-L(I)
   21 IP=I
      PRINT 22,IP
   22 FORMAT(1H0,35HTHE MATRIX WILL BE PARTITIONED INTO,I3,1X,
     . 6HLEVELS)
         L(IP)=M(IP)
C
C     CHECK NUMBER OF PARTITIONED SEGMENTS , 1= SKIP PARTITIONING
C
      IF(IP.EQ.1) GO TO 4
C
C     GAUSSIAN ELIMINATION USING PARTITIONED SUBSETS
C
      IM1=IP-1
      DO 3 I=1,IM1
C
C     READ AND STORE A11,A12 PARTITIONED MATRICES
C
         LL=L(I)
         MM=M(I)
         CALL RDA1A2(A,MM,LL,NN,I1)
C
C     INVERT A11
C
         CALL SPGEI(LL,MM,A,B,NN)
C
C     COMPUTE  A22=A22- A21*A11*A12  AND  A12=A11*A12
C
         CALL FWDELM(A,B,B,LL,MM,NN,ID,I1,I2)
C
C     COMPUTE  X1=A11*Y1 , AND X2=X2-A21*X1
         NMM=N-MM+1
C
         CALL FWDSLV(A,B,X(NMM),LL,MM,NN)
C
C     WRITE DECOMPOSED MATRICES ON UNIT 43  (A11,A12)
C
         CALL WTA1A2(A,MM,LL,NN,I3)
C
    3 CONTINUE
C
```

```
C     READ LAST PARTITIONED SEGMENT
C
    4 MM=M(IP)
      CALL RDA1A2(A,MM,MM,NN,I1)
C
C     INVERT LAST PARTITIONED SEGMENT
C
      CALL SPGEI (MM,MM,A,B,NN)
C
C     COMPUTE X2=A22*Y2
C
      NMM = N - MM + 1
      CALL FWDSLV(A,B,X(NMM),MM,MM,NN)
C
C     WRITE A22 ON UNIT I1
C
      I1=I2
      I2=I2+1
      IF(I2.GT.42)I2=41
      CALL WRTB22(A,B,MM,NN,I1)
C
C     IF THERE IS ONLY 1 PARTITIONED SEGMENT NO MORE PROCESSING
C     IS REQUIRED.
C
      IF(IP.EQ.1)RETURN
C
C     OBTAIN SOLUTION THROUGH BACK SUBSTITUTION
C
      I=IP
      DO 5  J=1,IM1
         I=I-1
C
C     READ  A11,A12
C
         MM=M(I)
         LL=L(I)
         CALL RDA1A2(A,MM,LL,NN,I3)
C
C     SOLVE  X1=X1-A12*X2
C
         NMM=N-MM+1
         CALL BCKSLV(A,X(NMM),X(NMM),LL,MM,NN)
C
C     COMPUTE  B11=A11+A12*B22*A21
```

```
C                 B12=-A12*B22
C
           CALL BCKSUB(A,B,B,LL,MM,NN,ID,I1,I2,I4)
C
    5 CONTINUE
C
      RETURN
      END
      SUBROUTINE BCKSLV(A,X1,X2,L,M,N,NN)
      DIMENSION A(NN)
      REAL*8 X1(L),X2(M)
C
C     SOLVES  X1 = X1 - A12*X2
C
      IJ=0
      LP1=L+1
C
      DO 1 I=1,L
         IJ=IJ+LP1-I
      DO 1 J=LP1,M
         IJ=IJ+1
    1 X1(I)=X1(I)-A(IJ)*X2(J)
C
      RETURN
      END
      SUBROUTINE BCKSUB(A,B,D,L,M,NN,ID,I1,I2,I4)
C
C     THIS SUBROUTINE ENLARGENS THE PARTITIONED SEGMENT OF THE
C     INVERSE MATRIX THROUGH THE FORMULAS
C                        B11= A11 + A12*B22*A21
C                        B12= -A12*B22
C
      DIMENSION A(NN),B(L),D(M)
      INTEGER*2 ID(1)
C
C     A ARRAY CONTAINS A11 AND A12
C     B ARRAY IS A TEMPORARY ARRAY FOR B21 MATRIX
C     D ARRAY IS A TEMPORARY ARRAY FOR B22 MATRIX
C     FOR EFFICIENT CORE USAGE B AND D SHOULD BE EQUIVALENT
C
C     CLEAR B
C
      DO 1 I=1,L
    1 B(I)=0.
      MM=M-L
      LP1=L+1
C
C     INITALIZE INDEX ARRAY , INDEX(I,J) = ID(I)+J , J.GE.I
C
      ID(1)=0
      DO 2 I=2,MM
    2 ID(I)=ID(I-1)+M-I+1
C
C     READ B22 FROM UNIT I1
C     COMPUTE  B11= B11 + A12*B22*A21
C              B12= -A12*B22
C
      DO 5 K=LP1,M
      CALL READ6(D(LP1),MM,I1)
C
         DO 3 I=1,L
         DO 3 J=LP1,M
         IJ=ID(I)+J
    3    B(I)=B(I)-A(IJ)*D(J)
C
         DO 4 I=1,L
         IK=ID(I)+K
         DO 4 J=I,L
         IJ=ID(I)+J
    4   A(IJ)=A(IJ)-A(IK)*B(J)
      CALL WRITE6 (B,L,I4)
    5 CONTINUE
C
      REWIND I4
      REWIND I1
C
C     READ B21 AND STORE IN CORE
C
      DO 6 K=LP1,M
      CALL READ6(B,L,I4)
        DO 6 I=1,L
        IK=ID(I)+K
    6 A(IK)=B(I)
C
C     WRITE B11,B12 ON OUTPUT UNIT I2
C
      DO 10 I=1,L
      DO  9 J=1,M
      IF(J-I) 7,8,8
    7 IJ=ID(J)+I
      GO TO 9
    8 IJ=ID(I)+J
    9 D(J)=A(IJ)
C
      CALL WRITE6 (D,M,I2)
   10 CONTINUE
C
C     WRITE  B21,B22 ON UNIT I2
C
      DO 12 K=LP1,M
      CALL READ6(D(LP1),MM,I1)
        DO 11 I=1,L
        IK=ID(I)+K
   11   D(I)=A(IK)
      CALL WRITE6 (D,M,I2)
   12 CONTINUE
C
C     SWITCH UNITS AND REWIND
C
      REWIND I1
      REWIND I2
      I1=I2
      I2=I2+1
```

```
IF(I2.GT.42)I2=41
RETURN
END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE FWDELM(A,B,D,L,M,NN,ID,I1,I2)
C
C     THIS SUBROUTINE COMPUTES    A22 = A22 - A21*A11*A12
C                                 A12= A11*A12
C
      DIMENSION A(NN),B(L),D(M)
      INTEGER*2 ID(1)
C
C     A ARRAY CONTAINS A11 AND A12
C     B ARRAY IS A BUFFER FOR TEMPORARY STORAGE OF A12=A11*A12
C     D ARRAY IS A BUFFER FOR TEMPORARY STORAGE OF A22
C     FOR EFFICENT USE OF CORE B(1) AND D(1) SHOULD BE EQUIVALENT
C     ID IS AN INDEX ARRAY   INDEX(I,J)= ID(I)+ J  WHERE J.GE.I
C
C     INITALIZE B ARRAY
C
      DO 1 I=1,L
    1 B(I)=0.
C
C     INITALIZE ID ARRAY
C
      ID(1)=0
      MM=M-L
      DO 2 I=2,MM
    2 ID(I)=ID(I-1)+M-I+1
C
      LP1=L+1
      LM1=L-1
C
C     READ A22 FROM UNIT I1
C     WRITE  A22=A22 - A21*A11*A12 ON UNIT I2
C     STORE  A12= A11*A12 IN CORE
C
      DO 7 K=LP1,M
      CALL READ6 (D(K),MM,I1)
      DO 3  I=1,LM1
        IJ=ID(I)+I
        IK=ID(I)+K
        B(I)=B(I)+A(IJ)*A(IK)
        IP1=I+1
          DO 3 J=IP1,L
          IJ=ID(I)+J
          JK=ID(J)+K
          B(I)=B(I)+A(IJ)*A(JK)
    3     B(J)=B(J)+A(IJ)*A(IK)
        IJ=ID(L)+L
        IK=ID(L)+K
      B(L)=B(L)+A(IJ)*A(IK)
C
        DO 5 I=K,M
          DO 5 J=1,L
          IJ= ID(J)+I
    5   D(I)=D(I) - A(IJ)*B(J)
C
        DO 6 J= 1,L
        JK=ID(J)+K
        A(JK)=B(J)
    6   B(J)=0.
      CALL WRITE6(D(K),MM,I2)
    7 MM=MM-1
C
      REWIND I1
      REWIND I2
      I1=I2
      I2=I2+1
      IF(I2.GT.42)I2=41
      RETURN
      END
      SUBROUTINE FWDSLV(A,B,X,L,M,NN)
      DIMENSION A(NN)
      REAL*8 X(M),B(L)
C
C     SOLVE X1=A11*X1
C           X2=X2-A21*X1
C
      LP1=L+1
      IJ=0
C
      DO 4 I=1,L
          IJ=IJ+1
          B(I)=B(I)+A(IJ)*X(I)
          IP1=I+1
          IF(IP1.GT.L)GO TO 2
C
       DO 1 J=IP1,L
           IJ=IJ+1
           B(I)=B(I)+A(IJ)*X(J)
           B(J)=B(J)+A(IJ)*X(I)
    1  CONTINUE
C
    2     IF(LP1.GT.M)GO TO 4
C
       DO 3 J=LP1,M
           IJ=IJ+1
           X(J)=X(J)-A(IJ)*X(I)
    3  CONTINUE
C
           X(I)=B(I)
           B(I)=0.
    4 CONTINUE
      RETURN
      END
      SUBROUTINE RDA1A2(A,M,L,NN,I1)
C
C     READ A11 AND A12 INTO ARRAY A ,READ UNIT I1
C
      DIMENSION A(NN)
C
C     IF A IS READ ON UNIT 43 THEN BACKSPACE THROUGH L ROWS
C
      IF(I1.NE.43)GO TO 2
      DO 1 I=1,L
    1 BACKSPACE I1
C
C     READ A11,A12
C
    2 M1=1
      MM=M
      DO 3 I=1,L
      CALL READ6(A(M1),MM,I1)
      M1=M1+MM
    3 MM=MM-1
C
      IF(I1.NE.43)RETURN
      DO 4 I=1,L
    4 BACKSPACE I1
      RETURN
      END
      SUBROUTINE SPGEI(N,M,A,B,NN)
C
C     MATRIX INVERSION USING GAUSSIAN ELIMATION
C
C     A ARRAY IS UPPER TRIANGLE OF A POSITIVE DEFINITE,SYMETRIC MATRIX
C     B ARRAY IS FOR TEMPORARY STORAGE OF B12 PARTITIONED ROW
C
      DIMENSION A(NN),B(N)
C
C     INDEX FUNCTION.     INDEX(I,J) = IND(I) + J
C
      IND(I)=(M*2-I)*(I-1)/2
C
C     M - NUMBER OF ELEMENTS IN THE FIRST ROW OF A MATRIX
C     N - NUMBER OF ROWS IN A MATRIX TO BE INVERTED
C     NN- NUMBER OF ELEMENTS IN A MATRIX ,   NN= N*M -N*(N-1)/2
C
      NM1=N-1
C
C     FORWARD ELIMINATION, A11= 1/A11
C                          A22=A22 - A21*A11*A12
C                          A12=A11*A12
C
      DO 2 I=1,NM1
          IP1=I+1
          INDI=IND(I)
          II=INDI+I
          A(II)=1./A(II)
C
        DO 2 J=IP1,N
          INDJ=IND(J)
          IJ=INDI+J
C
         DO 1 K= J,N
          JK=INDJ+K
          IK=INDI+K
    1    A(JK)=A(JK)-A(IJ)*A(II)*A(IK)
C
    2   A(IJ)=A(II)*A(IJ)
C
      II=IND(N)+N
      A(II)=1./A(II)
C
C     BACK SUBSTITUTION,   B22=A22
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C                                         B12= -A12*B22
C                                         B11= A11 - B12*A21
C       CLEAR B ARRAY
C
        DO 4 I=1,N
      4 B(I)=0.
C
        I=N
        DO 8 I1=2,N
            J1=I
            I=I-1
            INDI=IND(I)
C
C         COMPUTE B12 = -A12*B22  FOR K.GE.J
C
         DO 5 J=J1,N
            INDJ=IND(J)
         DO 5 K=J,N
            IK=INDI+K
            JK=INDJ+K
      5  B(J)=B(J)-A(IK)*A(JK)
C
C       COMPUTE B12 FOR  K.LT.J
C
            IF(J1.EQ.N)GO TO 7
            J2=J1+1
C
         DO 6 J=J2,N
            K1=J-1
            JK=INDI+J
        K2=I1
         DO 6 K=J1,K1
        K2=K2-1
            IK=INDI+K
        JK=JK+K2
      6  B(J)=B(J) - A(IK)*A(JK)
C
      7     II=INDI + I
C
C        COMPUTE  B11 = B11 - A12*B21
C
         DO 8 J=J1,N
            IJ=INDI+J
            A(II)=A(II) - B(J)*A(IJ)
```

```
                A(IJ)=B(J)
    8   B(J)=0.
C
        RETURN
        END
```

```
      SUBROUTINE WTA1A2(A,M,L,NN,I1)
      DIMENSION A(NN)
C     WRITE A11,A12  ON SCRATCH UNIT I1 L ROWS ,M COLLUMNS
C
      MM=M
      M1=1
      DO 1 I=1,L
         CALL WRITE6(A(M1),MM,I1)
         M1=M1+MM
    1    MM=MM-1
      RETURN
      END
```