A COMPARISON OF ALGORITHMS FOR LEAST SQUARES ESTIMATES OF PARAMETERS IN THE LINEAR MODEL

by

CHUL H. AHN

B. S., Seoul National University, 1976

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Statistics

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1981

Approved by :

K. E. Kemp
Major Professor

TABLE OF CONTENTS

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

ACKNOWLEDGEMENTS

To Dr. K. E. Kemp I wish to express my gratitude for his guidance and advice in the preparation of this report. Gratitude is also expressed to Dr. D. E. Johnson and Dr. R. M. Rubison for their serving on the advisory committee and to Dr. G. A. Milliken for allowing me to use his computer program for generating a correlation matrix.

To my parents and wife I wish to express my appreciation. They have provided tremendous support.

I. Introduction

Most linear models applications require a solution for the normal equations : $X'Xb = X'Y$, which frequently requires an inverse of the $X'X$ matrix. This report compares three methods of finding the inverse of $X'X$ and the solution vector, $\underline{b}$: Gaussian elimination, Cholesky method, and Sweep operator. Relative efficiencies for each method are computed and are based on the number of arithmetic operations and computation time each method requires compared to others. The accuracy of the computed inverse, and hence the solution vector as well, is obtained empirically.

II. Gaussian Elimination

Gaussian elimination is the process of subtracting suitable multiples of each row of the system of equation from all succeeding rows to produce a matrix with zeros below and to the left of the main diagonal. The method first developed by Gauss early in the nineteenth century was used to solve systems of linear equations. But it can also be used to find the inverse of a matrix. From the normal equations : $X'Xb = X'Y$, let $X'X = A$ and $X'Y = C$. Then the problem is to find the inverse, A^{-1}, and the solution, b, in $Ab = C$. The following steps are required to find A^{-1} and b.

1. A is decomposed into two triangular matrices, L and U
2. L and U are inverted.
3. A^{-1} is formed by premultiplying L^{-1} by U^{-1}
4. b is obtained from $Ly = C$ and $Ub = y$.

Step 1 is known as triangular decomposition, the most commonly used variant of Gaussian elimination.

1. Triangular Decomposition

A matrix A is first decomposed into a lower triangular matrix with unit diagonal, L, and an upper triangular matrix, U, in the following manner :

$$A = LU \qquad (1)$$

The computation of L and U is illustrated by the following example. Consider the case of a 4 x 4 matrix. We have to find the elements l_{ij} and u_{ij} corresponding to known elements for which the following matrix equation holds.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{12} & a_{22} & a_{23} & a_{24} \\ a_{13} & a_{23} & a_{33} & a_{34} \\ a_{14} & a_{24} & a_{34} & a_{44} \end{bmatrix} \qquad (2)$$

From (2) we have 10 independent equations.

$$u_{11} = a_{11}$$

$$u_{12} = l_{21}u_{11} = a_{12}$$

$$u_{13} = l_{31}u_{11} = a_{13}$$

$$u_{14} = l_{41}u_{11} = a_{14} \qquad (3)$$

giving successively u_{11}, u_{12}, u_{13}, u_{14}, l_{21}, l_{31}, and l_{41} ;

$$l_{21}u_{12} + u_{22} = a_{22}$$

$$l_{21}u_{13} + u_{23} = l_{31}u_{12} + l_{32}u_{22} = a_{23}$$

$$l_{21}u_{14} + u_{24} = l_{41}u_{12} + l_{42}u_{22} = a_{24} \qquad (4)$$

giving successively u_{22}, u_{23}, u_{24}, l_{32}, and l_{42} ;

$$l_{31}u_{13} + l_{32}u_{23} + u_{33} = a_{33}$$

$$l_{31}u_{14} + l_{32}u_{24} + u_{34} = l_{41}u_{13} + l_{42}u_{23} + l_{43}u_{33} = a_{34} \qquad (5)$$

giving successively u_{33}, u_{34}, and l_{43} ;

$$l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + u_{44} = a_{44} \qquad (6)$$

giving u_{44}.

The basic steps in the triangular decomposition are

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj} \qquad \begin{matrix} i = 1, \ \ldots \ , \ j-1 \\ j = 1, \ \ldots \ , \ n \end{matrix} \qquad (7)$$

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}}{u_{jj}} \qquad \begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, i-1 \end{matrix} \qquad (8)$$

From (7) and (8), (i-1) multiplications and (i-1) additions are needed to compute u_{ij} and j multiplications and (j-1) additions to compute l_{ij}.

In the case of 4 x 4 matrix,

$$\bar{U} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 \end{bmatrix} \qquad \bar{L} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 2 & 3 & 0 \end{bmatrix}$$

where each element $\bar{u}_{ij}$ and $\bar{l}_{ij}$ denotes the number of multiplications required to compute its own.

From (7) and (8) the number of multiplications required to decompose A into L and U is given by:

$$\sum_{j=1}^{n} \sum_{i=1}^{j} (i-1) + \sum_{i=1}^{n} \sum_{j=1}^{i-1} j = \frac{n^3}{3} - \frac{n}{3} \qquad (9)$$

Similiarly the number of additions is given by:

$$\sum_{j=1}^{n} \sum_{i=1}^{j} (i-1) + \sum_{i=1}^{n} \sum_{j=1}^{i-1} (j-1) = \frac{n^3}{3} - \frac{n^2}{2} + \frac{n}{6} \qquad (10)$$

2. Matrix Inversion

If U^{-1} has the elements v_{ij}, then by definition of the inverse we have

$$\sum_{k=1}^{n} u_{ik}v_{kj} = \delta_{ij} = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{otherwise} \end{cases}$$

e.g. if n=4, we have to find the v_{ij} such that

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} \\ 0 & v_{22} & v_{23} & v_{24} \\ 0 & 0 & v_{33} & v_{34} \\ 0 & 0 & 0 & v_{44} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To calculate v_{ij} we proceed in the following manner.

First for i=1...n calculate $v_{ii} = 1/u_{ii}$. Then calculate v_{in} in the order i=n...1. The next step is to calculate $v_{i,n-1}$ in the order i=(n-1)...1,etc.

The elements of U^{-1} are, thus, given by

$$v_{ii} = \frac{1}{u_{ii}} \qquad (11)$$

$$v_{ij} = -v_{ii} \sum_{p=i+1}^{j} u_{ip}v_{pj} \quad , \; i \neq j \qquad (12)$$

The calculation of n reciprocals used in (11), is performed as part of the decomposition of A into L and U. To compute v_{ij}, $i \neq j$, in (12), requires (j-i+1) multiplications and (j-i-1) additions which gives the number of multiplications as

$$\sum_{j=1}^{n} \sum_{i=1}^{j-1} (j-i+1) = \frac{n(n-1) (n+4)}{6} \qquad (13)$$

Similiarly the number of additions is

$$\sum_{j=1}^{n} \sum_{i=1}^{j-1} (j-i-1) = \frac{n(n-1)(n-2)}{6} \qquad (14)$$

The same analysis is applicable to L^{-1} but as the diagonal elements of L are unity the number of multiplications is reduced by n(n-1). To invert L requires

$$\frac{n(n-1)(n-2)}{6} \quad \text{multiplications} \qquad (15)$$

$$\frac{n(n-1)(n-2)}{6} \quad \text{additions} \qquad (16)$$

3. The formation of $A^{-1} = U^{-1}L^{-1}$

The formation of $A^{-1} = U^{-1}L^{-1}$ is the reverse process of the triangular decomposition of A into L and U. Let b_{ij} denote the elements of A^{-1}, v_{ij} the elements of U^{-1} and m_{ij} the elements of L^{-1}. Then the basic steps in the formation of $A^{-1} = U^{-1}L^{-1}$ are

$$b_{ij} = v_{ij} + \sum_{k=1}^{i-1} v_{ik} m_{kj} \quad , \quad i \leq j \qquad (17)$$

$$b_{ij} = v_{jj} m_{jj} + \sum_{k=1}^{j-1} v_{ik} m_{kj} \quad , \quad i > j \qquad (18)$$

Hence, it may be shown that the formation of $A^{-1} = U^{-1}L^{-1}$ requires the number of multiplications given as

$$\sum_{j=1}^{n}\sum_{i=1}^{j}(i-1) + \sum_{i=1}^{n}\sum_{j=1}^{i-1} j = \frac{n^3}{3} - \frac{n}{3} \qquad (19)$$

and the number of additions given as

$$\sum_{j=1}^{n}\sum_{i=1}^{j}(i-1) + \sum_{i=1}^{n}\sum_{j=1}^{i-1}(j-1) = \frac{n^3}{3} - \frac{n^2}{2} + \frac{n}{6} \qquad (20)$$

4. Back Substitution

To complete the solution of A = C, the following steps are required :

(i) y is evaluated from Ly = C.

(ii) b is evaluated from Ub = y.

(i) can be written in the form :

$$\begin{aligned} l_{11}y_1 &= c_1 \\ l_{21}y_1 + l_{22}y_2 &= c_2 \\ &\cdots\cdots\cdots \\ l_{n1}y_1 + l_{n2}y_2 + \cdots + l_{nn}y_n &= c_n \end{aligned}$$

The variables $y_1, y_2, \ldots, y_n$ are determined in succession from the first, second, . . . , n th equations, respectively,

i.e. $y_1 = c_1 / l_{11}$

$y_2 = (c_2 - l_{22}y_1) / l_{22}$, etc.

For a general step the evaluation of y_k requires one reciprocal, k multiplications and k-1 additions. Hence to solve the system of equations Ly = C requires n(n+1)/2 multiplications and n(n-1)/2 additions. The solution of Ub = y requires the same amount of arithmetic.

However in our case all $l_{ii} = 1$ and so the number of multiplications required is reduced by n. The amount of arithmetic required in the back substitution phase is

$$n^2 \quad \text{multiplications}$$
$$n^2 - n \quad \text{additions.} \qquad (21)$$

As shown in the table 1, the total number of arithmetic operations to find the inverse and the solution is

$$n^3 + n^2 - n \quad \text{multiplications}$$
$$n^3 - n \quad \text{additions} \qquad (22)$$

Table 1. Number of computations required for solution vector and inverse matrix using Gaussian elimination

	Multiplications	Additions
Decompose A into L & U	$(n^3 - n)/3$	$(2n^3 - 3n^2 + n)/6$
Invert U	$n(n-1)(n+4)/6$	$n(n-1)(n-2)/6$
Invert L	$n(n-1)(n-2)/6$	$n(n-1)(n-2)/6$
Formation of $A^{-1} = U^{-1}L^{-1}$	$(n^3 - n)/3$	$(2n^3 - 3n^2 + n)/6$
Back Substitution	n^2	$n^2 - n$
Total	$n^3 + n^2 - n$	$n^3 - n^2$

III. Cholesky Method

The Cholesky method is used when the matrix A is symmetric and positive definite. It uses a decomposition which transforms the matrix A into the product of a lower triangular matrix, L, and the transpose of itself, L^T, i.e. $A = LL^T$. The proof of $A = LL^T$, due to Cholesky, is by induction and may be found in Fox(1954) or Wilkinson(1965).

A particularly useful property of Cholesky decomposition is that if the matrix is scaled originally so that all elements of A are bounded by unity then so, too, are all elements of L, and no pivoting is required. The following processes are given for finding the inverse, A^{-1}, and the solution, X, in Ab=C.

1. A is decomposed into L and L^T.
2. A^{-1} is obtained from $L^T A^{-1} = L^{-1}$.
3. b is obtained from Ly=C and $L^T b = y$.

1. Cholesky Decomposition

A symmetric and positive definite matrix A is transformed as $A = LL^T$. (1)

The computation of L is illustrated by the following example.

Consider the case of a 4 x 4 matrix. We have to find the elements l_{ij} corresponding to known elements a_{ij} for which the following matrix equation holds.

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} & l_{41} \\ 0 & l_{22} & l_{32} & l_{42} \\ 0 & 0 & l_{33} & l_{43} \\ 0 & 0 & 0 & l_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{12} & a_{22} & a_{23} & a_{24} \\ a_{13} & a_{23} & a_{33} & a_{34} \\ a_{14} & a_{24} & a_{34} & a_{44} \end{bmatrix} \quad (2)$$

From (2), we have 10 independent equations.

$$\begin{aligned} l_{11}^2 &= a_{11}, \\ l_{11}l_{21} &= l_{21}l_{11} = a_{12}, \\ l_{11}l_{31} &= l_{31}l_{11} = a_{13}, \\ l_{11}l_{41} &= l_{41}l_{11} = a_{14}, \end{aligned} \quad (3)$$

giving successively l_{11}, l_{21}, l_{31}, and l_{41} ;

$$l_{21}^2 + l_{22}^2 = a_{22},$$

$$l_{21}l_{31} + l_{22}l_{32} = l_{31}l_{21} + l_{32}l_{22} = a_{23},$$

$$l_{21}l_{41} + l_{22}l_{42} = l_{41}l_{21} + l_{42}l_{22} = a_{24}, \qquad (4)$$

giving successively l_{22}, l_{32}, and l_{42} ;

$$\begin{aligned} l_{31}^2 + l_{32}^2 + l_{33}^2 &= a_{33}, \\ l_{31}l_{41} + l_{32}l_{42} + l_{33}l_{43} &= l_{41}l_{31} + l_{42}l_{32} + l_{43}l_{33} = a_{34}, \end{aligned} \qquad (5)$$

giving successively l_{33} and l_{43} ;

$$l_{41}^2 + l_{42}^2 + l_{43}^2 + l_{44}^2 = a_{44}, \qquad (6)$$

giving l_{44}.

The basic steps in the Cholesky decomposition are

$$l_{ii} = \left(a_{ii} - \sum_{j=1}^{i-1} l_{ij}^2 \right)^{1/2}, \quad i = 1, \ldots, n \qquad (7)$$

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}}{l_{jj}}, \quad \begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, i-1 \end{matrix} \qquad (8)$$

The division in (8) is usually replaced by computing instead the reciprocal of the square root in (7) and then multiplying the numerator in (8) by the corresponding reciprocal. In fact, the computed reciprocals are stored in the diagonal positions of L and used directly at later steps of computation. From (7), and (8) (i-1) multiplications and (i-1) additions are needed to compute l_{ii} and j multiplications and (j-1) additions to compute l_{ij}. In the case of 4 x 4 matrix,

$$\bar{L} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 2 & 0 \\ 1 & 2 & 3 & 3 \end{bmatrix}$$

Where each element $\bar{l}_{ij}$ denotes the number of multiplications required to compute its own. From (7) and (8) the number of multiplications required to decompose A into L and L^T is given by :

$$\sum_{i=1}^{n} \left[(i-1) + \sum_{j=1}^{i-1} j \right] = \frac{n^3}{6} + \frac{n^2}{2} - \frac{2n}{3} \qquad (9)$$

Similiarly the number of additions is given by :

$$\sum_{i=1}^{n} \left[(i-1) + \sum_{j=1}^{i-1} (j-1) \right] = \frac{n^3}{6} - \frac{n}{6} \qquad (10)$$

In fact, the upper triangular matrix, L^T by Cholesky decomposition can also be obtained from the upper triangular matrix, U formed by the triangular decomposition. This can be done by dividing each row of U by the square root to the diagonal elements of U.

2. Matrix Inversion

For finding the inverse of a symmetric and positive definite matrix A we use the different method from that in the previous section. Here we take advantage of the way the matrix is decomposed.

$$\begin{aligned} A &= LL^T \\ A^{-1} &= (L^T)^{-1} L^{-1} \\ L^T A^{-1} &= L^{-1} \end{aligned} \qquad (11)$$

The (11) does not need the explicit inversion of L. Consider the case of a 3 x 3 matrix. We denote l_{ij} as the elements of L^T, m_{ij} as the elements of L^{-1}, and b_{ij} as the elements of A^{-1}.

Then (11) can be expressed as

$$\begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix} = \begin{bmatrix} m_{11} & 0 & 0 \\ m_{21} & m_{22} & 0 \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

Thus $b_{33} = m_{33} / l_{33}$

$$b_{23} = -l_{32} b_{33} / l_{22}$$

$$b_{22} = (m_{22} - l_{32} b_{23}) / l_{22}$$

$$b_{13} = -(l_{21} b_{13} + l_{31} b_{33}) / l_{11}$$

$$b_{12} = -(l_{21} b_{22} + l_{31} b_{23}) / l_{11}$$

$$b_{11} = (m_{11} - l_{21} b_{12} - l_{31} b_{13}) / l_{11} \qquad (12)$$

From (12) it can be seen that only the diagonal elements m_{11}, m_{22}, and m_{33} of L^{-1} are needed to compute A^{-1}. But these are just the reciprocals of l_{ii}, the diagonal elements of L.

The elements of A^{-1} are given by

$$b_{ii} = \frac{1}{l_{ii}} \left[\frac{1}{l_{ii}} - \sum_{k=i+1}^{n} l_{ki} b_{ik} \right] \qquad (13)$$

which requires (n-i+1) multiplications and (n-i) additions, and

$$b_{ij} = -\frac{1}{l_{ii}} \sum_{k=i+1}^{n} l_{ki} b_{kj} , \quad i < j , \qquad (14)$$

which requires (n-i+1) multiplications and (n-i-1) additions.

From (13) and (14) the number of multiplications is

$$\sum_{i=1}^{n} (n-i+1) + \sum_{j=1}^{n} \sum_{i=1}^{j-1} (n-i+1) = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6} \qquad (15)$$

and the number of addition is

$$\sum_{i=1}^{n} (n-i) + \sum_{j=1}^{n} \sum_{i=1}^{j-1} (n-i-1) = \frac{n^3}{3} - \frac{n^2}{2} + \frac{n}{6} \qquad (16)$$

3. Back Substitution

To complete the solution of AX = C, the following steps are required :

(i) y is obtained from Ly = C.

(ii) b is obtained from $L^T b = y$.

As shown in the back substitution of Gaussian elimination, to solve the systems of equations Ly = C requires n(n+1)/2 multiplications and n(n-1)/2 additions. The solution of $L^T b = y$ requires the same amount of arithmetic. The amount of arithmetic required in the back substitution phase is

$n^2 + n$ multiplications and

$n^2 - n$ additions (17)

As shown in the table 2, the total number of arithmetic operations to find the inverse and the solution is

$\frac{n^3}{2} + 2n^2 + \frac{n}{2}$ multiplications and

$\frac{n^3}{2} + \frac{n^2}{2} - n$ additions . (18)

Table 2. Number of computations required for solution vector and inverse matrix using Cholesky's method

	Multiplications	Additions
Decompose A into L & L^T	$(n^3+3n^2-4n)/6$	$(n^3 - n)/6$
A^{-1} from $L^T A^{-1} = L^{-1}$	$(2n^3+3n^2+n)/6$	$(2n^3-3n^2+n)/6$
Back substitution	n^2+n	n^2-n
Total	$(n^3+4n^2+n)/2$	$(n^3+n^2-2n)/2$

IV. Sweep Operator

The method first developed by Beaton in 1964 is a modification of the Adjust operator that reduces the amount of core needed to compute the b values, ESS, and $(X'X)^{-1}$. Whereas the Adjust operator performs the mapping,

$$\begin{bmatrix} X'X & X'Y & I \\ Y'X & Y'Y & 0 \end{bmatrix} \xrightarrow[X'X \text{ columns}]{\text{Adjust}} \begin{bmatrix} I & b & (X'X)^{-1} \\ 0 & ESS & -b' \end{bmatrix}, \qquad (1)$$

the Sweep operator performs the in-place mapping,

$$\begin{bmatrix} X'X & X'Y \\ Y'X & Y'Y \end{bmatrix} \xrightarrow[X'X \text{ columns}]{\text{Sweep}} \begin{bmatrix} (X'X)^{-1} & b \\ -b & ESS \end{bmatrix}. \qquad (2)$$

It is illustrated by the following example. Suppose we have data

X_0	X_1	X_2	Y
1	1	1	1
1	2	1	3
1	3	1	3
1	1	-1	2
1	2	-1	2
1	3	-1	1

Adjust operator

$$\begin{bmatrix} X'X & X'Y & I \\ Y'X & Y'Y & 0 \end{bmatrix}$$

Sweep operator

$$\begin{bmatrix} X'X & X'Y \\ Y'X & Y'Y \end{bmatrix}$$

	Adjust Operator				Sweep Operator			
	X_0	X_1	X_2	b	X_0	X_1	X_2	
Step 0 Form Matrix	6	12	0	12	1	0	0	6 12 0 12
	12	28	0	25	0	1	0	12 28 0 25
	0	0	6	2	0	0	1	0 0 6 2
	12	25	2	28	0	0	0	12 25 2 28
Step 1 Enter X_0	1	2	0	2	1/6	0	0	1/6 2 0 2
	0	4	0	1	-2	1	0	-2 4 0 1
	0	0	6	2	0	0	1	0 0 6 2
	0	1	2	4	-2	0	0	-2 1 2 4
Step 2 Enter X_1	1	0	0	3/2	7/6	-1/2	0	7/6 -1/2 0 3/2
	0	1	0	1/4	-1/2	1/4	0	-1/2 1/4 0 1/4
	0	0	6	2	0	0	1	0 0 6 2
	0	0	2	15/4	-3/2	-1/4	0	-3/2 -1/4 2 15/4
Step 3 Enter X_2	1	0	0	3/2	7/6	-1/2	0	7/6 -1/2 0 3/2
	0	1	0	1/4	-1/2	1/4	0	-1/2 1/4 0 1/4
	0	0	1	1/3	0	0	1/6	0 0 1/6 -1/3
	0	0	0	37/12	-3/2	-1/4	-1/3	-3/2 -1/4 -1/3 37/12

These tableaus represent the following models with respect to Y.

Step	Model	b values	ESS
0	$Y = 0$	.	28
1	$Y = b_0$	2	4
2	$Y = b_0 + b_1X_1$	3/2, 1/4	15/4
3	$Y = b_0 + b_1X_1 + b_2X_2$	3/2, 1/4, 1/3	37/12

By observing tableaus in Adjust operator, note that whenever a variable is in the model, its associated X'X column is an identity column ; whenever a variable is not in the model, its associated column in the original identity matrix remains unchanged. The Sweep operator takes advantages of these features : After adjusting the tableau for a particular variable, the Sweep operator replaces the identity column by the associated column in the original identity matrix. From these tableaus it can be seen that if an $(n+1)x(n+1)$ matrix A is swept on k th pivotal element, then it is transformed into a matrix A^* such that

$$
\begin{aligned}
a^*_{kk} &= \frac{1}{a_{kk}}, \\
a^*_{ik} &= -a_{ik}a^*_{kk} && i \neq k, \\
a^*_{kj} &= a_{kj}a^*_{kk} && j \neq k, \\
a^*_{ij} &= a_{ij} - a_{ik}a_{kj}a^*_{kk} && i,j \neq k.
\end{aligned}
$$

To compute a^*_{kk} requires one reciprocal. To compute a^*_{ik} and a^*_{kj} requires one multiplication. To compute a^*_{ij} requires two multiplications and one addition. For example, when k=1 we have

$$
A^* = \begin{bmatrix}
\frac{1}{a_{11}} & \frac{a_{12}}{a_{11}} & \cdots & \frac{a_{1n+1}}{a_{11}} \\
\frac{-a_{21}}{a_{11}} & a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}} & & \\
\vdots & & & \\
\frac{-a_{n+1,1}}{a_{11}} & & &
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
0 & \overbrace{1 \quad 1 \quad \cdots \quad 1}^{n} \\
1 & 2 \quad 2 \quad \cdots \quad 2 \\
1 & 2 \quad 2 \quad \cdots \quad 2 \\
\vdots & \\
1 & 2 \quad 2 \quad \cdots \quad 2
\end{bmatrix}
\Big\} n
$$

The elements of the right matrix represent the number of multiplications required to compute its own. However, because the matrix A is symmetric, it is really necessary to work with only the upper triangle. The symmetry of the matrix A can be preserved by changing the sign of the pivot. In this case the number of multiplications required to obtain A^* is as follows :

$$\begin{bmatrix} 0 & 1 & 1 & \cdots & 1 \\ 0 & 2 & 2 & \cdots & 2 \\ 0 & 0 & 2 & \cdots & 2 \\ \vdots & \vdots & \vdots & & \\ 0 & 0 & 0 & \cdots & 2 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \} n \end{matrix} \quad \overbrace{\qquad\qquad}^{n}$$

$$n + \sum_{i=1}^{n} 2i = n^2 + 2n$$

And the number of additions is as follows :

$$\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \} n$$

$$\sum_{i=1}^{n} i = \frac{n^2}{2} + \frac{n}{2}$$

Since we need n sweepings on n pivotal elements to invert the matrix A, the number of multiplications required is

$$n(n^2+2n) = n^3 + 2n^2$$

and the number of additions required is

$$n\left(\frac{n^2}{2} + \frac{n}{2} \right) = \frac{n^3}{2} + \frac{n^2}{2}$$

V. Comparison based on the number of arithmetic operations

Table 3 summarizes the number of multiplications and additions required for computing $\underline{b}$, ESS, and the inverse of $X'X$ from the normal equations : $X'Xb=X'Y$. In the case of both Gaussian elimination and Cholesky method, an extra 2n multiplications and one addition are required to compute ESS.

Table 3. Number of computations required for b, ESS, and $(X'X)^{-1}$ using three algorithms

	Multiplications	Additions
Gaussian elimination	$n^3 + n^2 + n$	$n^3 - n + 1$
Cholesky method	$(n^3+4n^2+5n)/2$	$(n^3+ n^2-2n+2)/2$
Sweep operator	n^3+2n^2	$(n^3+ n^2)/2$

Gaussian elimination and the Sweep operator require about twice the number of multiplications as the Cholesky method. And Gaussian elimination requires twice the number of additions as the Cholesky method and the Sweep operator. By IBM SYSTEM/370 MODEL 158, 5 microseconds is taken to perform one addition, and 19 microseconds to perform one multiplication. Using this information we can find the execution time to obtain the b values, ESS, and $(X'X)^{-1}$ for three methods as follows :

Gaussian : $(24n^3+19n^2+14n+5)$ microsec.

Cholesky : $(12n^3+40.5n^2+42.5n+5)$ microsec.

Sweep : $(21.5n^3+40.5n^2)$ microsec.

Table 4 shows the execution time of algorithms for different n values.

Table 4. Comparison of execution time for three algorithms(in seconds)

n	5	10	20	30	50
Gaussian elimination	.004	.026	.200	.666	3.048
Cholesky method	.003	.017	.113	.362	1.603
Sweep operator	.004	.026	.188	.617	2.789

It can be seen that Cholesky method is the fastest and the Sweep operator is the next.

VI. Results and Discussion

We generated several X'X matrices, computed the inverse using all three methods and then compared their accuracies. The generated X'X matrices ranged from ill-conditioned to well-conditioned.

1. Condition number of X'X

A system of normal equations X'Xb=X'Y is said to be ill-conditioned if small variations in the elements of X'X and X'Y have a large effect on the exact solution, b. For example, the difference, δb, between the solution of X'Xb=X'Y and that of $(X'X+\delta X'X)(b+\delta b)=X'Y+\delta X'Y$ can be expressed as

$$\delta b = (X'X + \delta X'X)^{-1}(\delta X'Y - \delta X'Xb) ,$$

and its value depends critically on the inverse matrix. If X'X is nearly singular, that is, small changes in its elements can cause singularity, then δb could be very large. For the comparison of the accuracy of the computed solution vector, b, we concentrated on the accuracy of the computed inverse of X'X which would mainly affect the accuracy of the computed b. To obtain many X'X matrices, from well-conditioned to ill-conditioned, we started with an arbitrary correlation matrix, and produced correlation matrices with condition numbers 10, 10^3, 10^5, and 10^8. For this we used the subprogram GECORR(Milliken,1981) the input of which is any correlation matrix and a desired condition number, and the output of which is a new correlation matrix with the desired condition number. With each new correlation matrix, R, we generated data and constructed 20 uncorrected X'X matrices of dimension, (n+1) x (n+1). The sizes of the correlation matrices were 10 and 20 resulting in 11x11 and 21x21 uncorrected X'X matrices which were generated using

40 observations for each variable. Table 5 shows the range of the condition numbers of the X'X matrices generated from correlation matrices with the condition numbers 10, 10^3, 10^5, and 10^8.

Table 5. The range of the condition number of the X'X matrices

	Cond. # of correlation matrix	Cond. # of X'X matrix
A	10	$10 - 10^2$
B	10^3	$10^3 - 10^4$
C	10^5	$10^5 - 10^6$
D	10^8	$10^8 - 10^9$

In practice it is common for one variable to be highly correlated with a linear combination of other variables so that the columns of X may be close to being linearly dependent. This means that X'X may be near singularity, the smallest eigenvalue will be small, and the condition number will be large. Examples of ill-conditioned matrices include Longley's(1967) test data with condition number of 4.8×10^9 and Wampler's(1970) polynomial models which had a value of the order of 10^6. In this report the X'X matrices in D were considered ill-conditioned. Those in A were considered well-conditioned while those in B and C were in between.

2. Generating X'X matrix

First we generated data with distributions for each variable $X_i \sim N(\mu_i, \sigma_i^2)$ and specified correlations between the variables. In fact, it was assumed that each variable followed the Standard Normal distribution such that $\mu_i=0$, and $\sigma_i^2=1$ for all i. Correlations between variables were obtained from the correlation matrix, R, which depended on the condition number and the matrix size desired for X'X (See Appendix B).

To generate data for n variables, X_1 through X_n, we repeated the following formula for j=2 to n.

$$X_1 = Z_1 \cdot \sigma_1 + \mu_1 = Z_1$$

$$X_j = Z_j \cdot \sigma_j^* + \mu_j^*$$

Z_j was obtained from RNOR(0) of Marsaglia's(1976) Super-Duper random number generator as a Standard Normal deviate.

σ_j^{*2} is the conditional variance of X_j given $X_1=x_1, \ldots, X_{j-1}=x_{j-1}$.

$$\sigma_j^{*2} = \sigma_j^2 - (\sigma_{j,1} \cdots \sigma_{j,j-1}) \sum_{j-1}^{-1} \begin{pmatrix} \sigma_{1,j} \\ \vdots \\ \sigma_{j-1,j} \end{pmatrix}, \quad \sigma_j^2 = 1$$

where σ_{jk} is the covariance between X_j and X_k, and $\sum_{j-1}$ is the covariance matrix for $X_1, \ldots, X_{j-1}$ which can be obtained from the correlation matrix R (R_{j-1} is the submatrix of R, with first j-1 rows and columns) by $\sum_{j-1} = D'R_{j-1}D = R_{j-1}$ since the diagonal matrix, D, with i th diagonal element equal to σ_i is an identity matrix.

μ_j^* is the conditional mean of X_j given $X_1=x_1, \ldots, X_{j-1}=x_{j-1}$.

$$\mu_j^* = \mu_j + (\sigma_{j,1} \cdots \sigma_{j,j-1}) \sum_{j-1}^{-1} \begin{pmatrix} x_1 - \mu_1 \\ \vdots \\ x_{j-1} - \mu_{j-1} \end{pmatrix}, \text{ all } \mu_i = 0.$$

Repeating this procedure for the number of observations desired, we obtained a data matrix, X, and an X'X matrix of dimension (n+1)x(n+1) for the first order linear model, $Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n$.

3. Accuracy

To compare the accuracies of the inverse computed using the three algorithms, we computed the amount of error in the inverse as the sum of the absolute differences between the product of the original X'X and its' computed X'X inverse and the identity matrix(It is denoted by $\sum_{i=1}^{n+1} \sum_{j=1}^{n+1} \left| (X'X)(X'X)^{-1} - I \right|$). Tables 6 and 7 show the average

amount of error in the computed inverses for different condition numbers. Matrices of size 11 x 11 are in Table 6 and size 21 x 21 are in Table 7. In all cases 20 matrices of each size were generated as described above.

Table 6. The average error amount of the computed inverse for three algorithms (11 x 11 X'X matrices)

Cond. # of X'X	$10\text{-}10^2$	$10^3\text{-}10^4$	$10^5\text{-}10^6$	$10^8\text{-}10^9$
Gaussian elimination	.332D-13	.048D-11	.039D-09	.040D-06
Cholesky method	.328D-13	.055D-11	.041D-09	.046D-06
Sweep operator	.353D-13	.130D-11	.163D-09	.164D-06

Table 7. The average error amount of the computed inverses for three algorithms (21 x 21 X'X matrices)

Cond. # of X'X	$10\text{-}10^2$	$10^3\text{-}10^4$	$10^5\text{-}10^6$	$10^8\text{-}10^9$
Gaussian elimination	.203D-12	.851D-11	.807D-09	.923D-06
Cholesky method	.165D-12	.559D-11	.650D-09	.508D-06
Sweep operator	.223D-12	.810D-11	.869D-09	.693D-06

It can be seen that Gaussian elimination and Cholesky method are more accurate than the Sweep operator for size 11 x 11 matrices, but Cholesky method is the most accurate for size 21 x 21 matrices.

4. Execution Time

Table 8 and 9 show the average execution time to find the b values, ESS, and $(X'X)^{-1}$ for different condition numbers of X'X matrices.

Table 8. The average execution time to find the b values, ESS, and $(X'X)^{-1}$ for three algorithms(11x11 X'X matrices)

Cond. # of X'X	$10-10^2$	10^3-10^4	10^5-10^6	10^8-10^9
Gaussian elimination	.053	.061	.054	.060
Cholesky method	.021	.025	.024	.037
Sweep operator	.036	.044	.038	.040

Table 9. The average execution time to find the b values, ESS, and $(X'X)^{-1}$ for three algorithms(21x21 X'X matrices)

Cond. # of X'X	$10-10^2$	10^3-10^4	10^5-10^6	10^8-10^9
Gaussian elimination	.343	.340	.337	.331
Cholesky method	.133	.109	.111	.113
Sweep operator	.218	.221	.212	.234

As shown by the number of arithmetic operations Cholesky's method is the fastest and the Sweep operator is the next.

In conclusion Cholesky's method appeared to be the most accurate and fastest of the three algorithms and, therefore, the one we would recommend. The Sweep operator's accuracy seemed to be slightly less than that of Gaussian elimination but its execution time was consistently less.

References

Conte, S.D. and de Boor, Carl(1972), Elementary Numerical Analysis, New York : McGraw-Hill

Draper, N.R.,and Smith, H.(1981), Applied Regression Analysis, New York : John Wiley & Sons

Golub, G.H.(1969), " Matrix Decomposition and Statistical Calculations," Statistical Computation, 365-398, New York : Academic Press

Goodnight, J.H.(1979), " A Tutorial on the Sweep Operator," The American Statistician, Vol 33, No.3, 149-158

Graybill, F.A.(1976), Theory and Application of the Linear Model, North Scituate : Duxbury Press

IBM SYSTEM/370 MODEL 158 Functional Characteristics

Issaacson, E., and Keller, H.B.(1966), Analysis of Numerical Methods, New York : John Wiley & Sons

Kronsjo, L.I.(1979), Algorithms- Their Complexity and Efficience, New York : John Wiley & Sons

Longley, J.W.(1967), " An Appraisal of Least Squares Programs for the Electronic Computer from the Point of View of Use," Journal of the American Statistical Association,62, 819-841

Marsaglia,G.(1976)," Improvements on Fast Methods for Generating Normal Random Variables," Information Processing Letters, Vol 5, No.2

Milliken, G.A.(1981), " Computer program for generating correlation matrix "

Wampler, R.H.(1970), " A Report on the Accuracy of Some Wideley Used Least Squares Computer Programs," Journal of the American Statistical Association, 65, 549-565

Wilkinson, J.H.(1961), " Error Analysis of Direct Methods of Matrix Inversion," Journal of the Association of Computing Machinery, Vol 8, 281-330

Wilkinson, J.H.(1963), Rounding Errors in Algebraic Processes, Englewood Cliffs : Prentice-Hall

APPENDICES

APPENDIX A

COMPUTER PROGRAM FOR GENERATING X'X MATRIX AND COMPUTING ITS INVERSE USING THREE ALGORITHMS

```
**********************************************************

THIS PROGRAM PERFORMS THE FOLLOWING PROCEDURES.

1. GENERATE DATA GIVEN DISTRIBUTIONS AND CORRELATION
COEFFICIENTS BETWEEN VARIABLES.
2. CONSTRUCT UNCORRECTED X'X MATRIX
3. COMPUTE THE CONDITION NUMBER OF X'X
   ( WE GENERATE MANY X'X MATRICES WITH DIFFERENT
     CONDITION NUMBERS )
4. COMPUTE THREE INVERSES OF X'X , SOLUTION VECTORS ,
   AND ERROR SUM OF SQUARES USING THREE ALGORITHMS.
5. COMPARE THEIR ACCURACIES, AND EXECUTION TIME

**********************************************************
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 TITLE(10),FMT(10)
      REAL*8 MEAN(20),VAR(20),R(20,20),D(20,20)
      REAL*8 COV(20,20),LTCOV(260),ICOV(20,20),PCIC(21)
      REAL*8 DX(100,21),XPX(21,21),RSUM(21),INORM
      REAL*8 ERR(3),DSUM(3),SSQS(3),EXTIME(3),TSUM(3)
      REAL*8 UPPTR(400),EV(400),STOR(400),Y(100),XPY(21)
      REAL*8 B(21),A(460),X(100,21),IXPX(21,21),TSQS(3)
      REAL*8 DMIN(22),COV1(20,20),LTXPX(460)
      INTEGER V(22)
      READ(5,5)  TITLE
    5 FORMAT(10A8)
      WRITE(6,6)  TITLE
    6 FORMAT(1X,10A8)
      READ(5,7) IX,IY
    7 FORMAT(1X,I10,1X,I10)
      WRITE(6,7) IX,IY
   10 READ(5,15,END=900) NOV,NOBS,NOMT,CCND,IPRINT
   15 FORMAT(I2,I3,I3,1X,E16.5,14X,I1)
      WRITE(6,16) NOV,NOBS,NOMT,COND,IPRINT
   16 FORMAT(1X,I2,I3,I3,1X,E16.5,14X,I1)
      N=1
      DJ=1.0
      DJJ=2.0
      NOV1 = NOV + 1
      NOV2 = NOV1+1
      IF(NOV1.GT.21.OR.NOBS.GT.100) GO TO 900
C
C   ***  NOV ; TOTAL NO. OF VARIABLES
C   ***  NOBS ; TOTAL NO. OF OBSERVATIONS
C   ***  MEAN(I) ; MEAN OF ITH VAR.
C   ***  VAR(I) ; VARIANCE OF ITH VAR.
C   ***  N  ;  NO. OF X'X MATRIX GENERATED
C   ***  NOV1  ;  DIMENSION OF X'X MATRIX
C   ***  FMT ; VARIABLE FORMAT FOR READING CORR. MATRIX
C
C
      READ(5,20) (MEAN(I),I=1,NOV)
      READ(5,20) (VAR(I),I=1,NOV)
      WRITE(6,21) (MEAN(I),I=1,NOV)
      WRITE(6,21) (VAR(I),I=1,NOV)
   20 FORMAT(20F4.1)
   21 FORMAT(1X,20F6.1)
      READ(5,5) FMT
      WRITE(6,6) FMT
      DO 25 I=1,NOV
      READ(5,FMT) (R(I,J),J=1,NOV)
      WRITE(6,22) (R(I,J),J=1,NOV)
   22 FORMAT(1X,11F12.4/1X,11F12.4)
   25 CONTINUE
C
C   ***  CALL THE SUBROUTINE GECORR FOR GENERATING CORR.
C        MATRIX GIVEN A DESIRED CONDITION NUMBER
C
      CALL GECORR(R,NOV ,COND,IPRINT)
C
      DO 30 I=1,NOV
      DO 30 J=1,NOV
      D(I,J) = 0.
   30 CONTINUE
```

```
      DO 32 I=1,NOV
   32 D(I,I) = DSQRT(VAR(I))
      DO 35 I=1,NOV
      DO 35 J=1,NOV
      SUM=0.
      DO 36 K=1,NOV
   36 SUM = SUM + D(I,K)*R(K,J)
      COV1(I,J) = SUM
   35 CONTINUE
      DO 40 I=1,NOV
      DO 40 J=1,NOV
      SUM=0.
      DO 41 K=1,NOV
   41 SUM = SUM + COV1(I,K)*D(K,J)
   40 COV(I,J) = SUM
      DO 50 I=1,NOV
      II = I*(I-1)/2
      DO 50 J=1,I
   50 LTCOV(II+J) = COV(I,J)
C
C  ***  COV(I,J) ; COVARIANCE MATRIX
C  ***  LTCOV(K) ; LOWER TRIANGULAR OF COVARIANCE MATRIX
C
      CALL RSTART(IX,IY)
C
C  ***  RSTART GIVES SEED FOR GENERATING RANDOM NUMBER
C  ***  RNOR(0) ; SUPER-DUPER RANDOM NUMBER GENERATOR AS
C                 A STANDARD NORMAL DEVIATE
C
      ITNOV=(NOV*NOV+NOV)/2
   53 IF(N.GT.NOMT)  GO TO 800
      WRITE(6,55) TITLE ,N
   55 FORMAT(/10A8,'MATRIX # ',I3,//)
      IF(IPRINT.EQ.0) GO TO 77
      IF(IPRINT.EQ.1) GO TO 77
      WRITE(6,60)
   60 FORMAT( 1X,'COVARIANCE MATRIX',///)
      DO 70 I=1,NOV
   70 WRITE(6,71) (COV(I,J),J=1,NOV)
   71 FORMAT(1X,10F12.5)
   77 SQRTVR = DSQRT(VAR(1))
      DO 290 IOBS=1,NOBS
      STDN = RNOR(0)
      X(IOBS,1) = STDN*SQRTVR + MEAN(1)
      IVAR=1
      IF(IPRINT.EQ.0) GO TO 82
      IF(IPRINT.EQ.1) GO TO 82
      WRITE(6,80) IVAR,STDN,X(IOBS,1)
   80 FORMAT(5X,'X',I2,5X,'RAN. #',F8.4,5X,'DATA',F10.4)
   82 DC = COV(2,1)/COV(1,1)
      CONM = MEAN(2) + DC*(X(IOBS,1)-MEAN(1))
      CONV = VAR(2) - DC*COV(1,2)
      STDN = RNOR(0)
      IVAR=2
      X(IOBS,2) = STDN*DSQRT(CONV) + CONM
      IF(IPRINT.EQ.0) GO TO 85
      IF(IPRINT.EQ.1) GO TO 85
      WRITE(6,80) IVAR,STDN,X(IOBS,2)
C
C  ***  X(I,J) ; ITH OBSERVATION OF JTH VARIABLE
```

```
C  ***  CONM ; CONDITIONAL MEAN OF X2 GIVEN X1=C1
C  ***  CONV ; CONDITIONAL VARIANCE OF X2 GIVEN X1=C1
C
   85 DO 200 IVAR=3,NOV
      IVARM1 = IVAR - 1
      ITVAR = IVARM1*(IVARM1+1)/2
      DO 90 I=1,ITVAR
   90 A(I) = LTCOV(I)
C
C  ***  A ; LOWER TRIANGULAR OF COVARIANCE MATRIX
C  ***  SUBROUTINE DMFSD AND DSINV IS TO CALCULATE THE
C       INVERSE OF COVARIANCE MATRIX
C
      CALL DMFSD(A,IVARM1)
      CALL DSINV(A,IVARM1)
      DO 100 I=1,IVARM1
      DO 100 J=1,I
      K = I*(I-1)/2 + J
      ICOV(I,J)=A(K)
      ICOV(J,I) = ICOV(I,J)
  100 CONTINUE
C
C  ***  ICOV ; FULL MATRIX OF COVARIANCE MATRIX INVERSE
C
      IF(IOBS.GT.1) GO TO 115
      IF(IPRINT.EQ.0) GO TO 115
      IF(IPRINT.EQ.1) GO TO 115
      WRITE(6,105)
  105 FORMAT(//T10,'INVERSE OF COVARIANCE MATRIX UNDER',
     11X,'CONSIDERATION',//)
      DO 110 I=1,IVARM1
  110 WRITE(6,71) (ICOV(I,J),J=1,IVARM1)
  115 DO 130 J=1,IVARM1
      PCIC(J) = 0.
      DO 120 I=1,IVARM1
      PCIC(J) = PCIC(J) + COV(IVAR,I)*ICOV(I,J)
  120 CONTINUE
  130 CONTINUE
      ADDM = 0.
      ADDV = 0.
      DO 140 J=1,IVARM1
      ADDM = ADDM + PCIC(J)*(X(IOBS,J)-MEAN(J))
      ADDV = ADDV + PCIC(J)*COV(J,IVAR)
  140 CONTINUE
      CONM = MEAN(IVAR) + ADDM
      CONV = VAR(IVAR) - ADDV
C
C  ***  CONM ; CONDITIONAL MEAN OF VARIABLE IVAR
C  ***  CONV ; CONDITIONAL VARIANCE OF VARIABLE IVAR
C
      STDN = RNOR(0)
      X(IOBS,IVAR) = STDN*DSQRT(CONV) + CONM
      IF(IPRINT.EQ.0) GO TO 200
      IF(IPRINT.EQ.1) GO TO 200
      WRITE(6,80) IVAR,STDN,X(IOBS,IVAR)
  200 CONTINUE
      IF(IPRINT.EQ.0) GO TO 290
      IF(IPRINT.EQ.1) GO TO 290
      WRITE(6,210) IOBS,(X(IOBS,J),J=1,NOV)
  210 FORMAT(//1X,'#',I2,10F12.4////)
```

```
  290 CONTINUE
      DO 295 I=1,NOBS
      DX(I,1) = 1.
      DO 295 J=1,NOV
      JJ=J+1
      DX(I,JJ)=X(I,J)
  295 CONTINUE
      IF(IPRINT.EQ.0) GO TO 345
      WRITE(6,300)
  300 FORMAT(///1X,'DATA MATRIX',//)
      DO 320 I=1,NOBS
      WRITE(6,310) (DX(I,J),J=1,NOV1)
  310 FORMAT(1X,11E12.4/1X,11E12.4)
  320 CONTINUE
      WRITE(6,340)
  340 FORMAT(///1X,'X''X MATRIX',//)
  345 DO 370 I=1,NOV1
      DO 360 J=1,NOV1
      XPX(I,J) = 0.
      DO 350 K=1,NOBS
  350 XPX(I,J) = XPX(I,J) +DX(K,I)*DX(K,J)
  360 CONTINUE
      IF(IPRINT.EQ.0) GO TO 370
      WRITE(6,310) (XPX(I,JJ),JJ=1,NOV1)
  370 CONTINUE
      DO 375 I=1,NOBS
  375 Y(I)=RNOR(0)
      DO 390 I=1,NOV1
      SUM=0.
      DO 380 J=1,NOBS
  380 SUM = SUM + DX(I,J)*Y(J)
  390 XPY(I) = SUM
      SUM = 0.
      DO 395 I=1,NOBS
  395 SUM = SUM + Y(I)*Y(I)
      YPY = SUM
C
C  ***  SUBROUTINE TARRAY IS CALLED TO TRANSFORM X'X
C       INTO UPPER TRIANGULAR MATRIX
C  ***  SUBROUTINE EIGEN IS CALLED TO FIND THE EIGEN
C       VALUES OF X'X
C
      CALL TARRAY(NOV1,21,UPPTR,XPX)
      CALL EIGEN(UPPTR,STOR,NOV1,1)
      NC=1
      DO 400 I=1,NOV1
      NC=I+(I**2-I)/2
      EV(I)=UPPTR(NC)
  400 CONTINUE
      WRITE(6,402)
  402 FORMAT(///,1X,'EIGEN VALUE FOR X''X MATRIX')
      WRITE(6,310) (EV(I),I=1,NOV1)
C
C  ***  SUBROUTINE SORT FINDS THE MAX AND MIN OF EIGEN
C       VALUES FOR CONDITION NUMBER
C
      CALL SORT(EV,NOV1)
      CONDNO = EV(1)/EV(2)
      WRITE(6,410) CONDNO
  410 FORMAT(//1X,'CONDITION NUMBER OF X''X = ',E13.5)
```

```
      DO 420 I=1,NOV1
      II = I*(I-1)/2
      DO 420 J=1,I
  420 LTXPX(II+J) = XPX(I,J)
C
C  ***  LTXPX ; LOW TRIANGULAR OF XPX MATRIX
C
      IF(IPRINT.EQ.0) GO TO 425
      WRITE(6,422)
  422 FORMAT('1',/1X,'INVERSE BY CHOLESKY METHOD'//)
C
C  ***  SUBROUTINE DMFSD FINDS CHOLESKY DECOMPO. OF X'X
C  ***  SUBROUTINE DSINV CALCULATES THE INVERSE OF X'X
C
  425 CONTINUE
C
C  ***  LET LL'=A. THEN AX=B IS LL'X=B.
C       FOR THE SOLUTION X, WE SOLVE LY=B AND L'X=Y.
C  ***  SUBROUTINE SOLTON IS TO SOLVE LY=B AND L'X=Y.
C
      CALL INTIME(ITIME1)
      CALL DMFSD(LTXPX,NOV1)
      CALL SOLTON(NOV1,LTXPX,B,XPY,YPY,ESS)
      CALL DSINV(LTXPX,NOV1)
      CALL INTIME(ITIME2)
      EXTIME(1)  = (ITIME2 - ITIME1)/100.
      DO 430 I=1,NOV1
      II = I*(I-1)/2
      DO 430 J=1,I
      IXPX(I,J) = LTXPX(II+J)
      IXPX(J,I) = IXPX(I,J)
  430 CONTINUE
      IF(IPRINT.EQ.0) GO TO 440
      DO 435 I=1,NOV1
      WRITE(6,310) (IXPX(I,J),J=1,NOV1)
  435 CONTINUE
  440 DERR=0.
      CALL ERRINV(XPX,IXPX,NOV1,DERR,IPRINT)
C
C  ***  SUBROUTINE ERRINV IS TO CALCULATE THE AMOUNT OF
C       THE ERROR OF THE COMPUTED INVERSE OF X'X
C
      ERR(1) = DERR
      WRITE(6,480) N,ERR(1)
  480 FORMAT(//1X,'MATRIX #',I3,T15,
     1    'CHOLESKY ERROR = ',E12.5)
      WRITE(6,490) EXTIME(1)
  490 FORMAT(T15,'EX. TIME = ',F12.7)
      WRITE(6,494) (B(I),I=1,NOV1)
  494 FORMAT(T15,'SOLUTION : ',8E13.5)
C
C  ***  COMPUTE THE INVERSE OF X'X USING SWEEP OPERATOR
C
      DMIN(1)=0.
      DO 500 I=1,NOV2
      V(I)=1
  500 CONTINUE
      DO 505 I=1,NOV1
      II = I*(I-1)/2
      DO 505 J=1,I
```

```
  505 LTXPX(II+J)  = XPX(I,J)
      IF(IPRINT.EQ.0) GO TO 507
      WRITE(6,506)
  506 FORMAT('1',/1X,'INVERSE BY SWEEP OPERATOR' //)
C
C  ***  SUBROUTINE LTG2SP IS CALLED FOR X'X INVERSE
C
  507 NTOT = (NOV1*NOV1+NOV1)/2
      DO 508 I=1,NOV1
  508 LTXPX(NTOT+I) = XPY(I)
      LTXPX(NTOT+NOV2) = YPY
      CALL INTIME(ITIME1)
      CALL LTG2SP(1,NOV2,LTXPX,DMIN,V)
      DO 510 I=1,NOV2
  510 DMIN(I) = 1.0E-12*LTXPX(I*(I+1)/2)
      DO 520 K=2,NOV1
      CALL LTG2SP(K,NOV2,LTXPX,DMIN,V)
  520 CONTINUE
      CALL INTIME(ITIME2)
      EXTIME(2)  = (ITIME2 - ITIME1)/100.
      DO 530 I=1,NOV1
      II = I*(I-1)/2
      DO 530 J=1,I
      IXPX(I,J) = LTXPX(II+J)
      IXPX(J,I) = IXPX(I,J)
  530 CONTINUE
      IF(IPRINT.EQ.0) GO TO 540
      DO 535 I=1,NOV1
      WRITE(6,310) (IXPX(I,J),J=1,NOV1)
  535 CONTINUE
  540 DERR=0.
      CALL ERRINV(XPX,IXPX,NOV1,DERR,IPRINT)
      ERR(2) = DERR
      WRITE(6,580) N,ERR(2)
  580 FORMAT(//1X,'MATRIX #',I3,T15,
     1       'SWEEP ERROR = ',E12.5)
      WRITE(6,490) EXTIME(2)
      DO 590 I=1,NOV1
  590 LTXPX(NTOT+I) = -LTXPX(NTOT+I)
      WRITE(6,494) (LTXPX(NTOT+I),I=1,NOV1)
      DO 600 I=1,NOV1
      DO 600 J=1,NOV1
      IXPX(I,J) = XPX(I,J)
  600 CONTINUE
C
C  ***  COMPUTE X'X INVERSE USING GAUSSIAN ELIMINATION
C
      CALL INTIME(ITIME1)
      CALL GAUEL(NOV1,IXPX,B,XPY,YPY,ESS,IPRINT)
      CALL INTIME(ITIME2)
      EXTIME(3)  = (ITIME2 - ITIME1)/100.
      IF(IPRINT.EQ.0) GO TO 640
      WRITE(6,610)
  610 FORMAT(///1X,'INVERSE BY GAUSSIAN ELIMINATION',//)
      DO 635 I=1,NOV1
      WRITE(6,310) (IXPX(I,J),J=1,NOV1)
  635 CONTINUE
  640 DERR=0.
      CALL ERRINV(XPX,IXPX,NOV1,DERR,IPRINT)
      ERR(3) = DERR
```

```
      WRITE(6,680) N,ERR(3)
  680 FORMAT(//1X,'MATRIX #',I3,T15,
     1    'GAUSSIAN ERROR = ',E12.5)
      WRITE(6,490) EXTIME(3)
      WRITE(6,494) (B(I),I=1,NOV1)
      IF(N.GE.2) GO TO 710
      DO 700 I=1,3
      SSQS(I) = 0.0
      TSQS(I) = 0.0
      TSUM(I) = EXTIME(I)
  700 DSUM(I) = ERR(I)
      N=N+1
      GO TO 53
  710 N=N+1
      DJ=DJ+1.0
      DO 720 I=1,3
      DSUM(I) = DSUM(I) + ERR(I)
      DXI = DJ*ERR(I)-DSUM(I)
      SSQS(I) = SSQS(I) + DXI*DXI/DJJ
      TSUM(I) = TSUM(I) + EXTIME(I)
      DXT = DJ*EXTIME(I)-TSUM(I)
      TSQS(I) = TSQS(I) + DXT*DXT/DJJ
  720 CONTINUE
      DJJ = DJJ+DJ+DJ
      GO TO 53
  800 DJM1 = DJ-1
      SQRTD=DSQRT(DJ)
      WRITE(6,805)
  805 FORMAT('1',/,T38,'ERROR',T78,'EX. TIME',//)
      WRITE(6,810)
  810 FORMAT(T31,'AVERAGE',T43,'STD. ERROR',T71,
     1  'AVERAGE',T83,'STD. ERROR')
      DO 840 I=1,3
      AVG = DSUM(I)/DJ
      VARI= SSQS(I)/DJM1
      STDDEV =DSQRT(VARI)
      STDERR = STDDEV/SQRTD
      TAVG = TSUM(I)/DJ
      TVAR = TSQS(I)/DJM1
      TDEV =DSQRT(TVAR)
      TERR = TDEV/SQRTD
      WRITE(6,820) I,AVG,STDERR,TAVG,TERR
  820 FORMAT(/T10,'METHOD ',I1,T25,E13.5,T40,E13.5,T65,
     1  E13.5,T80,E13.5)
  840 CONTINUE
      CALL RSTOP(JX,JY)
      WRITE(6,890) JX,JY
  890 FORMAT(///,1X,'FOR A NEW SEED ',2I12)
  900 STOP
      END
```

```
      SUBROUTINE GECORR(A,NR,C,IPRINT)
      IMPLICIT REAL*8(A-H,O-Z)
C     GENERATE A P.D. CORRELATION MATRIX
      DIMENSION A(20,20), W(20,20), AS(20,20), BS(400),
     1          EV(20),FMT(10)
C     NR-- NUMBER OF ROWS (AND COLS) OF PROPOSED MATRIX
C     C--- DESIRED CONDITION NUMBER
      MNR = 20
C     A--- THE PROPOSED CORRELATION MATRIX
C     W--- THE WEIGHT MATRIX
      DO 4 I=1,NR
      DO 4 J=1,NR
      W(I,J) = 1.0
      IF(I.EQ.J) W(I,J)=0.0
    4 CONTINUE
      WRITE (6,6) NR,C
    6 FORMAT (1H1,10X,22HTHE NUMBER OF ROWS IS ,I3,3X,
     130HTHE IDEAL CONDITION NUMBER IS ,E16.5)
      IF(IPRINT.EQ.0) GO TO 50
      WRITE (6,7)
    7 FORMAT (10X,28H THE PROPOSED MATRIX IS--A--)
      DO 8 I=1,NR
    8 WRITE (6,11) (A(I,J),J=1,NR)
   11 FORMAT(/1X,10F13.4/1X,10F13.4)
C     CALL THE SUBROUTINE FOR COMPUTING THE P.D.
C     CORRELATION MATRIX
   50 CALL PDCORR (A,W,AS,NR,C,AK,CODE,CT,BS,MNR,EV,
     1             IPRINT)
C     OUTPUT RESULTS
      IF(IPRINT.EQ.0) GO TO 60
      WRITE (6,12)
   12 FORMAT (10X,36H THE NEW CORRELATION MATRIX IS--AS--)
      DO 13 I=1,NR
   13 WRITE (6,11) (AS(I,J),J=1,NR)
   60 WRITE (6,16) CT
   16 FORMAT (10X,31H THE CONDITION NUMBER OF AS IS ,E16.5)
      WRITE (6,17) (EV(I),I=1,NR)
   17 FORMAT(10X,28HTHE CHARACTERISTIC ROOTS ARE/,
     1       10X,10E12.5)
      DO 20 I=1,NR
      DO 20 J=1,NR
   20 A(I,J) = AS(I,J)
      RETURN
      END
```

```
      SUBROUTINE ARRAY (MODE,I,J,N,M,S,D)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION S(1), D(1)
      NI = N-I
      IF (MODE-1) 1,1,3
    1 IJ = I*J+1
      NM = N*J+1
      DO 2 K=1,J
      NM = NM-NI
      DO 2 L=1,I
      IJ = IJ-1
      NM = NM-1
    2 D(NM) = S(IJ)
      RETURN
    3 IJ = 0
      NM = 0
      DO 5 K=1,J
      DO 4 L=1,I
      IJ = IJ+1
      NM = NM+1
    4 S(IJ) = D(NM)
    5 NM = NM+NI
      RETURN
      END

      SUBROUTINE ASTAR(A,W,AS,NR,CT,AK,STOR,EV,EPL,
     1                 BS,IPRINT)
C     COMPUTE THE NEW MATRIX TO BE CONSIDERED FOR OPTIMUM
C     CORR MATRIX
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(20,20), W(20,20), AS(20,20), EV(20),
     1          STOR(400),BS(400)
      NRM1 = NR-1
      WRITE (6,1) AK
    1 FORMAT (10X,4HAK= ,E15.8)
   10 DO 2 I=1,NRM1
      K = I+1
      DO 2 J=K,NR
      AS(I,J) = A(I,J)*(1.0-1.0/(AK*W(I,J)))
    2 AS(J,I) = AS(I,J)
      DO 3 I=1,NR
    3 AS(I,I) = 1.0
      CALL TARRAY (NR,20,BS,AS)
      CALL EIGEN (BS,STOR,NR,1)
      N = 1
      DO 4 I=1,NR
      N = I+(I*I-I)/2
      EV(I) = BS(N)
    4 CONTINUE
C     COMPUTE THE CHARACTERISTIC ROOTS OF AS
C     SORT CHAR ROOTS TO GET MAX AND MIN
      CALL SORT (EV,NR)
C     COMPUTE CONDITION NUMBER
      CT = 2.0E20
      IF (DABS(EV(2)).GT.0.00000001) CT = EV(1)/EV(2)
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (6,5) CT,EV(1),EV(2)
    5 FORMAT (10X,3E16.8)
   20 RETURN
      END
```

```
      SUBROUTINE CHECK (CT,C,CHECKY,CS,SK,AK1)
      IMPLICIT REAL*8(A-H,O-Z)
C     DETERMINE IF THE CONDITION NUMBER IS SMALL ENOUGH---
C     IF SO CHEKY=1
      CHECKY = 0.0
      C20P = C*0.2
      DEL = (DABS(CT-C))*10000.0/C
      IF(DEL.LE.C20P) CHECKY=1.0
      IF (CHECKY.EQ.1.0) RETURN
      IF (CT.GT.C) CALL DECREA (SK,CS,AK1)
      IF (CT.LE.C) CALL INCREA (SK,CS,AK1)
      RETURN
      END



      SUBROUTINE DECREA (SK,CS,AK1)
      IMPLICIT REAL*8(A-H,O-Z)
C     DECREASE VALUE OF AK, IF JUST INCREASED THEN REDUCE
C     STEP SIZE--CS
      IF (SK.EQ.2.0) CS = CS/2.0
      AK1 = CS+(1.0-CS)*AK1
      SK = 1.0
      RETURN
      END
```

```
      SUBROUTINE EIGEN (A,R,N,MV)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1), R(1)
C     GENERATE IDENTITY MATRIX
      RANGE = 1.0E-6
      IF (MV-1) 1,4,1
    1 IQ = -N
      DO 3 J=1,N
      IQ = IQ+N
      DO 3 I=1,N
      IJ = IQ+I
      R(IJ) = 0.0
      IF (I-J) 3,2,3
    2 R(IJ) = 1.0
    3 CONTINUE
C       COMPUTE INITIAL AND FINAL NORMS(ANORM & ANORMX)
    4 ANORM = 0.0
      DO 6 I=1,N
      DO 6 J=I,N
      IF (I-J) 5,6,5
    5 IA = I+(J*J-J)/2
      ANORM = ANORM+A(IA)*A(IA)
    6 CONTINUE
      IF (ANORM) 32,32,7
    7 ANORM = 1.414*DSQRT(ANORM)
      ANRMX = ANORM*RANGE/ FLOAT(N)
C       INITIALIZE INDICATORS AND COMPUTE THRESHOLD, THR
      IND = 0
      THR = ANORM
    8 THR = THR/ FLOAT(N)
    9 L = 1
   10 M = L+1
C        COMPUTE SIN AND COS
   11 MQ = (M*M-M)/2
      LQ = (L*L-L)/2
      LM = L+MQ
      IF (DABS(A(LM))-THR) 25,12,12
   12 IND = 1
      LL = L+LQ
      MM = M+MQ
      X = 0.5*(A(LL)-A(MM))
      Y = -A(LM)/DSQRT(A(LM)*A(LM)+X*X)
      IF (X) 13,14,14
   13 Y = -Y
   14 SINX = Y/DSQRT(2.0*(1.0+(DSQRT(1.0-Y*Y))))
      SINX2 = SINX*SINX
      COSX = DSQRT(1.0-SINX2)
      COSX2 = COSX*COSX
      SINCS = SINX*COSX
C        ROTATE L AND M COLUMNS
      ILQ = N*(L-1)
      IMQ = N*(M-1)
      DO 24 I=1,N
      IQ = (I*I-I)/2
      IF (I-L) 15,22,15
   15 IF (I-M) 16,22,17
   16 IM = I+MQ
      GO TO 18
   17 IM = M+IQ
   18 IF (I-L) 19,20,20
```

```
   19 IL = I+LQ
      GO TO 21
   20 IL = L+IQ
   21 X = A(IL)*COSX-A(IM)*SINX
      A(IM) = A(IL)*SINX+A(IM)*COSX
      A(IL) = X
   22 IF (MV-1) 23,24,23
   23 ILR = ILQ+I
      IMR = IMQ+I
      X = R(ILR)*COSX-R(IMR)*SINX
      R(IMR) = R(ILR)*SINX+R(IMR)*COSX
      R(ILR) = X
   24 CONTINUE
      X = 2.0*A(LM)*SINCS
      Y = A(LL)*COSX2+A(MM)*SINX2-X
      X = A(LL)*SINX2+A(MM)*COSX2+X
      A(LM) = (A(LL)-A(MM))*SINCS+A(LM)*(COSX2-SINX2)
      A(LL) = Y
      A(MM) = X
C        TESTS FOR COMPLETION
C        TEST FOR M = LAST COLUMN
   25 IF (M-N) 26,27,26
   26 M = M+1
      GO TO 11
C        TEST FOR L = SECOND FROM LAST COLUMN
   27 IF (L-(N-1)) 28,29,28
   28 L = L+1
      GO TO 10
   29 IF (IND-1) 31,30,31
   30 IND = 0
      GO TO 9
C        COMPARE THRESHOLD WITH FINAL NORM
   31 IF (THR-ANRMX) 32,32,8
C        SQRT EIGENVALUES AND EIGENVECTORS
   32 IQ = -N
      N2 = N*N
      DO 36 I=1,N
      IQ = IQ+N
      LL = I+(I*I-I)/2
      JQ = N*(I-2)
      DO 36 J=I,N
      JQ = JQ+N
      MM = J+(J*J-J)/2
      IF (A(LL)-A(MM)) 33,36,36
   33 X = A(LL)
      A(LL) = A(MM)
      A(MM) = X
      IF (MV-1) 34,36,34
   34 DO 35 K=1,N
      ILR = IQ+K
      IMR = JQ+K
      X = R(ILR)
      R(ILR) = R(IMR)
   35 R(IMR) = X
   36 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PDCORR (A,W,AS,NR,C,AK,CODE,CT,BS,MNR,
     1                   EV,IPRINT)
C     THIS ROUTINE DETERMINES SHRIKUN CORRELATION MATRIX
C     BY USING WEIGHT MATRIX AND SATISFYING A PARTICULAR
C     CONDITION NUMBER
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(20,20), W(20,20), AS(20,20), EV(20),
     1          STOR(400),BS(400)
      EPL = 2.0E-20
C     FIRST DETERMINE IF SELECTED MATRIX SATISFIES
C     THE CONDITIONS
      CALL TARRAY (NR,20,BS,A)
      CALL EIGEN (BS,STOR,NR,1)
      NR2 = NR*NR
      N = 1
      DO 1 I=1,NR
      N = I+(I*I-I)/2
      EV(I) = BS(N)
    1 CONTINUE
      IF(IPRINT.EQ.0) GO TO 30
      WRITE (6,2) (EV(I),I=1,NR)
    2 FORMAT (10X,20F6.2)
   30 CONTINUE
      CALL SORT (EV,NR)
      IF(IPRINT.EQ.0) GO TO 40
      WRITE (6,2) (EV(I),I=1,NR)
   40 CONTINUE
C     DETERMINE MAX AND MIN CHAR ROOTS--COMPUTE COND. #
      CT = EV(1)/EV(2)
      AK = 1.0
C     DETERMINE IF THE WEIGHT MATRIX WILL YIELD A P.D.
C     MATRIX FOR AK=1
    3 CALL ASTAR (A,W,AS,NR,CT,AK,STOR,EV,EPL,BS,IPRINT)
      IF (EV(2).GE.0.0.AND.CT.LE.C) GO TO 5
C     IF ABOVE CONDITION IS NOT TRUE, THEN
C                         SHRINK THE ELEMENTS OF W
      DO 4 I=1,NR
      DO 4 J=1,NR
    4 W(I,J) = DSQRT(W(I,J))
      GO TO 3
C     THE WEIGHT MATRIX IS OK, NOW FIND OPTIMUM AK---
C     START SEARCH
    5 CS = .5
      AK = 2.0E20
      CALL ASTAR (A,W,AS,NR,CT,AK,STOR,EV,EPL,BS,IPRINT)
C     COMPUTE STARTING VALUE FOR AK
      AK1 = 1.0+(C-1.0)/(EV(1)-C*EV(2))
      CHECKY = 0.0
      SK = 0.0
C     DO ITERATIVE SEARCH
    6 AK = AK1
      CALL ASTAR (A,W,AS,NR,CT,AK,STOR,EV,EPL,BS,IPRINT)
      IF (EV(2).LE.0) CALL DECREA (SK,CS,AK1)
      IF (EV(2).GT.0) CALL CHECK (CT,C,CHECKY,CS,SK,AK1)
C     IF OPTIMUM MATRIX IS FOUND CHECKY=1
      IF (CHECKY.EQ.1.0) RETURN
      GO TO 6
      END
```

```
      SUBROUTINE INCREA (SK,CS,AK1)
      IMPLICIT REAL*8(A-H,O-Z)
C     INCREASE VALUE OF AK. IF LAST DECREASED THEN REDUCE
C     STEP SIZE--CS
      AK1 = AK1+(AK1-1.0)*CS/(2.0*(1.0-CS))
      IF (SK.EQ.1.0) CS = CS/1.5
      SK = 2.0
      RETURN
      END
      SUBROUTINE SORT (EV,NR)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION EV(1)
C     SORT ROUTINE DETERMINES MAX AND MIN---
C     STORES IN EV(1) AND EV(2)
C     FIND MAX
      EVX = -2.0E20
      DO 1 I=1,NR
      IF (EV(I).GT.EVX) EVX = EV(I)
    1 CONTINUE
C     FIND MIN
      EVN = 2.0E20
      DO 2 I=1,NR
      IF (EV(I).LT.EVN) EVN = EV(I)
    2 CONTINUE
C     STORE MAX IN EV(1) AND MIN IN EV(2)
      EV(1) = EVX
      EV(2) = EVN
      RETURN
      END
      SUBROUTINE TARRAY (NR,MR,T,S)
      IMPLICIT REAL*8(A-H,O-Z)
C     CONVERTS SYMMETRIC MATRIX S INTO UPPER TRIANGULAR
C     STORAGE MATRIX T USED FOR EIGEN
      DIMENSION T(1), S(1)
      M = 1
      DO 1 J=1,NR
      N = (J-1)*MR
      DO 1 I=1,J
      NN = N+I
      T(M) = S(NN)
    1 M = M+1
      RETURN
      END
```

```
      SUBROUTINE GAUEL(N,A,X,B,YPY,ESS,IPRINT)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 A(1)   ,AMULT(450),AUPP(450),ALOW(450)
      REAL*8 X(1),B(1),XPY(21)
      DO 5 I=1,N
    5 XPY(I) = B(I)
      NR=21
      NN=NR*NR
      DO 10 I=1,NN
      AMULT(I) = 0.
      AUPP(I) = 0.
   10 ALOW(I) = 0.
      NM1=N-1
      KK=-NR
      DO 200  K=1,NM1
      KK=KK+NR
      IF(IPRINT.EQ.0) GO TO 40
      WRITE(6,33) K,A(KK+K)
   33 FORMAT(//'  K=',I3,E16.5)
   40 IF (A(KK+K).LT.10.E-10) GO TO 990
      J=K
      L=K+1
      P =  1.0/A(KK+K)
      DO 170 I = L,N
      R = A(KK+I)*P
      AMULT(KK+I) = R
      A(KK+I) = 0.
      JJ=NR*(L-1)-NR
      DO 160 J=L,N
      JJ=JJ+NR
      A(JJ+I) = A(JJ+I)- R*A(JJ+K)
  160 CONTINUE
      B(I) = B(I) - R*B(K)
  170 CONTINUE
  200 CONTINUE
      II=-NR
      DO 205 I=1,N
      II=II+NR
  205 AMULT(II+I) = 1.
      DET = 1.
      II=-NR
      DO 220 I=1,N
      II=II+NR
      DET = DET *A(II+I)
  220 CONTINUE
      IF(DET.LT.10.E-10) GO TO 990
      NIN = NR*(N-1)
      X(N) = B(N)/A(NIN+N)
      K=N-1
      KK=NIN-NR
  305 SUM=0.
      J=N
      JJ=NIN
  310 SUM = SUM + A(JJ+K)*X(J)
      J=J-1
      JJ=JJ-NR
      IF(J-K) 999,330,310
  330 X(K) = (B(K)-SUM)/A(KK+K)
      K=K-1
      KK=KK-NR
```

```
      IF(K-1) 340,305,305
  340 IF(IPRINT.EQ.0) GO TO 400
      WRITE(6,341)
  341 FORMAT('1',/1X,'GAUSSIAN ELIMINATICN',//,1X,
     1   'UPPER TRIANGULAR MATRIX',//)
      WRITE(6,360) (A(I),I=1,NN)
  360 FORMAT(1X,10E13.4)
      WRITE(6,370)
  370 FORMAT(///, 1X,'LOWER TRIANGULAR MATRIX',//)
      WRITE(6,360) (AMULT(I),I=1,NN)
C
C  ***  COMPUTE THE INVERSE OF UPPERTRIANGULAR OF A
C
  400 II=-NR
      DO 470 I=1,N
      II=II+NR
  470 AUPP(II+I) = 1.0/A(II+I)
      DO 500 JJ=1,NM1
      J=N-JJ+1
      JX=NR*(J-1)
      JM1=J-1
      DO 490 II=1,JM1
      I=J-II
      IX=NR*(I-1)
      DSUM=0.
      I1=I+1
      KK=NR*I-NR
      DO 480 K=I1,J
      KK=KK+NR
  480 DSUM = DSUM + A(KK+I)*AUPP(JX+K)
      AUPP(JX+I) = -AUPP(IX+I)*DSUM
  490 CONTINUE
  500 CONTINUE
      IF(IPRINT.EQ.0) GO TO 535
      WRITE(6,510)
  510 FORMAT(//1X,'INVERSE OF UPPER TRIANGULAR MATRIX'/)
      WRITE(6,360)  (AUPP(I),I=1,NN)
C
C  ***  COMPUTE THE INVERSE OF LOWERTRIANGULAR OF A
C
  535 II=-NR
      DO 540 I=1,N
      II=II+NR
  540 ALOW(II+I) = 1.0
      NM1 = N-1
      JJ=-NR
      DO 600 J=1,NM1
      JJ=JJ+NR
      J1=J+1
      DO 590 I=J1,N
      DSUM=0.0
      IM1=I-1
      KK=NR*(J-1)-NR
      DO 580 K=J,IM1
      KK=KK+NR
  580 DSUM = DSUM + AMULT(KK+I)*ALOW(JJ+K)
      ALOW(JJ+I) = -DSUM
  590 CONTINUE
  600 CONTINUE
      IF(IPRINT.EQ.0) GO TO 650
```

```
      WRITE(6,610)
  610 FORMAT(//1X,'INVERSE OF LOWER TRIANGULAR MATRIX'/)
      WRITE(6,360)  (ALOW(I),I=1,NN)
C
C  ***  COMPUTE THE INVERSE OF A
C
  650 DO 700 I=1,N
      JJ=-NR
      DO 680 J=1,N
      JJ=JJ+NR
      JJI=JJ+I
      A(JJI) = 0.0
      KK=-NR
      DO 680 K=1,N
      KK=KK+NR
  680 A(JJI) = A(JJI) + AUPP(KK+I)*ALOW(JJ+K)
  700 CONTINUE
      SUM=0.
      DO 800 I=1,N
  800 SUM = SUM + X(I)*XPY(I)
      ESS = YPY - SUM
      GO TO 999
  990 WRITE(6,991)
  991 FORMAT( 1X,'THE MATRIX X''X IS SINGULAR.')
  999 RETURN
      END
```

```
      SUBROUTINE DSINV(A,N)
      REAL*8 A(1),DIN,WORK
      IPIV=N*(N+1)/2
      IND=IPIV
      DO 6 I=1,N
      DIN=1.0/A(IPIV)
      A(IPIV)=DIN
      MIN=N
      KEND=I-1
      LANF=N-KEND
      IF(KEND.LE.0)GOTO 5
      J=IND
      DO 4 K=1,KEND
      WORK=0.
      MIN=MIN-1
      LHOR=IPIV
      LVER=J
      DO 3 L=LANF,MIN
      LVER=LVER+1
      LHOR=LHOR+L
    3 WORK=WORK+A(LVER)*A(LHOR)
      A(J)=-WORK*DIN
    4 J=J-MIN
    5 IPIV=IPIV-MIN
    6 IND=IND-1
      DO 8 I=1,N
      IPIV=IPIV+I
      J=IPIV
      DO 8 K=I,N
      WORK=0.
      LHOR=J
      DO 7 L=K,N
      LVER=LHOR+K-I
      WORK=WORK+A(LHOR)*A(LVER)
    7 LHOR=LHOR+L
      A(J)=WORK
    8 J=J+K
      RETURN
      END
```

```
      SUBROUTINE SOLTON(N,A,X,B,YPY,ESS)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 A(1),X(1),B(1),XPY(1),Y(100)
C
C  ***  SOLVE LY=B FOR Y.
C
      Y(1) = B(1)/A(1)
      DO 60 I=2,N
      ISTART = (I*I-I)/2 + 1
      LAST = ISTART + I - 1
      LASTM1 = LAST - 1
      DSUM = 0.
      II=0
      DO 50 J=ISTART,LASTM1
      II = II+1
      DSUM = DSUM + A(J)*Y(II)
   50 CONTINUE
      Y(I) = (B(I)-DSUM)/A(LAST)
   60 CONTINUE
C
C  ***  SOLVE L'X=Y FOR X.
C
      NTT = (N*N+N)/2
      X(N) = Y(N)/A(NTT)
      I=N-1
   70 DSUM=0.
      J=N
      II = (I*I+I)/2
   80 IJ = (J*J-J)/2 + I
      DSUM = DSUM + A(IJ)*X(J)
      J=J-1
      IF(J-I) 100,90,80
   90 X(I) = (Y(I)-DSUM)/A(II)
      I=I-1
      IF(I-1) 100,70,70
  100 SUM=0.
      DO 200 I=1,N
  200 SUM = SUM + X(I)*B(I)
      ESS = YPY - SUM
      RETURN
      END
```

```
      SUBROUTINE DMFSD(A,N)
      REAL*8 DOUB
      REAL*8 A(1),DPIV,DSUM,DSQRT
      KPIV=0
      DO 11 K=1,N
      KPIV=KPIV+K
      IND=KPIV
      LEND=K-1
      DOUB = A(KPIV)
      TOL=ABS(.01*SNGL(DOUB))
      DO 11 I=K,N
      DSUM=0.
      IF(LEND.EQ.0)GOTO 4
      DO 3 L=1,LEND
      LANF=KPIV-L
      LIND=IND-L
    3 DSUM=DSUM+A(LANF)*A(LIND)
    4 DSUM=A(IND)-DSUM
      IF(I.NE.K)GO TO 10
      DOUB = DSUM
      IF(SNGL(DOUB)-TOL)6,6,9
    6 IF(DSUM.LE.0.0)GOTO 12
      KT=K-1
      WRITE(6,100)KT
  100 FORMAT(20X,'ROUNDING ERROR IN ROW ',I2)
    9 DPIV=DSQRT(DSUM)
      A(KPIV)=DPIV
      DPIV=1.0/DPIV
      GO TO 11
   10 A(IND)=DSUM*DPIV
   11 IND=IND+I
      RETURN
   12 WRITE(6,101)K
  101 FORMAT(20X,'MATRIX IS SINGULAR AT ROW ',I2)
      RETURN
      END
```

```
      SUBROUTINE LTG2SP(K,N,A,DMIN,V)
C      LOWER TRIANGULAR G2 SWEEP OPERATOR AS DESCRIBED
C        BY GOODNIGHT - AMERICAN STATISTICIAN,
C        AUGUST 1979 VOL 33 NO. 3
C        WRITTEN BY K. E. KEMP FOR 285-725
      REAL*8 A(1),DMIN(N),D,B,C
      INTEGER V(N)
      KDIAG=(K*K+K)/2
      KK=KDIAG-K
CHECK FOR LINEAR DEPENDENCY
      IF(V(K).EQ.1 .AND. A(KDIAG).LE.DMIN(K))GO TO 1
      D= 1.0/A(KDIAG)
C        SWEEP ROWS 1 THRU N
      DO 3 I=1,N
      IF(I.EQ.K)GO TO 3
      II=(I*I-I)/2
      IF(I.LT.K)GO TO 10
      B=A(II+K)*D
      GO TO 11
   10 B=A(KK+I)*D*V(K)*V(I)
C        SWEEP COLUMNS UP TO DIAGONAL OF ROW I
   11 DO 4 J=1,I
      IF(J.EQ.K)GO TO 4
      IF(K.LT.J)GO TO 12
      C=A(KK+J)
      GO TO 13
   12 C=V(J)*V(K)*A((J*J-J)/2+K)
   13 A(II+J)=A(II+J)-B*C
    4 CONTINUE
    3 CONTINUE
C        PIVOT ON KTH ROW AND COLUMN
      DO 8 J=1,K
    8 A(KK+J)=A(KK+J)*D
      DO 9 I=K,N
      IK=(I*I-I)/2+K
    9 A(IK)=-A(IK)*D
      A(KDIAG)=D
C        SET INDICATOR VECTOR V
      V(K)=-V(K)
      RETURN
    1 WRITE(6,100)K
  100 FORMAT('-ROW',I3,'IS A LINEAR FUNCTION OF PREVIOUS'
     1,'ROWS IN MODEL')
C        ZERO ROW K
      DO 2 J=1,K
  2   A(KK+J)=0.0
      IF(K.EQ.N)GO TO 6
C        ZERO COLUMN K
      NM1=N-1
      IK=KDIAG
      DO 5 I=K,NM1
      IK=IK+I
  5   A(IK)=0.0
  6   RETURN
      END
```

```
      SUBROUTINE ERRINV(A,B,N,DERR,IPRINT)
      IMPLICIT REAL*8 (D)
      REAL*8 A(21,21),B(21,21),D(21,21)
C
C     THE AMOUNT OF THE ERROR OF THE COMPUTED INVERSE
C     --- THE ABSOLUTE SUM OF ELEMENTS OF
C     A*B-D WHERE B IS THE COMPUTED INVERSE OF A
C     AND D IS THE IDENTITY MATRIX
C
      IF(IPRINT.EQ.0) GO TO 8
      WRITE(6,5)
    5 FORMAT(// 1X,'PRODUCT OF XPX AND IXPX',//)
    8 DO 30 I=1,N
      DO 20 J=1,N
      D(I,J) = 0.0
      DO 10 K=1,N
      D(I,J) = D(I,J) + A(I,K)*B(K,J)
   10 CONTINUE
   20 CONTINUE
      IF(IPRINT.EQ.0) GO TO 30
      WRITE(6,25) (D(I,JJ),JJ=1,N)
   25 FORMAT(1X,10E13.4)
   30 CONTINUE
      DO 40 I=1,N
      DO 40 J=1,N
      IF(I.EQ.J) D(I,J)=D(I,J)-1.0
      DERR = DERR + DABS(D(I,J))
   40 CONTINUE
      RETURN
      END
```

APPENDIX B1

10 X 10 CORRELATION MATRICES WITH DIFFERENT CONDITION NUMBERS

ORIGINAL CORRELATION MATRIX (10X10)

1	1.0000						
2	0.3832	1.0000					
3	0.3056	0.9436	1.0000				
4	0.4743	-.1261	-.1437	1.0000			
5	0.1367	0.3821	0.2482	-.3168	1.0000		
6	0.5361	0.6851	0.7645	0.2311	0.0201	1.0000	
7	0.6407	-.1911	-.2264	0.5581	-.2048	0.1169	1.0000
8	-.8452	-.0019	0.0677	-.6163	0.0774	-.2098	-.8576
9	0.3945	-.1314	-.1342	0.9900	-.3210	0.2125	0.4915
10	0.3821	0.6163	0.6013	0.0739	-.0533	0.6006	0.1175

	1	2	3	4	5	6	7

8	1.0000		
9	-.5414	1.0000	
10	-.2370	0.0284	1.0000

	8	9	10

CONDITION NUMBER = 0.10000D 02

1	1.0000						
2	0.2682	1.0000					
3	0.2139	0.6605	1.0000				
4	0.3320	-.0883	-.1006	1.0000			
5	0.0957	0.2675	0.1738	-.2217	1.0000		
6	0.3753	0.4795	0.5351	0.1618	0.0141	1.0000	
7	0.4484	-.1338	-.1584	0.3906	-.1433	0.0818	1.0000
8	-.5916	-.0013	0.0474	-.4314	0.0542	-.1468	-.6003
9	0.2762	-.0919	-.0939	0.6929	-.2247	0.1487	0.3440
10	0.2675	0.4314	0.4209	0.0517	-.0373	0.4204	0.0823

	1	2	3	4	5	6	7

8	1.0000		
9	-.3790	1.0000	
10	-.1659	0.0199	1.0000

	8	9	10

CONDITION NUMBER = 0.10000D 04

1	1.0000						
2	0.3831	1.0000					
3	0.3055	0.9434	1.0000				
4	0.4742	-.1261	-.1436	1.0000			
5	0.1367	0.3820	0.2482	-.3167	1.0000		
6	0.5360	0.6849	0.7642	0.2311	0.0201	1.0000	
7	0.6405	-.1911	-.2263	0.5579	-.2047	0.1168	1.0000
8	-.8450	-.0019	0.0677	-.6162	0.0774	-.2097	-.8574
9	0.3944	-.1313	-.1342	0.9897	-.3209	0.2124	0.4914
10	0.3820	0.6161	0.6011	0.0739	-.0533	0.6004	0.1175

	1	2	3	4	5	6	7

8	1.0000		
9	-.5413	1.0000	
10	-.2369	0.0284	1.0000

	8	9	10

CONDITION NUMBER = 0.10000D 06

1	1.0000						
2	0.3846	1.0000					
3	0.3066	0.9470	1.0000				
4	0.4760	-.1265	-.1442	1.0000			
5	0.1372	0.3835	0.2491	-.3179	1.0000		
6	0.5380	0.6875	0.7672	0.2320	0.0202	1.0000	
7	0.6429	-.1918	-.2272	0.5601	-.2055	0.1173	1.0000
8	-.8483	-.0019	0.0680	-.6185	0.0777	-.2105	-.8607
9	0.3960	-.1318	-.1347	0.9935	-.3221	0.2132	0.4933
10	0.3835	0.6185	0.6035	0.0742	-.0535	0.6027	0.1179

	1	2	3	4	5	6	7

8	1.0000		
9	-.5434	1.0000	
10	-.2378	0.0285	1.0000

	8	9	10

CONDITION NUMBER = 0.10000D 09

1	1.0000						
2	0.3846	1.0000					
3	0.3067	0.9471	1.0000				
4	0.4760	-.1265	-.1442	1.0000			
5	0.1372	0.3835	0.2491	-.3179	1.0000		
6	0.5381	0.6876	0.7672	0.2320	0.0202	1.0000	
7	0.6430	-.1918	-.2272	0.5601	-.2055	0.1173	1.0000
8	-.8483	-.0019	0.0680	-.6186	0.0777	-.2105	-.8607
9	0.3960	-.1318	-.1347	0.9935	-.3222	0.2132	0.4933
10	0.3835	0.6185	0.6035	0.0742	-.0535	0.6028	0.1179

	1	2	3	4	5	6	7

8	1.0000		
9	-.5434	1.0000	
10	-.2378	0.0285	1.0000

	8	9	10

APPENDIX B2

20 X 20 CORRELATION MATRICES WITH DIFFERENT CONDITION NUMBERS

ORIGINAL CORRELATION MATRIX (20X20)

1	1.0000						
2	0.3832	1.0000					
3	0.3056	0.9436	1.0000				
4	0.4743	-.1261	-.1437	1.0000			
5	0.1367	0.3821	0.2482	-.3168	1.0000		
6	0.5361	0.6851	0.7645	0.2311	0.0201	1.0000	
7	0.6407	-.1911	-.2264	0.5581	-.2048	0.1169	1.0000
8	-.8452	-.0019	0.0677	-.6163	0.0774	-.2098	-.8576
9	0.3945	-.1314	-.1342	0.9900	-.3210	0.2125	0.4915
10	0.3821	0.6163	0.6013	0.0739	-.0533	0.6006	0.1175
11	0.3527	0.1252	0.4815	0.0914	0.2154	-.1574	0.0184
12	-.2418	0.0015	0.0924	-.2718	0.0015	-.4165	-.0092
13	-.0091	-.0718	0.5391	-.9356	-.1584	-.7147	0.0854
14	0.0854	-.3258	0.1258	0.6154	0.0159	0.2475	-.2451
15	0.2538	0.0017	-.8192	-.0018	0.0278	0.3124	0.1159
16	-.6352	0.0824	0.1456	0.0174	0.2471	-.5124	-.2417
17	0.0295	-.6719	0.4124	0.0914	-.5174	-.0058	0.4840
18	0.9153	-.2542	0.0180	-.2415	-.0170	0.0159	0.2154
19	-.8124	0.1798	0.1892	0.4714	0.0090	0.0680	-.1184
20	0.1736	0.3157	0.4718	0.2735	0.5481	0.2715	0.1514

1	2	3	4	5	6	7

8	1.0000						
9	-.5414	1.0000					
10	-.2370	0.0284	1.0000				
11	0.3696	0.2596	-.0724	1.0000			
12	-.2848	0.1741	0.0058	0.2330	1.0000		
13	-.0167	-.5371	-.9248	0.2884	0.1855	1.0000	
14	0.0581	-.4148	0.2746	-.0673	-.2505	0.3476	1.0000
15	-.0074	0.0059	-.6125	-.9592	-.1297	-.2426	0.0270
16	-.0185	0.0842	-.0073	0.6402	0.5047	0.3021	-.2306
17	0.6174	-.2606	0.1852	-.0831	-.0849	0.0348	0.1184
18	0.4145	-.1561	0.3524	-.1912	-.3600	0.1199	0.1074
19	0.6158	0.3475	0.0096	-.0418	-.0761	0.0156	0.0921
20	-.0254	-.2682	0.1142	-.1955	0.0880	0.0443	0.0950

8	9	10	11	12	13	14

15	1.0000					
16	-.4517	1.0000				
17	0.1027	-.0525	1.0000			
18	0.1734	-.1657	0.0836	1.0000		
19	0.0288	-.0448	0.0269	-.0021	1.0000	
20	0.1919	-.0573	0.0124	-.0529	-.0059	1.0000

15	16	17	18	19	20

CONDITION NUMBER = 0.10000D 02

1	1.0000						
2	0.1357	1.0000					
3	0.1082	0.3341	1.0000				
4	0.1679	-.0446	-.0509	1.0000			
5	0.0484	0.1353	0.0879	-.1122	1.0000		
6	0.1898	0.2426	0.2707	0.0818	0.0071	1.0000	
7	0.2268	-.0677	-.0801	0.1976	-.0725	0.0414	1.0000
8	-.2993	-.0007	0.0240	-.2182	0.0274	-.0743	-.3036
9	0.1397	-.0465	-.0475	0.3505	-.1136	0.0752	0.1740
10	0.1353	0.2182	0.2129	0.0262	-.0189	0.2126	0.0416
11	0.1249	0.0443	0.1705	0.0324	0.0763	-.0557	0.0065
12	-.0856	0.0005	0.0327	-.0962	0.0005	-.1475	-.0033
13	-.0032	-.0254	0.1909	-.3312	-.0561	-.2530	0.0302
14	0.0302	-.1153	0.0445	0.2179	0.0056	0.0876	-.0868
15	0.0899	0.0006	-.2900	-.0006	0.0098	0.1106	0.0410
16	-.2249	0.0292	0.0515	0.0062	0.0875	-.1814	-.0856
17	0.0104	-.2379	0.1460	0.0324	-.1832	-.0021	0.1714
18	0.3241	-.0900	0.0064	-.0855	-.0060	0.0056	0.0763
19	-.2876	0.0637	0.0670	0.1669	0.0032	0.0241	-.0419
20	0.0615	0.1118	0.1670	0.0968	0.1941	0.0961	0.0536
	1	2	3	4	5	6	7

8	1.0000						
9	-.1917	1.0000					
10	-.0839	0.0101	1.0000				
11	0.1309	0.0919	-.0256	1.0000			
12	-.1008	0.0616	0.0021	0.0825	1.0000		
13	-.0059	-.1902	-.3274	0.1021	0.0657	1.0000	
14	0.0206	-.1469	0.0972	-.0238	-.0887	0.1231	1.0000
15	-.0026	0.0021	-.2169	-.3396	-.0459	-.0859	0.0096
16	-.0065	0.0298	-.0026	0.2267	0.1787	0.1070	-.0816
17	0.2186	-.0923	0.0656	-.0294	-.0301	0.0123	0.0419
18	0.1468	-.0553	0.1248	-.0677	-.1275	0.0425	0.0380
19	0.2180	0.1230	0.0034	-.0148	-.0269	0.0055	0.0326
20	-.0090	-.0950	0.0404	-.0692	0.0312	0.0157	0.0336
	8	9	10	11	12	13	14

15	1.0000					
16	-.1599	1.0000				
17	0.0364	-.0186	1.0000			
18	0.0614	-.0587	0.0296	1.0000		
19	0.0102	-.0159	0.0095	-.0007	1.0000	
20	0.0679	-.0203	0.0044	-.0187	-.0021	1.0000
	15	16	17	18	19	20

CONDITION NUMBER = 0.10000D 04

1	1.0000						
2	0.1737	1.0000					
3	0.1385	0.4279	1.0000				
4	0.2151	-.0572	-.0651	1.0000			
5	0.0620	0.1733	0.1126	-.1436	1.0000		
6	0.2431	0.3106	0.3466	0.1048	0.0091	1.0000	
7	0.2905	-.0867	-.1026	0.2531	-.0928	0.0530	1.0000
8	-.3832	-.0009	0.0307	-.2795	0.0351	-.0951	-.3889
9	0.1789	-.0596	-.0608	0.4489	-.1455	0.0963	0.2229
10	0.1733	0.2794	0.2726	0.0335	-.0242	0.2723	0.0533
11	0.1599	0.0568	0.2183	0.0414	0.0977	-.0714	0.0083
12	-.1096	0.0007	0.0419	-.1232	0.0007	-.1888	-.0042
13	-.0041	-.0326	0.2444	-.4242	-.0718	-.3241	0.0387
14	0.0387	-.1477	0.0570	0.2790	0.0072	0.1122	-.1111
15	0.1151	0.0008	-.3714	-.0008	0.0126	0.1416	0.0526
16	-.2880	0.0374	0.0660	0.0079	0.1120	-.2323	-.1096
17	0.0134	-.3047	0.1870	0.0414	-.2346	-.0026	0.2195
18	0.4150	-.1153	0.0082	-.1095	-.0077	0.0072	0.0977
19	-.3684	0.0815	0.0858	0.2137	0.0041	0.0308	-.0537
20	0.0787	0.1431	0.2139	0.1240	0.2485	0.1231	0.0686

1	2	3	4	5	6	7

8	1.0000						
9	-.2455	1.0000					
10	-.1074	0.0129	1.0000				
11	0.1676	0.1177	-.0328	1.0000			
12	-.1291	0.0789	0.0026	0.1056	1.0000		
13	-.0076	-.2435	-.4193	0.1308	0.0841	1.0000	
14	0.0263	-.1881	0.1245	-.0305	-.1136	0.1576	1.0000
15	-.0034	0.0027	-.2777	-.4349	-.0588	-.1100	0.0122
16	-.0084	0.0382	-.0033	0.2903	0.2288	0.1370	-.1046
17	0.2799	-.1182	0.0840	-.0377	-.0385	0.0158	0.0537
18	0.1879	-.0708	0.1598	-.0867	-.1632	0.0544	0.0487
19	0.2792	0.1576	0.0044	-.0190	-.0345	0.0071	0.0418
20	-.0115	-.1216	0.0518	-.0886	0.0399	0.0201	0.0431

8	9	10	11	12	13	14

15	1.0000					
16	-.2048	1.0000				
17	0.0466	-.0238	1.0000			
18	0.0786	-.0751	0.0379	1.0000		
19	0.0131	-.0203	0.0122	-.0010	1.0000	
20	0.0870	-.0260	0.0056	-.0240	-.0027	1.0000

15	16	17	18	19	20

CONDITION NUMBER = 0.10000D 06

1	1.0000						
2	0.1742	1.0000					
3	0.1389	0.4289	1.0000				
4	0.2156	-.0573	-.0653	1.0000			
5	0.0622	0.1737	0.1128	-.1440	1.0000		
6	0.2437	0.3114	0.3475	0.1051	0.0091	1.0000	
7	0.2912	-.0869	-.1029	0.2537	-.0931	0.0531	1.0000
8	-.3842	-.0009	0.0308	-.2802	0.0352	-.0954	-.3898
9	0.1793	-.0597	-.0610	0.4500	-.1459	0.0966	0.2234
10	0.1737	0.2802	0.2733	0.0336	-.0242	0.2730	0.0534
11	0.1603	0.0569	0.2189	0.0415	0.0979	-.0715	0.0084
12	-.1099	0.0007	0.0420	-.1236	0.0007	-.1893	-.0042
13	-.0041	-.0326	0.2451	-.4253	-.0720	-.3249	0.0388
14	0.0388	-.1481	0.0572	0.2797	0.0072	0.1125	-.1114
15	0.1154	0.0008	-.3724	-.0008	0.0126	0.1420	0.0527
16	-.2887	0.0375	0.0662	0.0079	0.1123	-.2329	-.1099
17	0.0134	-.3054	0.1875	0.0415	-.2352	-.0026	0.2200
18	0.4161	-.1156	0.0082	-.1098	-.0077	0.0072	0.0979
19	-.3693	0.0817	0.0860	0.2143	0.0041	0.0309	-.0538
20	0.0789	0.1435	0.2145	0.1243	0.2491	0.1234	0.0688
	1	2	3	4	5	6	7

8	1.0000						
9	-.2461	1.0000					
10	-.1077	0.0129	1.0000				
11	0.1680	0.1180	-.0329	1.0000			
12	-.1295	0.0791	0.0026	0.1059	1.0000		
13	-.0076	-.2441	-.4204	0.1311	0.0843	1.0000	
14	0.0264	-.1886	0.1248	-.0306	-.1139	0.1580	1.0000
15	-.0034	0.0027	-.2784	-.4360	-.0590	-.1103	0.0123
16	-.0084	0.0383	-.0033	0.2910	0.2294	0.1373	-.1048
17	0.2807	-.1185	0.0842	-.0378	-.0386	0.0158	0.0538
18	0.1884	-.0710	0.1602	-.0869	-.1636	0.0545	0.0488
19	0.2799	0.1580	0.0044	-.0190	-.0346	0.0071	0.0419
20	-.0115	-.1219	0.0519	-.0889	0.0400	0.0201	0.0432
	8	9	10	11	12	13	14

15	1.0000					
16	-.2053	1.0000				
17	0.0467	-.0239	1.0000			
18	0.0788	-.0753	0.0380	1.0000		
19	0.0131	-.0204	0.0122	-.0010	1.0000	
20	0.0872	-.0260	0.0056	-.0240	-.0027	1.0000
	15	16	17	18	19	20

CONDITION NUMBER = 0.10000D 09

1	1.0000						
2	0.1742	1.0000					
3	0.1389	0.4290	1.0000				
4	0.2156	-.0573	-.0653	1.0000			
5	0.0622	0.1737	0.1128	-.1440	1.0000		
6	0.2437	0.3114	0.3475	0.1051	0.0091	1.0000	
7	0.2912	-.0869	-.1029	0.2537	-.0931	0.0531	1.0000
8	-.3842	-.0009	0.0308	-.2802	0.0352	-.0954	-.3899
9	0.1793	-.0597	-.0610	0.4500	-.1459	0.0966	0.2234
10	0.1737	0.2802	0.2733	0.0336	-.0242	0.2730	0.0534
11	0.1603	0.0569	0.2189	0.0415	0.0979	-.0716	0.0084
12	-.1099	0.0007	0.0420	-.1236	0.0007	-.1893	-.0042
13	-.0041	-.0326	0.2451	-.4253	-.0720	-.3249	0.0388
14	0.0388	-.1481	0.0572	0.2797	0.0072	0.1125	-.1114
15	0.1154	0.0008	-.3724	-.0008	0.0126	0.1420	0.0527
16	-.2887	0.0375	0.0662	0.0079	0.1123	-.2329	-.1099
17	0.0134	-.3054	0.1875	0.0415	-.2352	-.0026	0.2200
18	0.4161	-.1156	0.0082	-.1098	-.0077	0.0072	0.0979
19	-.3693	0.0817	0.0860	0.2143	0.0041	0.0309	-.0538
20	0.0789	0.1435	0.2145	0.1243	0.2492	0.1234	0.0688
	1	2	3	4	5	6	7

8	1.0000						
9	-.2461	1.0000					
10	-.1077	0.0129	1.0000				
11	0.1680	0.1180	-.0329	1.0000			
12	-.1295	0.0791	0.0026	0.1059	1.0000		
13	-.0076	-.2442	-.4204	0.1311	0.0843	1.0000	
14	0.0264	-.1886	0.1248	-.0306	-.1139	0.1580	1.0000
15	-.0034	0.0027	-.2784	-.4360	-.0590	-.1103	0.0123
16	-.0084	0.0383	-.0033	0.2910	0.2294	0.1373	-.1048
17	0.2807	-.1185	0.0842	-.0378	-.0386	0.0158	0.0538
18	0.1884	-.0710	0.1602	-.0869	-.1636	0.0545	0.0488
19	0.2799	0.1580	0.0044	-.0190	-.0346	0.0071	0.0419
20	-.0115	-.1219	0.0519	-.0889	0.0400	0.0201	0.0432
	8	9	10	11	12	13	14

15	1.0000					
16	-.2053	1.0000				
17	0.0467	-.0239	1.0000			
18	0.0788	-.0753	0.0380	1.0000		
19	0.0131	-.0204	0.0122	-.0010	1.0000	
20	0.0872	-.0260	0.0056	-.0240	-.0027	1.0000
	15	16	17	18	19	20

A COMPARISON OF ALGORITHMS FOR LEAST SQUARES ESTIMATES OF PARAMETERS IN THE LINEAR MODEL

by

CHUL H. AHN

B.S., Seoul National University, 1976

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Statistics

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1981

This report compared three algorithms for least squares estimates in the linear model : Gaussian elimination, Cholesky method, and the Sweep operator.

The comparison criteria were :

1. the number of arithmetic operations each algorithm requires
2. empirical data(accuracy and execution time).

As for the number of arithmetic operations Cholesky method requires much fewer operations than others. For accuracy comparison based on empirical data we concentrated on the accuracy of the computed inverse of X'X which would mainly effect the accuracy of the solution vector estimates because we couldn't find the exact solution to the solution vector. We generated X'X matrices from well-conditioned to ill-conditioned(condition number : $10 - 10^9$), and computed their inverse using the three algorithms.

To generate X'X matrix we first generated data matrix given the distribution and the correlation matrix.

Results showed that Cholesky method was best for both accuracy and execution time.