General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

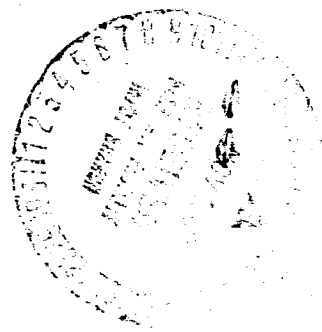
X-551-69-422
PREPRINT

NASA TM X-63722

AN EXPERIMENTAL COMPARISON OF SEVERAL APPROACHES TO THE LINEAR LEAST SQUARES PROBLEM

EUGENE J. LEFFERTS
DANIEL P. MUHONEN

SEPTEMBER 1969



GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

X-551-69-422
PREPRINT

AN EXPERIMENTAL COMPARISON OF SEVERAL APPROACHES TO THE LINEAR LEAST SQUARES PROBLEM

Eugene J. Lefferts
Daniel P. Muhonen*

September 1969

*Bendix Field Engineering Corporation

GODDARD SPACE FLIGHT CENTER
Greenbelt, Maryland

AN EXPERIMENTAL COMPARISON OF SEVERAL APPROACHES TO THE LINEAR LEAST SQUARES PROBLEM

Eugene J. Lefferts
Daniel P. Muhonen

ABSTRACT

Several computational algorithms for obtaining least square solutions to a system of equations are made. The algorithms include both conventional and pseudo inversion methods. Comparisons of running time, computer storage and accuracy of recovery are made for each algorithm in both single and double precision.

AN EXPERIMENTAL COMPARISON OF SEVERAL APPROACHES
TO THE LINEAR LEAST SQUARES PROBLEM

SUMMARY

Experiments were made using five procedures to solve the linear least squares problem. These were:

(1) The Gauss-Jordan algorithm on the normal equations

(2) An Andree algorithm for the Penrose pseudoinverse on the normal equations

(3) A Gram-Schmidt orthogonalization pseudoinversion scheme on the normal equations

(4) The Gram-Schmidt procedure applied to the original rectangular data matrix

(5) Householder's algorithm for direct triangulation of the original data matrix

The experiments involved the recovery of polynomial coefficients in both double and single precision. Computer running time and storage requirements are presented. The following conclusions and recommendations can be made as a result of these experiments:

- Procedures (4) and (5) above, which avoid the formation of the normal equations, give considerably better results than do the other schemes.

- Procedure (4) should be used in double precision when there are no storage or timing restrictions. It will handle situations of reduced rank.

- Procedure (5) should be used in double precision when timing but not storage restrictions exist. It does not handle situations of reduced rank.

- Procedures (1), (2), and (3) give better results in double precision than do procedures (4) and (5) in single precision.

- Procedure (1) is most efficient in running time and storage requirements, but it should be avoided whenever possible because it cannot handle situations of reduced computational rank.

- Procedure (2) gives somewhat better results than procedure (3), but it requires much more storage.

CONTENTS

ILLUSTRATIONS

AN EXPERIMENTAL COMPARISON OF SEVERAL APPROACHES TO THE LINEAR LEAST SQUARES PROBLEM

INTRODUCTION

In solving the linear least squares problem, complications arise due to the introduction of errors in the computation. The magnitudes of these errors vary considerably with the procedures used to obtain a least squares solution. One difficulty that frequently arises is that, because of accumulated truncation error in the computation, only a few decimal figures in the original data matrix are of significance in the least squares process. Hence, if only the computationally significant portions of the matrix elements are considered, a matrix of maximal rank may become one of reduced rank. Unless the computational errors are accounted for in some way, a meaningless solution can result.

Five approaches to the least squares problem were investigated experimentally. Using the normal equations, comparisons were made between the Gauss-Jordan inversion algorithm and two pseudoinversion schemes: an Andree algorithm and a Gram-Schmidt orthogonalization procedure. The advantage of implementing a pseudoinverse is that the minimal norm solution can be found when the matrix is not of maximal rank. Both of these pseudoinverse algorithms use a-priori truncation error limits to determine the effective rank of the matrix. In addition, experiments were made using two approaches which arrive at a solution directly from the rectangular data matrix without forming the normal equations. The Gram-Schmidt pseudoinversion scheme is compared here with the Householder algorithm which forms the Cholesky decomposition of $A^T A$ directly from A. The effect of truncation error resulting from the formation of the normal equations is quite evident from these experiments. Computer running time and storage information are also presented in order that a more complete evaluation can be made of the advantages and disadvantages of all procedures.

PROBLEM STATEMENT

This presentation will be concerned with the linear system

$$A\vec{X} = \vec{B},$$

where A is a known m x n matrix, $\vec{B}$ is a known m x 1 vector, and $\vec{X}$ is an unknown solution vector n x 1. The system has a solution if and only if $\vec{B}$ is in the column space of A. If m > n, as is the case in many physical situations, this condition will generally not be satisfied. However, it is still possible to find a solution that is "best" in the least squares sense; that is, one can find a vector $\vec{X}_0$ such that

$$|| A\vec{X}_0 - \vec{B} || = \min.,$$

where $|| \; ||$ represents the vector Euclidean norm. If A is of maximal column rank (i.e., rank (A) = n), then the solution $\vec{X}_0$ is unique, and

$$\vec{X}_0 = (A^T A)^{-1} A^T \vec{B} .$$

This matrix equation represents what is commonly known as the normal equations.

The solution $\vec{X}_0$ in the above equation can be found by actually evaluating the square, nonsingular matrix $A^T A$ and finding its inverse by some standard algorithm. There are computational disadvantages in forming the normal equations, however, since information may be lost by truncation in evaluating the inner products which form the elements of $A^T A$ (ref. 1). There is an alternative approach developed by Householder which can be used to find the solution vector $\vec{X}_0$ without requiring the formation of $A^T A$ (see Appendix C). In solving a system of n linear equations in n unknowns, a common procedure is to reduce the matrix of coefficients to upper triangular form by a series of elementary row operations (ref. 2). If these transformations are performed simultaneously on the constant vector $A^T \vec{B}$, the solution vector can then be found by back substitution. In our problem this can be represented by the following equation:

$$P A^T A \vec{X}_0 = P A^T \vec{B} ,$$

where P represents the product of the elementary transformation matrices necessary to reduce $A^T A$ to upper triangular form; that is, $PA^T A$ is an upper triangular matrix. Householder's scheme provides for the evaluation of $PA^T A$ and $PA^T \vec{B}$ directly from A and $\vec{B}$ without requiring the formation of the inner products which can result in serious truncation error.

The problem still remains that the matrix A may not be of maximal rank, in which case the normal equations will not have a unique solution. From a computational point of view, this is a very real problem in many physical situations. Certainly if the level of noise is sufficiently high in the data represented by two or more columns in the matrix A, very few figures of significance exist in this data. What significance there is can easily be lost in the least squares solution processes. It may also be true that a near-linear dependence does exist in the observables represented by two or more columns. Unless some means is provided to allow for these errors, a solution may result having no physical meaning whatsoever. One approach is to reduce the dimension of the problem so that $A^T A$ is computationally nonsingular. If a triangular decomposition of $A^T A$ is formed such that row operations are done in a selective order, this can be accomplished by examining the relative magnitudes of resulting diagonal elements as they are formed. This comparison is made relative to some a-priori significance level which allows for errors both in the data and in the computation. If the test fails, the column under investigation is eliminated and the corresponding element of the solution vector $\vec{X}_0$ is assumed to be zero.

The above procedure will result in a least squares solution to the linear system, but it has the disadvantage that observables are eliminated from the system, and full use is not made of the information available. The solution obtained is also rather arbitrary in that the elements of $\vec{X}_0$ which are assumed to be zero are determined by the order in which the reduction takes place. One way of circumventing these objections is to solve for the solution vector which has minimal Euclidian norm.

The problem can then be stated as follows:

$$||A\vec{X}_0 - \vec{B}|| = \min.,$$

$$||\vec{X}_0|| = \min .$$

The solution $\vec{X}_0$ exists and is unique regardless of whether A is of maximal rank.

One way of obtaining the solution $\vec{X}_0$ is to make use of the Penrose pseudoinverse (see Appendix A), which has the following definition: For any rectangular matrix A, let $A^{\#}$ denote the Penrose pseudoinverse of A. It can be shown that $A^{\#}$ is well defined if it satisfies the following four axioms:

$$\text{(i)} \quad AA^{\#}A = A$$

$$\text{(ii)} \quad A^{\#}AA^{\#} = A^{\#}$$

$$\text{(iii)} \quad (A^{\#}A)^T = A^{\#}A$$

$$\text{(iv)} \quad (AA^{\#})^T = AA^{\#} .$$

The axioms imply that:

$$A^{\#\#} = A ,$$

$$A^{\#} = A^{-1} \text{ if } A^{-1} \text{ exists} ,$$

$$(A^T)^{\#} = (A^{\#})^T .$$

If we now consider the original linear system

$$A\vec{X} = \vec{B} ,$$

and if we let

$$\vec{X}_0 = A^{\#}\vec{B} ,$$

then it follows that, for any vector $\vec{X}$ of dimension n ,

$$||A\vec{X}_0 - \vec{B}|| \leq ||A\vec{X} - \vec{B}|| .$$

Furthermore, if

$$||A\vec{X}_0 - \vec{B}|| = ||A\vec{X} - \vec{B}|| ,$$

then

$$||\vec{X}_0|| \leq ||\vec{X}|| .$$

Hence, the Penrose pseudoinverse will provide the least squares solution with minimal Euclidean norm.

LEAST SQUARES PROCEDURES INVESTIGATED

Computational experiments were performed on the following five approaches to obtain a least squares solution. Three decimal digits of computational error were allowed for in the procedures that tested for reduction in computational rank.

(1) A Gauss-Jordan matrix inversion algorithm applied to the normal equations (ref. 2). No check was made for computational singularity of A^TA. The solution $\vec{X}_0$ was evaluated as follows:

$$\vec{X}_0 = (A^TA)^{-1}(A^T\vec{B}) \ .$$

(2) An Andree pseudoinversion algorithm applied to the normal equations (see Appendixes B and C). Here a series of congruence transformations were made on the A^TA matrix such that in each step the reduction was made to pivot about the largest remaining diagonal element. A computational reduction in rank was performed under the following test:

Let α_1 be the first pivot element and α_i be a pivot element being tested. Then reduction in rank was performed when

$$\alpha_i < 10^3 \left(\frac{\alpha_1}{10^k}\right) \ ,$$

where k is the number of decimal figures carried in a computer word. Rank reduction was effected by setting α_i and all subsequent pivot elements to zero. The solution $\vec{X}_0$ was evaluated as follows:

$$\vec{X}_0 = (A^TA)^{\#}(A^T\vec{B}) \ .$$

(3) A Gram-Schmidt orthoganalization procedure for evaluating the pseudoinverse, applied to the normal equations (see Appendixes B and E). Under this scheme an orthonormal basis for the column space of A^TA was evaluated, and rank reduction was performed under the following test:

Let $||\vec{\alpha}_j||$ represent the magnitude of a column vector of A^TA being tested and let $||\vec{\beta}_j||$ be the magnitude of its projection on the independent column space already determined. Then reduction in rank was performed when

$$||\vec{\alpha}_j - \vec{\beta}_j|| < 10^3 \left(\frac{||\vec{\alpha}_j||}{10^k}\right) \ ,$$

where k is the number of decimal figures carried in a computer word. The solution $\vec{X}_0$ was evaluated as follows:

$$\vec{X}_0 = (A^T A)^{\#} (A^T \vec{B}) .$$

(4) A Gram-Schmidt orthoganalization procedure for evaluating the pseudoinverse, applied to the rectangular m x n matrix A. The same routine as in (3) was used, but operations were done on the columns of A rather than $A^T A$. The solution $\vec{X}_0$ was evaluated as follows:

$$\vec{X}_0 = A^{\#} \vec{B} .$$

(5) The Householder procedure for evaluating the least squares solution by finding the triangular decomposition of $A^T A$ directly from A (see Appendixes C and F). Under this scheme both A and $\vec{B}$ are pre-multiplied by an orthogonal matrix Q which transforms A into upper triangular form, with all elements below the main diagonal equal to zero; that is:

$$QA = R = \begin{pmatrix} \tilde{R} \\ \cdots \\ 0 \end{pmatrix} ,$$

$$Q\vec{B} = \vec{C} = \begin{pmatrix} \vec{C}_1 \\ \vec{C}_2 \end{pmatrix} ,$$

where $\tilde{R}$ is an n x n upper triangular matrix and $\vec{C}_1$ represents the first n elements of $Q\vec{B}$. Since the Euclidean norm is preserved under an orthogonal transformation,

$$||A\vec{X} - \vec{B}|| = ||R\vec{X} - \vec{C}|| ,$$

and the least squares solution is clearly

$$\vec{X}_0 = \tilde{R}^{-1} \vec{C}_1 ,$$

provided that $\tilde{R}^{-1}$ exists. The triangulation of A was performed by operations on the columns of A in such an order that the diagonal elements formed were maximized. Since $\tilde{R}$ is the triangular decomposition of the positive definite matrix $A^T A$, a test was made for computational reduction in rank by examining the relative magnitudes of the diagonal elements of $\tilde{R}$. Let r_1 be the first diagonal element formed and let r_i represent a diagonal element being tested. Then $\tilde{R}$, and hence A, was assumed to be of reduced rank when

$$r_i < 10^3 \left(\frac{r_1}{10^k} \right) ,$$

where k is the number of decimal figures carried in a computer word.

If rank was not maximal, no solution was evaluated. With maximal rank, $\vec{X}_0$ was evaluated by back substitution from the equation

$$\tilde{R}\,\vec{X}_0 = \vec{C}_1 \ .$$

COMPUTATIONAL EXPERIMENTS

In all of the experiments performed, attempts were made to recover the coefficients for the polynomial $Z^2 + 10Z + 1$. Matrices were generated for situations of varying dimension and original data error. For a problem of dimension n, with m data points, the matrix $A = (a_{ij})$ and the vector $\vec{B} = (b_i)$ were generated from a starting Z_1 and ΔZ in the following manner:

$$Z_i = Z_{i-1} + \Delta Z \ , \ i = 2, \cdots, m$$

$$b_i = 1 + 10\,Z_i + Z_i^2 \ , \ i = 1, \cdots, m$$

$$a_{ij} = Z_i^{j-1} \ , \ j = 1, \cdots, n$$

$$i = 1, \cdots, m \ .$$

If there were no computational errors involved, then the solution $\vec{X}_0^{\,T} = (1, 10, 1, 0, 0, \cdots, 0)$ would clearly be the least squares solution, since

$$||A\vec{X}_0 - \vec{B}|| = 0 \ .$$

Figures 1-a and 1-b represent the results with the matrices generated from $Z_1 = -1$, $\Delta Z = 1/16$, and $m = 33$. The column dimension of A was varied from 5 to 25. Single precision (24 binary digits) was used in the generation of matrices A and $\vec{B}$; since each Z_i can be exactly represented by four binary digits, no error was present in the A matrix up to column dimension 7. Results are presented with the least squares solution processes, including the formation of the normal equations, carried out both in single and double precision. In both instances matrices A and $\vec{B}$ were generated in single precision. Figure 1-a shows the common logarithm of the norm of the solution error vector ($\log_{10} ||\vec{X}_0^{\,T} - (1, 10, 1, 0, \cdots, 0)||$) plotted against the column dimension of the A matrix. About 16 decimal digits are carried on the computer in double precision, and it is clear from the graph that little significance was lost up to column dimension 7 for the two double precision procedures which avoid the formation of the normal equations. Figure 1-b represents the computational rank as determined by the various algorithms tested. The three routines which make use of the normal equations gave similar results in recovery until rank suppression occurred. In general, better recovery was observed with the Andree and Gram - Schmidt routines when rank was suppressed. Double precision arithmetic gave much better results up to dimension 20, at which point computational error became very high with the normal equations. It is evident that much better

results can be expected at high dimensions when the normal equations are avoided.

Figures 2-a and 2-b demonstrate the results of data generated with high original error. Here matrices A and $\vec{B}$ were generated in single precision with row dimension $m = 100$, $Z_1 = 0.01$, and $\Delta Z = 0.01$. Double precision was used to obtain the least squares solutions. Most Z_i generated here do not have exact binary representation, and a high degree of error was present in the higher powers of Z_i. Up to dimension 9, all procedures gave nearly the same results because the original data error in the A and $\vec{B}$ matrices was much greater than the computational error which occurred in the least squares soultions. Beyond dimension 9, the Andree algorithm applied to the normal equations gave the best recovery because there was greater suppression of computational rank. The Gram-Schmidt algorithm gave a different computational rank because non-negativeness was not forced in the $(A^TA)^{\#}$ matrix as it was with the Andree algorithm. Throughout the entire dimension range, both schemes which avoid the normal equations gave essentially the same results because original data error predominated. The results applying single precision to obtain the least squares solutions from these data matrices are shown in figures 3-a and 3-b. At high dimensions both single precision and double precision tended to give equally poor results, which further demonstrates the predominance of original data error.

In figures 4-a and 4-b the same data matrices were used, but they were generated in double precision rather than in single precision. A marked improvement in recovery is clearly evident, and the advantage of avoiding the normal equations is again demonstrated. The superiority of using the pseudoinverse with reduced computational rank when the normal equations are used is best shown in this example. Above dimension 11 the Andree algorithm gave the best recovery with the normal equations, and at very high dimensions results were comparable with the schemes which operate directly with the A matrix. However, at the high dimension level, computational noise became very great with all algorithms, so the comparison has little meaning at this point.

COMPUTER RUNNING TIME

Figures 5 and 6 show comparisons of computer running time in double and single precision for all schemes tested. The A matrix here is the one reflected in figures 2-a, 2-b, 3-a, and 3-b; that is, it was generated in single precision with $m = 100$, $Z_1 = 0.01$, and $\Delta Z = 0.01$. On the IBM 360/95 computer, which was used in all experiments, the minimum measurable time interval is 0.01664 seconds, so comparisons could not be made when the running time was less than this. Only the time required for the matrix inversion or pseudoinversion is reflected in these graphs, except for the Householder algorithm which does not evaluate an inverse. The back substitution time for obtaining the solution vector is included with the Householder scheme.

In general, the Gauss-Jordan algorithm was the most economical in running time, and the Gram-Schmidt procedure applied to the A matrix required the most computer time. One can expect an even greater difference in running time if the A matrix has a larger row dimension. Observe that on the IBM 360/95 computer running time is essentially the same in both double precision and single precision where maximal rank is maintained. The Gram-Schmidt algorithm requires more time when there is reduction in rank, so a greater running time is shown for this

routine at the higher dimensions on the single precision plot. However, less time is required for the Andree algorithm with reduction in rank, so this routine appears more efficient in running time on the single precision graph at higher dimensions. The Householder algorithm also showed less running time with reduced rank because the solution was not completed.

COMPUTER STORAGE REQUIREMENTS

Storage restrictions may be a factor in choosing the subroutine one wishes to use in the least squares solution. All subroutines used in this study destroy the original data matrix, A or A^TA, whichever applies. The table below lists the IBM 360/95 storage requirements for all subroutines tested. The original matrix (A or A^TA) plus working arrays required are included in this list; with the the Householder algorithm the constant vector $\vec{B}$ is also included since it too is destroyed. The matrix A is assumed to be of dimension m x n. (Assume m = n for the Gram-Schmidt algorithm applied to the normal equations.)

If m is large, one may be forced to store only the A^TA matrix and operate with the normal equations. Note that the Andree algorithm requires a large amount of working storage.

Algorithm	Double Precision Storage Requirements (Bytes)	Single Precision Storage Requirements (Bytes)
Gauss-Jordan	$1876 + 8\ (n^2 + n + \frac{3n}{2})$	$1795 + 4\ (n^2 + 4n)$
Andree	$4538 + 8\ (6n^2 + n)$	$4472 + 4\ (6n^2 + n)$
Gram-Schmidt	$3304 + 8\ (mn + n^2 + 2n)$	$3260 + 4\ (mn + n^2 + 2n)$
Householder	$3190 + 8\ (mn + m + 2n + \frac{n}{2})$	$3150 + 4\ (mn + m + 3n)$

CONCLUSIONS

It is clear from these experiments that it best whenever possible to avoid formation of the normal equations in solving the linear least squares problem. However, storage restrictions may not allow the entire data matrix to be placed in core. In this case, some advantage is gained by making use of a pseudoinversion subroutine which allows for reduced computational rank. Of the two pseudoinversion schemes investigated, the Andree algorithm gave the best recovery when the normal equations were employed. This routine requires a great deal of temporary storage, however, and the Gram-Schmidt subroutine may be a necessary substitute. In some situations a Gauss-Jordan subroutine must be used if storage limitations are critical.

Of the two procedures which operate directly on the original data matrix, the Householder algorithm is more efficient in both computer running time and storage requirements. However, it does not adequately handle situations of reduced computational rank. The Householder and the Gram-Schmidt routines gave comparable results in recovery.

There was a highly significant improvement in recovery when double precision rather than single precision was employed. This was true both with the least squares solution procedures and with the generation of the data matrices. Use of the normal equations with double precision generally gave better results than did the Gram-Schmidt and Householder schemes applied to the original rectangular data matrix in single precision.

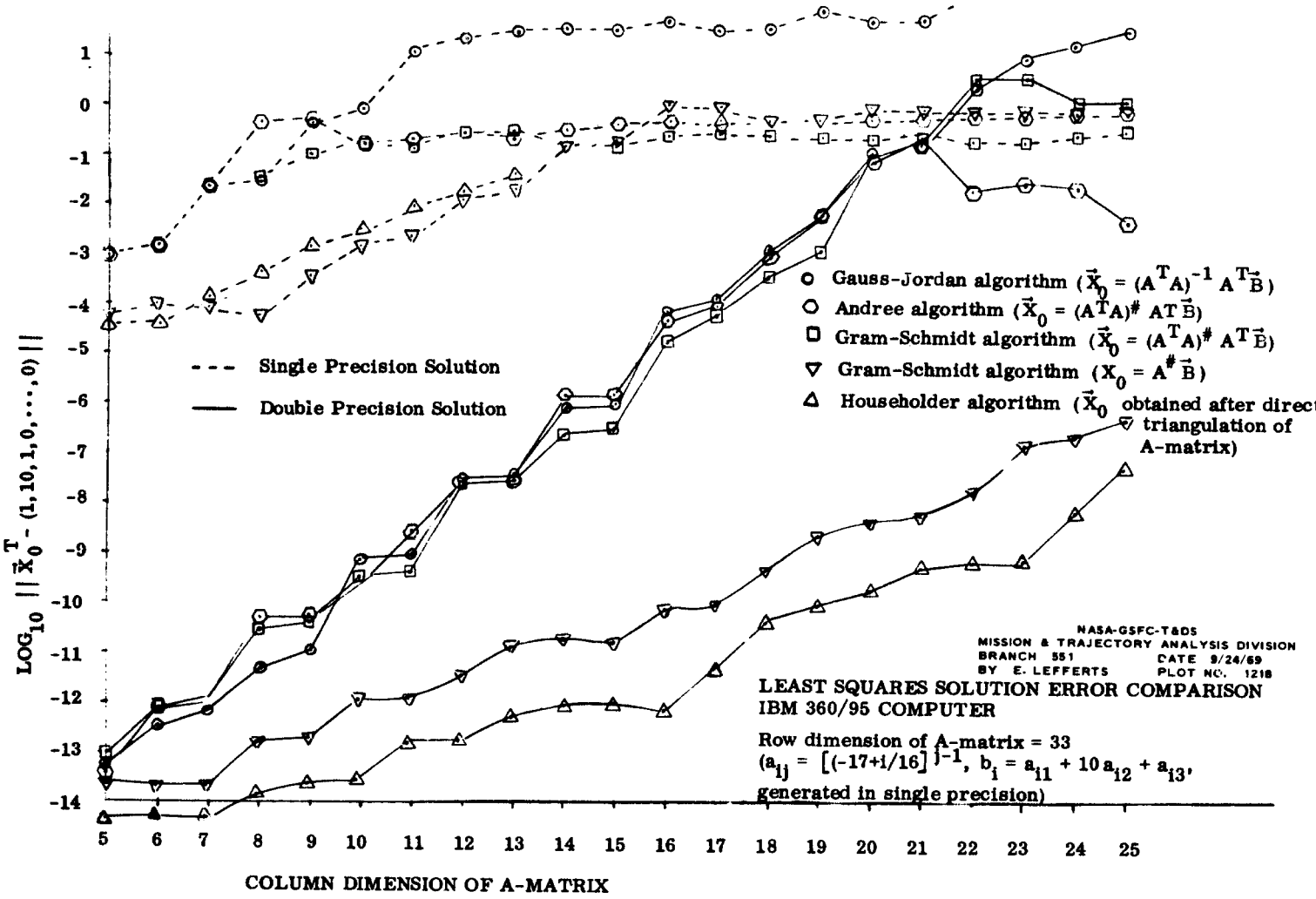

Single Precision Solution
Double Precision Solution
Gauss-Jordan algorithm ($\vec{X}_0 = (A^TA)^{-1} A^T\vec{B}$)
Andree algorithm ($\vec{X}_0 = (A^TA)^{\#} A^T\vec{B}$)
Gram-Schmidt algorithm ($\vec{X}_0 = (A^TA)^{\#} A^T\vec{B}$)
Gram-Schmidt algorithm ($X_0 = A^{\#}\vec{B}$)
Householder algorithm ($\vec{X}_0$ obtained after direct triangulation of A-matrix)
NASA-GSFC-T&DS
MISSION & TRAJECTORY ANALYSIS DIVISION
BRANCH 551
DATE 9/24/69
BY E. LEFFERTS
PLOT NO. 1218
LEAST SQUARES SOLUTION ERROR COMPARISON
IBM 360/95 COMPUTER
Row dimension of A-matrix = 33
($a_{ij} = [(-17+i/16]^{j-1}$, $b_i = a_{11} + 10\,a_{12} + a_{13}$, generated in single precision)
$LOG_{10} ||\vec{X}_0^T - (1,10,1,0,\cdots,0)||$
COLUMN DIMENSION OF A-MATRIX

Figure 1-a

COMPUTATIONAL RANK COMPARISON FOR DOUBLE AND SINGLE PRECISION MATRIX INVERSION SUBROUTINES. IBM 360/95 COMPUTER

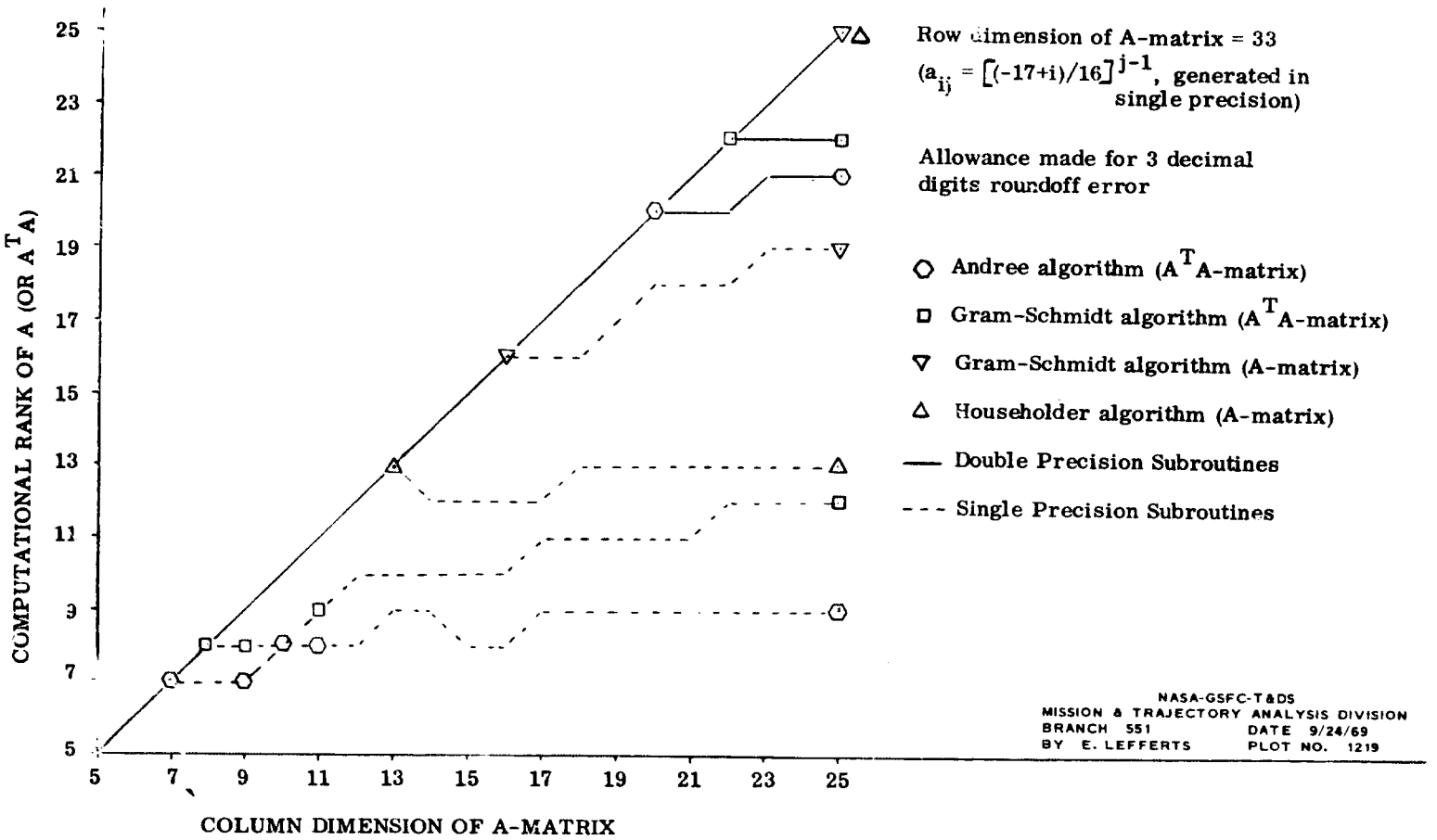


Figure 1-b

LEAST SQUARES SOLUTION ERROR COMPARISON USING DOUBLE PRECISION MATRIX INVERSION SUBROUTINES. IBM 360/95 COMPUTER.

Row dimension of A-matrix = 100

($a_{ij}=(i/100)^{j-1}$, $b_i = a_{i1} + 10a_{i2} + a_{i3}$, generated in single precision.)

- ○ Gauss-Jordan algorithm ($\vec{X}_0=(A^TA)^{-1}A^T\vec{B}$)
- ○ Andree algorithm ($\vec{X}_0=(A^TA)^{\#}A^T\vec{B}$)
- □ Gram-Schmidt algorithm ($\vec{X}_0=(A^TA)^{\#}A^T\vec{B}$)
- ▽ Gram-Schmidt algorithm ($\vec{X}_0=A^{\#}\vec{B}$)
- ▽ Householder algorithm ($\vec{X}_0$ obtained after triangulation of A-matrix)

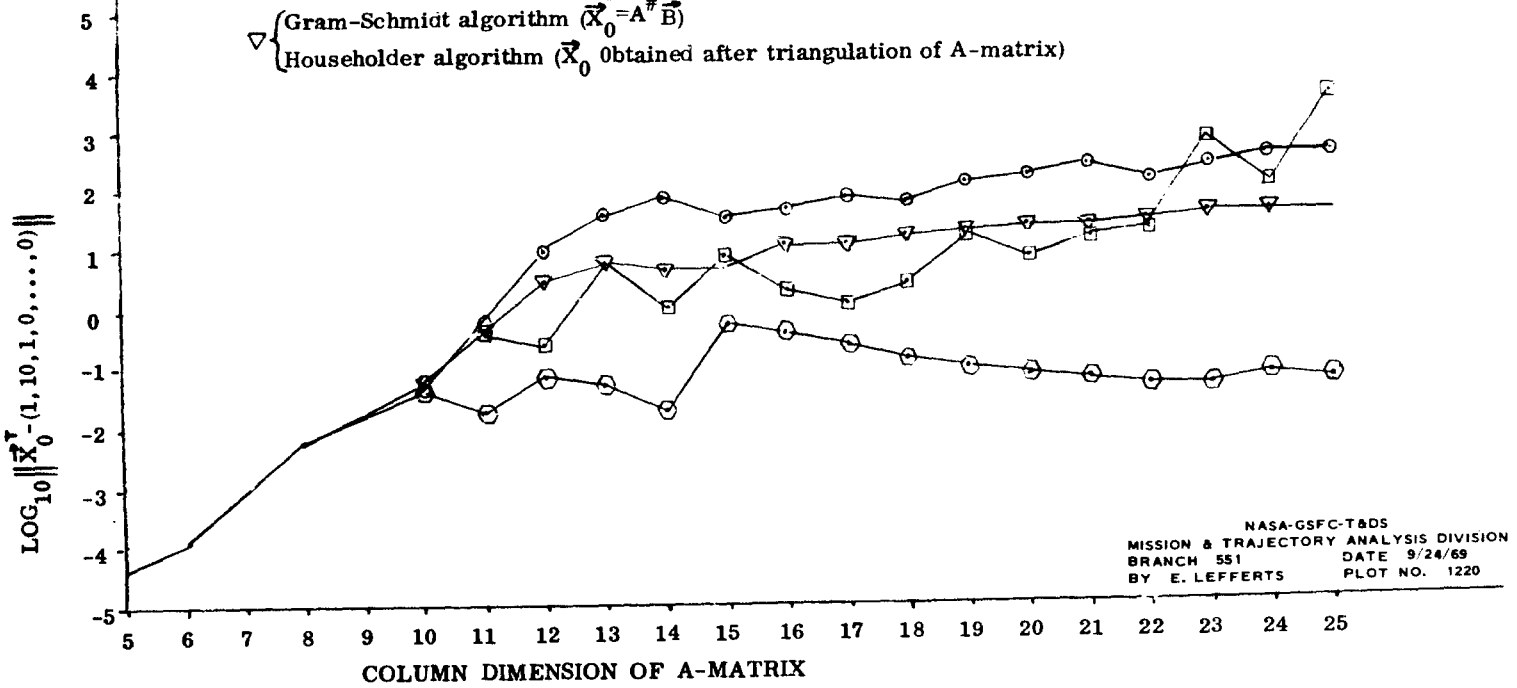


Figure 2-a

COMPUTATIONAL RANK COMPARISON FOR DOUBLE PRECISION MATRIX INVERSION SUBROUTINES. IBM 360/95 COMPUTER.

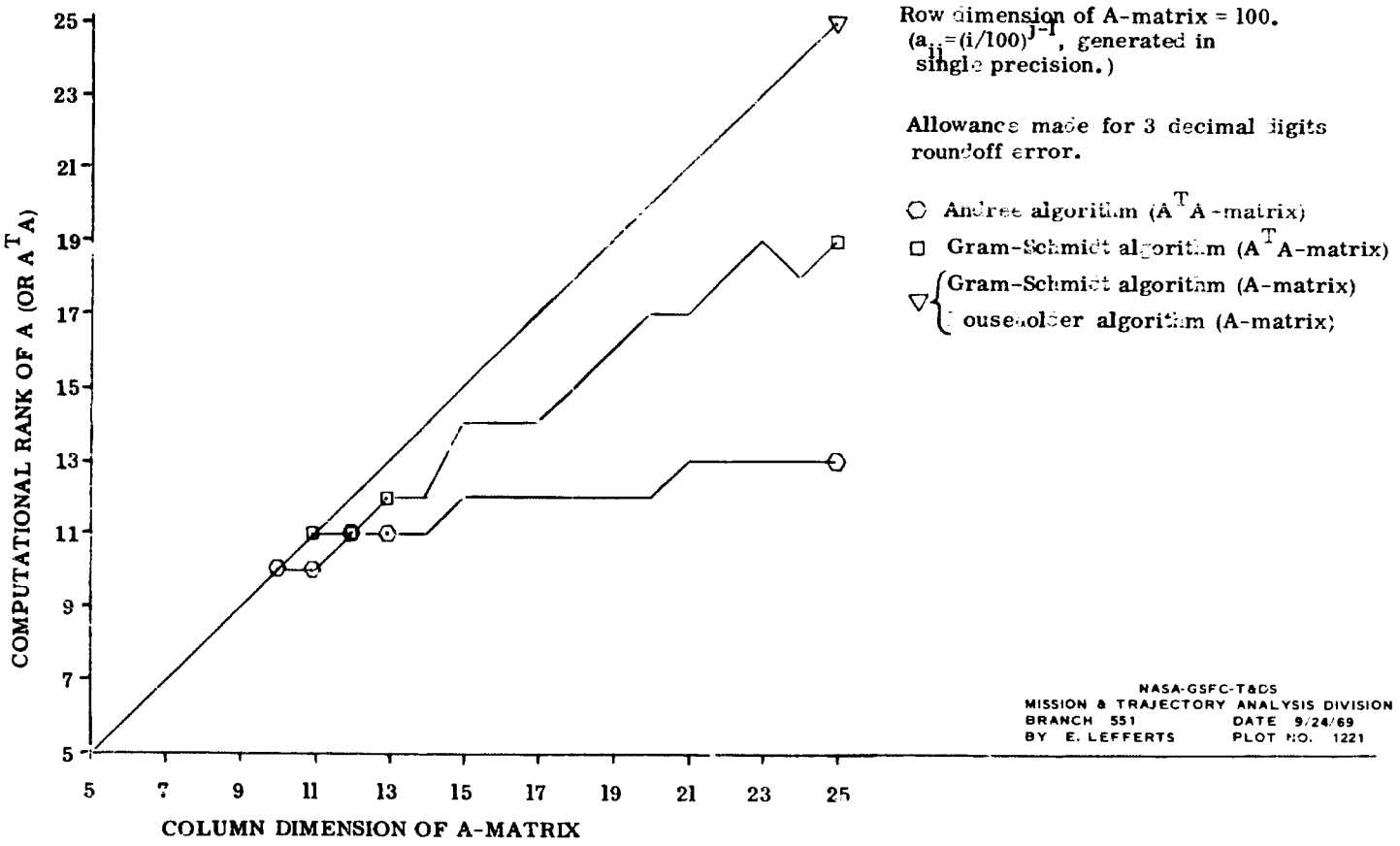


Figure 2-b

LEAST SQUARES SOLUTION ERROR COMPARISON USING
SINGLE PRECISION MATRIX INVERSION SUBROUTINES
IBM 360/95 COMPUTER

Row dimension of A-matrix = 100

($a_{ij} = [1/100]^{j-1}$, $b_i = a_{i1} + 10\,a_{i2} + a_{i3}$, generated in single precision)

o Gauss-Jordan algorithm ($\vec{X}_0 = (A^T A)^{-1} A^T \vec{B}$)

O Andree algorithm ($\vec{X}_0 = (A^T A)^{\#} A^T \vec{B}$)

□ Gram-Schmidt algorithm ($\vec{X}_0 = (A^T A)^{\#} A \vec{B}$)

▽ Gram-Schmidt algorithm ($\vec{X}_0 = A^{\#} \vec{B}$)

△ Householder algorithm ($\vec{X}_0$ obtained after direct triangulation of A-matrix)

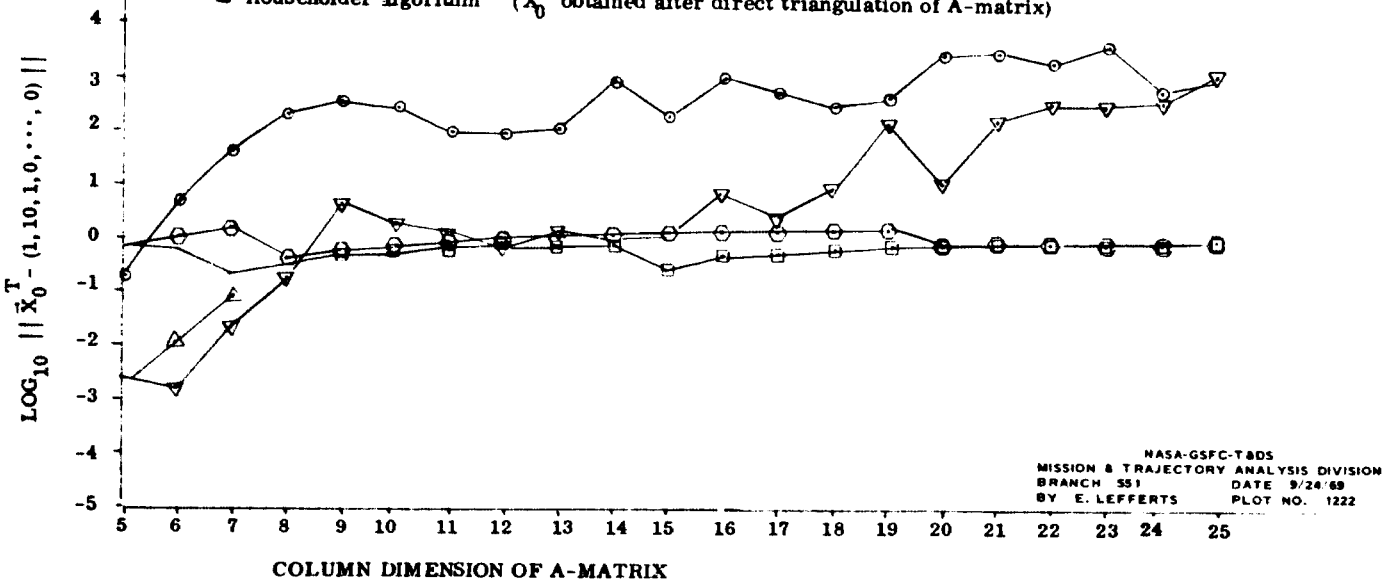


Figure 3-a

COMPUTATIONAL RANK COMPARISON FOR SINGLE PRECISION
MATRIX INVERSION SUBROUTINES IBM 360/95 COMPUTER

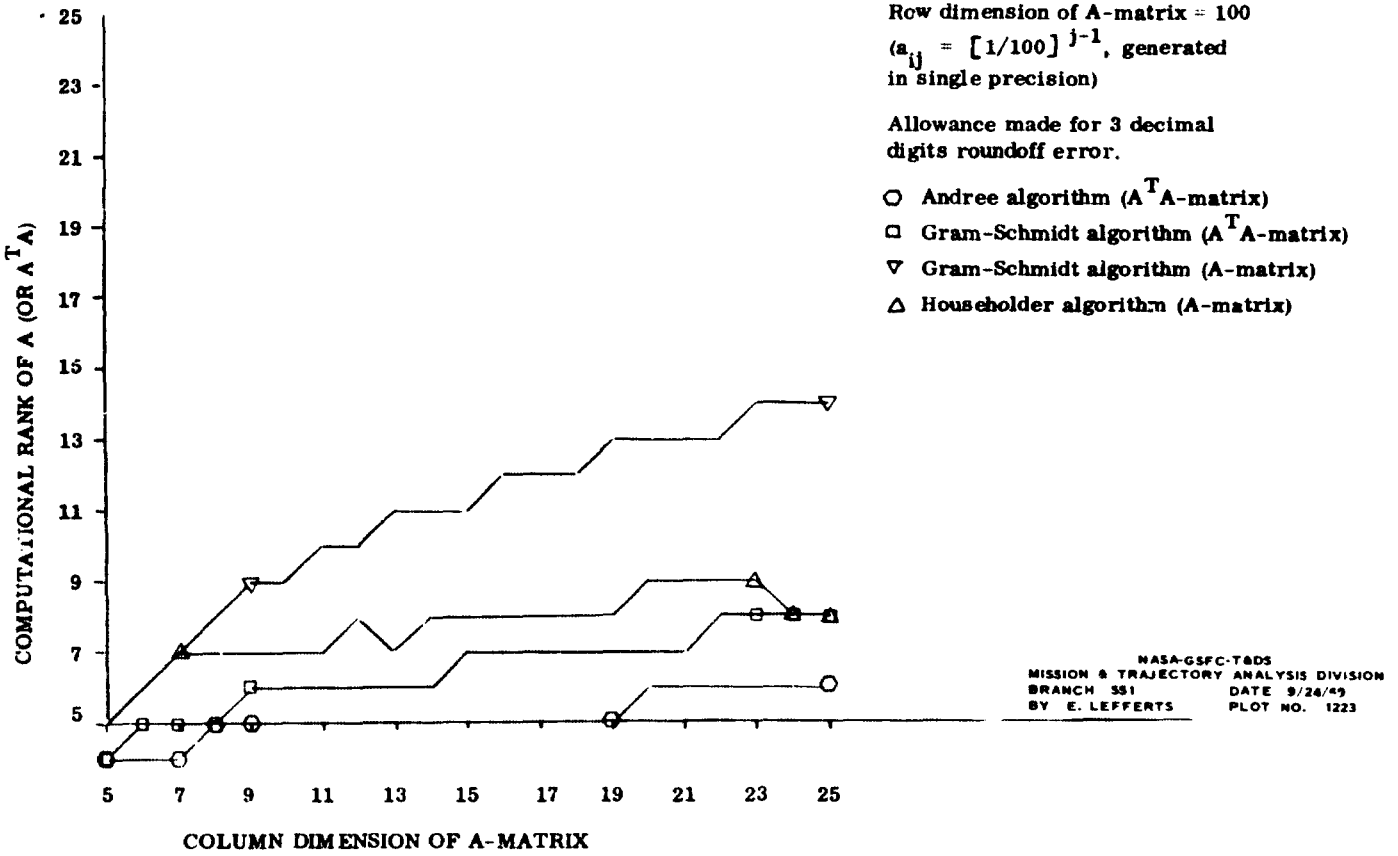


Figure 3-b

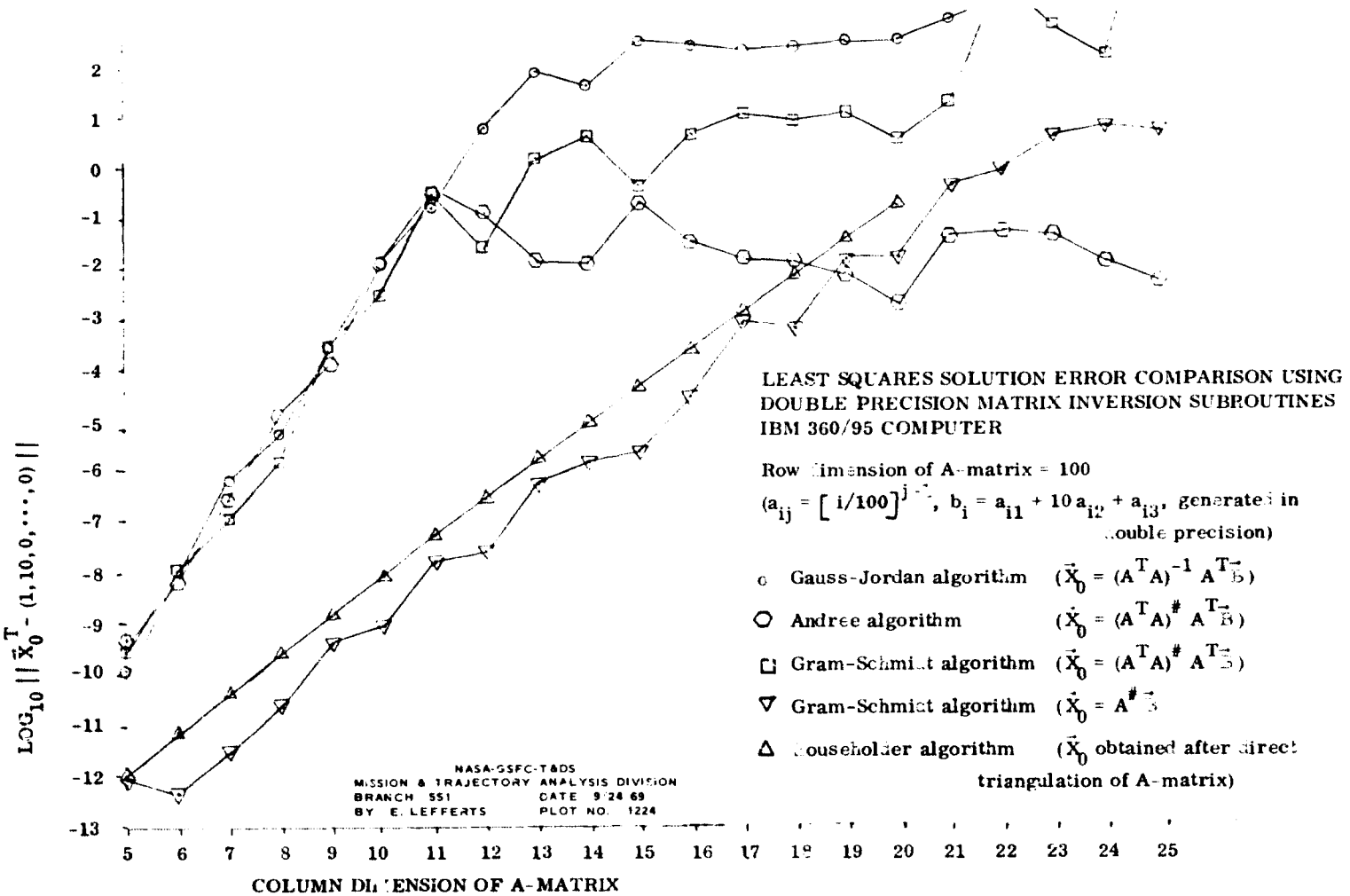

LEAST SQUARES SOLUTION ERROR COMPARISON USING
DOUBLE PRECISION MATRIX INVERSION SUBROUTINES
IBM 360/95 COMPUTER
Row dimension of A-matrix = 100
(a_ij = [i/100]^(j-1), b_i = a_i1 + 10 a_i2 + a_i3, generated in double precision)
Gauss-Jordan algorithm (X_0 = (A^T A)^-1 A^T B)
Andree algorithm (X_0 = (A^T A)^# A^T B)
Gram-Schmidt algorithm (X_0 = (A^T A)^# A^T B)
Gram-Schmidt algorithm (X_0 = A^# B)
Householder algorithm (X_0 obtained after direct triangulation of A-matrix)
NASA-GSFC-T&DS
MISSION & TRAJECTORY ANALYSIS DIVISION
BRANCH 551 DATE 9/24/69
BY E. LEFFERTS PLOT NO. 1224
LOG10 || X_0^T - (1,10,0,...,0) ||
COLUMN DIMENSION OF A-MATRIX
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-11
-12
-13
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Figure 4-c

COMPUTATIONAL RANK COMPARISON FOR DOUBLE PRECISION
MATRIX INVERSION SUBROUTINES IBM 360/95 COMPUTER

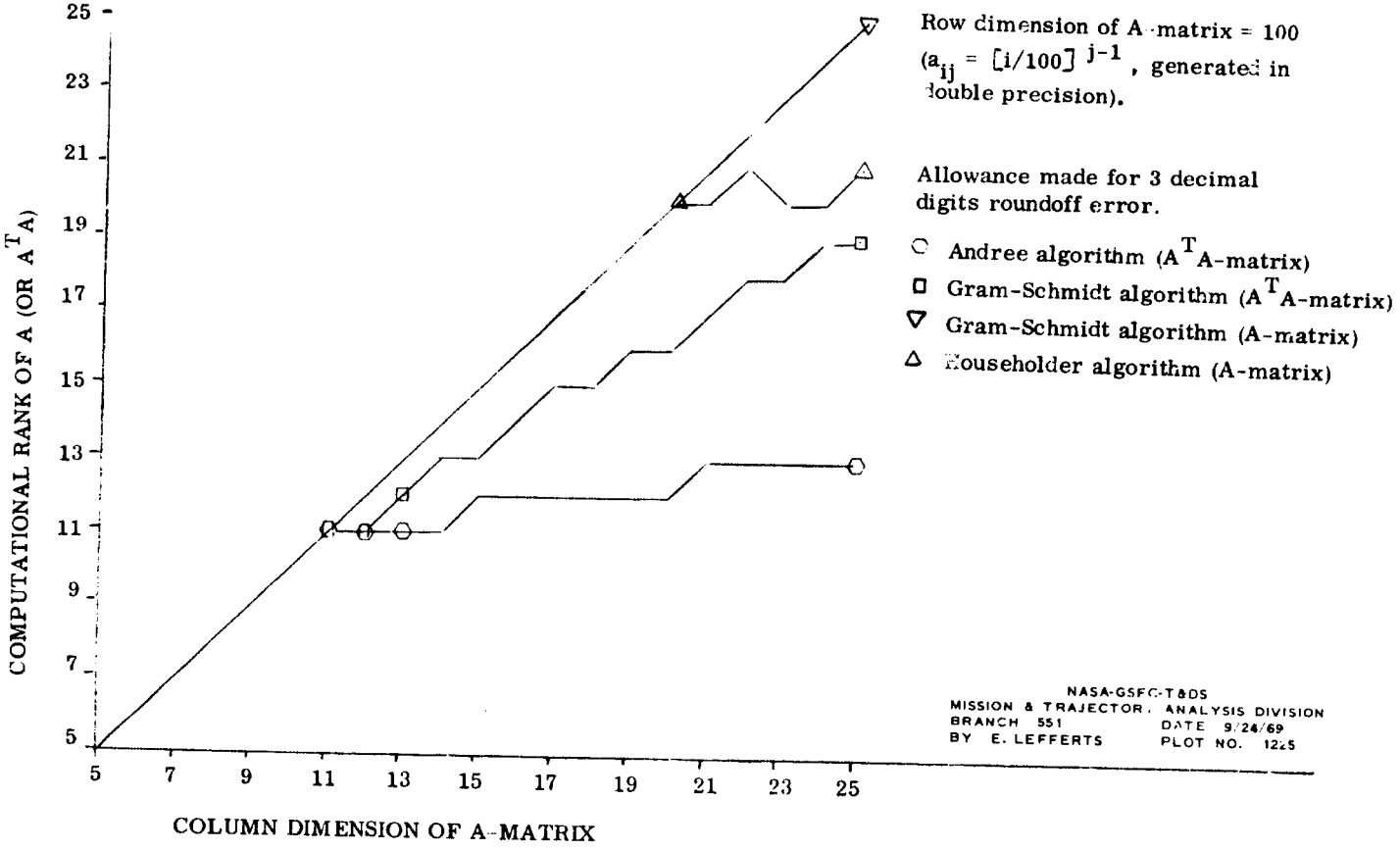


Figure 4-b

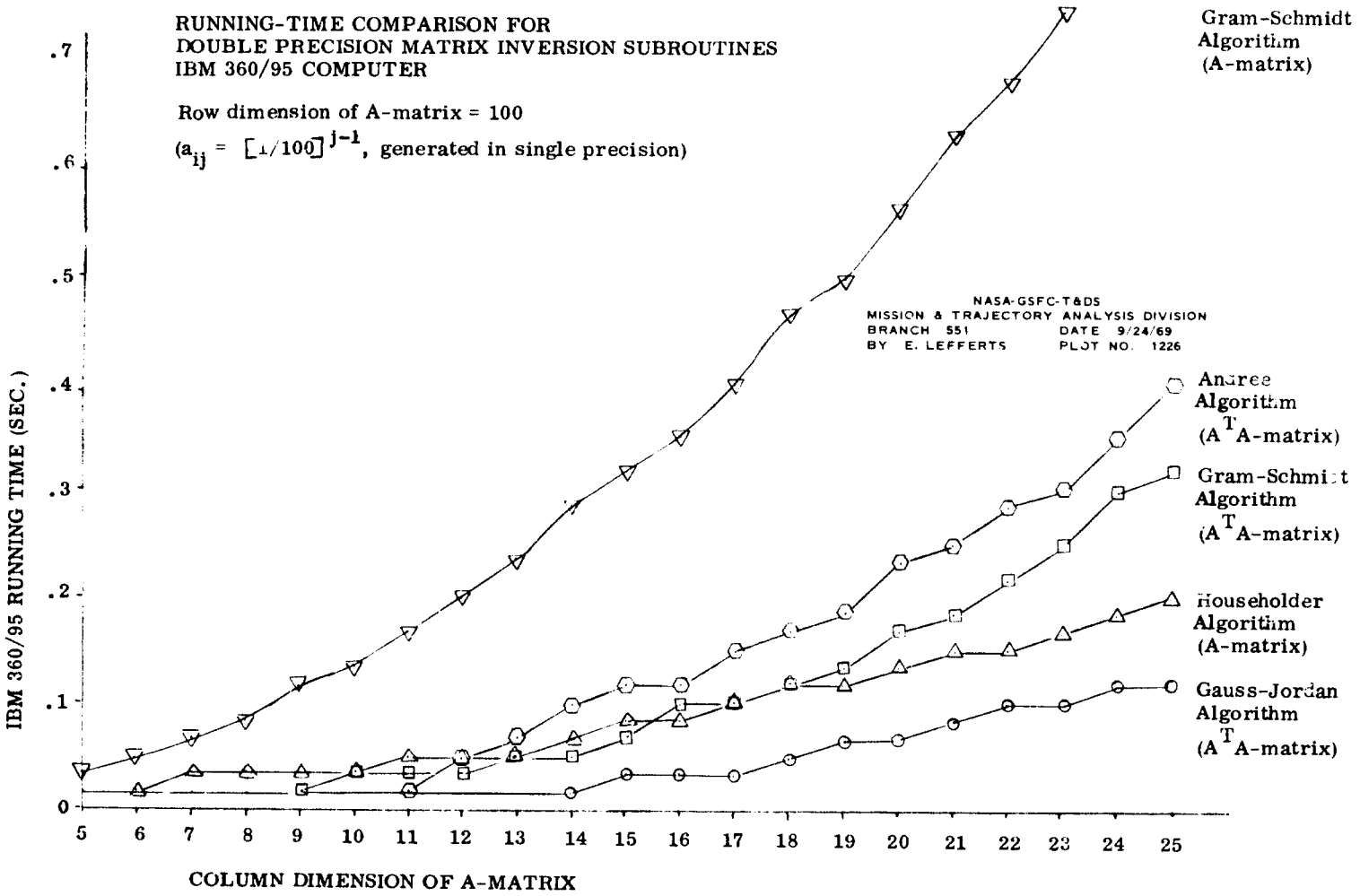

RUNNING-TIME COMPARISON FOR
DOUBLE PRECISION MATRIX INVERSION SUBROUTINES
IBM 360/95 COMPUTER
Row dimension of A-matrix = 100
(a_{ij} = $[i/100]^{j-1}$, generated in single precision)
NASA-GSFC-T&DS
MISSION & TRAJECTORY ANALYSIS DIVISION
BRANCH 551 DATE 9/24/69
BY E. LEFFERTS PLOT NO. 1226
Gram-Schmidt Algorithm (A-matrix)
Andree Algorithm (A^TA-matrix)
Gram-Schmidt Algorithm (A^TA-matrix)
Householder Algorithm (A-matrix)
Gauss-Jordan Algorithm (A^TA-matrix)
IBM 360/95 RUNNING TIME (SEC.)
0
.1
.2
.3
.4
.5
.6
.7
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
COLUMN DIMENSION OF A-MATRIX

Figure 5

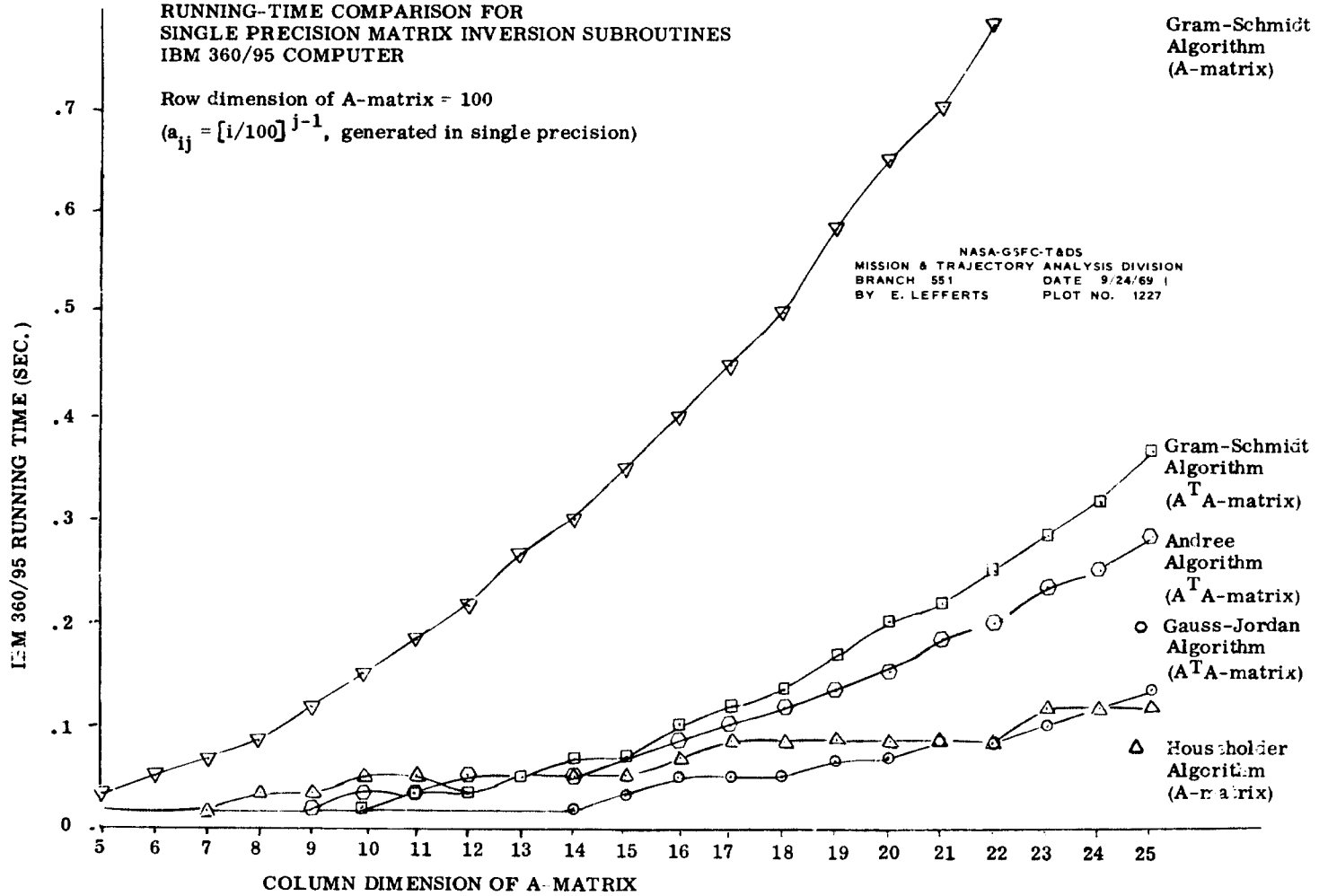

RUNNING-TIME COMPARISON FOR
SINGLE PRECISION MATRIX INVERSION SUBROUTINES
IBM 360/95 COMPUTER
Row dimension of A-matrix = 100
($a_{ij} = [i/100]^{j-1}$, generated in single precision)
NASA-GSFC-T&DS
MISSION & TRAJECTORY ANALYSIS DIVISION
BRANCH 551 DATE 9/24/69
BY E. LEFFERTS PLOT NO. 1227
Gram-Schmidt Algorithm (A-matrix)
Gram-Schmidt Algorithm (A^TA-matrix)
Andree Algorithm (A^TA-matrix)
Gauss-Jordan Algorithm (A^TA-matrix)
Householder Algorithm (A-matrix)
IBM 360/95 RUNNING TIME (SEC.)
0
.1
.2
.3
.4
.5
.6
.7
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
COLUMN DIMENSION OF A-MATRIX

Figure 6

REFERENCES

1. Wilkinson, J. H.: The Algebraic Eigenvalue Problem. Clarendon Press (Oxford), 1965, Chapter 3.

2. Fox, L.: An Introduction to Numerical Linear Algebra. Clarendon Press (Oxford), 1964, Chapter 3.

3. Lefferts, Eugene J.: Least Squares and Pseudoinversion. X-551-69-73.

APPENDIX A

PENROSE PSEUDOINVERSE

Let $A^{\#}$ be defined as the Penrose pseudoinverse of A, where $A^{\#}$ satisfies the four axioms:

Axiom 1: $AA^{\#}A = A$.

Axiom 2: $A^{\#}AA^{\#} = A^{\#}$.

Axiom 3: $(AA^{\#})^T = A^{\#T}A^T = AA^{\#}$.

Axiom 4: $(A^{\#}A)^T = A^TA^{\#T} = A^{\#}A$.

We will show that the Penrose pseudoinverse always exists and is unique. If the matrix A is non-singular, then the pseudoinverse is identical to the inverse.

THEOREM A-1: If A^{-1} exists, then $A^{\#} = A^{-1}$.

Proof:

$$AA^{\#}A = A,$$

$$A^{-1}AA^{\#}AA^{-1} = (A^{-1}A)\,A^{\#}\,(AA^{-1}) = A^{-1}AA^{-1},$$

$$IA^{\#}I = A^{-1}I = IA^{-1} = A^{-1},$$

$$A^{\#} = A^{-1}.$$

THEOREM A-2: $A^{\#}$ is unique.

Proof: Assume X and Y are Penrose pseudoinverses of A. Then, both X and Y satisfy axioms 1 through 4. Thus:

$$X = XAX = (XA)X = (XA)^TX = A^TX^TX = (A^TY^TA^T)X^TX$$

$$= (A^T Y^T) A^T X^T X = (YA)^T A^T X^T X = YAA^T X^T X = YA(A^T X^T) X$$

$$= YA(XA)^T X = YA(XAX) = YAX = Y(AX) = Y(AX)^T = YX^T A^T$$

$$= YX^T A^T Y^T A^T = Y(X^T A^T)\ (Y^T A^T) = Y(AX)^T\ (AY)^T$$

$$= YAXAY = Y(AXA)Y = YAY = Y.$$

THEOREM A-3: $A^{\#T} = A^{T\#}$.

Proof: Since $(A^T)^\#$ is the unique Penrose pseudoinverse of A^T, it is sufficient to show that $A^{\#T}$ satisfies axioms 1 through 4:

(1) $$AA^\# A = A.$$

Transposing,

$$A^T A^{\#T} A^T = A^T.$$

(2) $$A^\# AA^\# = A^\#.$$

Transposing,

$$A^{\#T} A^T A^{\#T} = A^{\#T}.$$

(3) $$(AA^\#)^T = AA^\# = A^{\#T} A.$$

Thus,

$$(A^{\#T} A^T)^T = (AA^\#)^T = AA^\# = A^{\#T} A^T.$$

(4) $$(A^\# A)^T = A^T A^{\#T} = A^\# A.$$

Thus,

$$(A^T A^{\#T})^T = (A^\# A)^T = A^\# A = A^T A^{\#T}.$$

THEOREM A-4: $(A^{\#})^{\#} = A$.

Proof: Since $A^{\#}$ is the pseudoinverse of A, it follows that A is the pseudoinverse of $(A^{\#})$, from the symmetry of the axioms. By definition $(A^{\#})^{\#}$ is the pseudoinverse of $A^{\#}$, thus $(A^{\#})^{\#} = A$ by theorem A-2.

THEOREM A-5: $A^{\#}$ exists.

Proof 1: Let A be a diagonal matrix:

$$A = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n).$$

Define $A^{\#}$ to be the diagonal matrix:

$$A^{\#} = \text{diag}(\mu_1, \mu_2, \ldots, \mu_n),$$

where

$$\mu_i = 1/\lambda_i, \ \lambda_i \neq 0$$

$$\mu_i = 0, \lambda_i = 0.$$

We now show that $A^{\#}$ satisfies axioms 1 and 2:

(1) $AA^{\#}A = \text{diag}(\lambda_i)\,\text{diag}(\mu_i)\,\text{diag}(\lambda_i) = \text{diag}(\lambda_i^2 \mu_i) = \text{diag}(\lambda_i) = A.$

(2) $A^{\#}AA^{\#} = \text{diag}(\mu_i)\,\text{diag}(\lambda_i)\,\text{diag}(\mu_i) = \text{diag}(\mu_i^2 \lambda_i) = \text{diag}(\mu_i) = A^{\#}.$

Since the product of diagonal matrices is diagonal, symmetry is preserved, thus satisfying axioms 3 and 4.

Proof 2: Let A be symmetric; that is, let $A = A^T$. Then, A has the representation

$$A = S^T D S,$$

where D is a diagonal matrix [$D = \text{diag}(\lambda_i)$] and S is orthogonal, that is $S^T = S^{-1}$. Define $A^\#$ as:

$$A^\# = S^T D^\# S.$$

We now show that $A^\#$ satisfies axioms 1 through 4:

(1) $AA^\#A = S^TDSS^TD^\#SS^TDS^\# = S^TDD^\#DS = S^TDS = A.$

(2) $A^\#AA^\# = S^TD^\#SS^TDSS^TD^\#S = S^TD^\#DD^\#S = S^TD^\#S = A^\#.$

(3) $(AA^\#)^T = (S^TDSS^TD^\#S)^T = (S^TDD^\#S)^T = S^T(DD^\#)^TS = S^TDD^\#S$

$= S^TDSS^TD^\#S = AA^\#.$

(4) $(A^\#A)^T = (S^TD^\#SS^TDS)^T = (S^TD^\#DS)^T = S^T(D^\#D)^TS = S^TD^\#DS$

$= S^TD^\#SS^TDS = A^\#A.$

Proof 3: Let A be arbitrary, then we define $A^\#$ as either

$$A^\# = (A^TA)^\# A^T$$

or

$$A^\# = A^T(AA^T)^\#,$$

depending upon which resulting symmetric matrix, A^TA or AA^T, has the smaller dimension.

We now show that $A^\#$ satisfies axioms 1 through 4:

(1) $AA^\#A = A(A^TA)^\#A^TA$

Let

$$C = A(A^TA)^{\#}A^TA - A,$$

then C^T is given by

$$C^T = A^TA(A^TA)^{\#}A^T - A^T$$

and C^TC is given by

$$C^TC = \left[A^TA(A^TA)^{\#}A^T - A^T\right]\left[A(A^TA)^{\#}A^TA - A\right]$$

$$= A^TA(A^TA)^{\#}A^TA(A^TA)^{\#}A^TA - A^TA(A^TA)^{\#}A^TA$$

$$- A^TA(A^TA)^{\#}A^TA + A^TA$$

$$= A^TA - A^TA - A^TA + A^TA$$

$$= 0.$$

Thus, $C = 0$, or

$$A(A^TA)^{\#}A^TA = A.$$

(2) $A^{\#}AA^{\#} = (A^TA)^{\#}A^TA(A^TA)^{\#}A^T = (A^TA)^{\#}A^T = A^{\#}.$

(3) $(A^{\#}A)^T = \left[(A^TA)^{\#}A^TA\right]^T = \left[\vec{B}^{\#}\vec{B}\right]^T = \vec{B}^{\#}\vec{B} = (A^TA)^{\#}A^TA = A^{\#}A.$

(4) $(AA^{\#})^T = \left[A(A^TA)^{\#}A^T\right]^T = A^T(A^TA)^{\#T}A = A^T(A^TA)^{T\#}A$

$$= A^T(A^TA)^{\#}A = AA^{\#}$$

The main utility of the Penrose pseudoinverse is contained in the following result. Let the system of equations

$$A\vec{X} = \vec{B}$$

be given, and let

$$\vec{X}_0 = A^{\#}\vec{B}.$$

Then, for any $\vec{X}$,

$$||A\vec{X}_0 - \vec{B}|| \leq ||A\vec{X} - \vec{B}||.$$

If for some $\vec{X}$ the equality holds, then

$$||\vec{X}_0|| < ||\vec{X}||.$$

Thus, the solution given by the Penrose pseudoinverse is the least squares solution. If there is more than one least squares solution, the solution given by the Penrose pseudoinverse is the smallest solution in norm.

THEOREM A-6 (Projection Lemma):

$$||A\vec{X}_0 - \vec{B}|| = \underset{\vec{X}}{MIN} ||A\vec{X} - \vec{B}||$$

if and only if

$$\vec{X}^T A^T (A\vec{X}_0 - \vec{B}) = 0$$

for all $\vec{X}$.

Proof: Assume that

$$\vec{X}^T A^T (A\vec{X}_0 - \vec{B}) = 0$$

for all $\vec{X}$, and let $\vec{Y}$ be given by

$$\vec{Y} = \vec{X}_0 + \vec{Z}.$$

We then have

$$||A\vec{Y} - \vec{B}||^2 = ||A\vec{X}_0 - \vec{B} + A\vec{Z}||^2 = ||A\vec{X}_0 - \vec{B}||^2 + 2\vec{Z}^T A^T(A\vec{X}_0 - \vec{B}) + ||A\vec{Z}||^2,$$

since by assumption

$$\vec{Z}^T A^T(A\vec{X}_0 - \vec{B}) = 0.$$

Then,

$$||A\vec{Y} - \vec{B}||^2 = ||A\vec{X}_0 - \vec{B}||^2 + ||A\vec{Z}||^2.$$

Thus,

$$||A\vec{Y} - \vec{B}||^2 \leq ||A\vec{X}_0 - \vec{B}||^2$$

since

$$||A\vec{Z}||^2 \geq 0.$$

We now assume that

$$||A\vec{X}_0 - \vec{B}|| \leq ||A\vec{Y} - \vec{B}||$$

and wish to show that this implies that

$$\vec{X}^T A^T(A\vec{X}_0 - \vec{B}) = 0$$

for all $\vec{X}$. Assume that

$$\vec{X}^T A^T (A\vec{X}_0 - \vec{B}) = \alpha \neq 0$$

for some $\vec{X}$. Let $\vec{V}$ and $\vec{Y}$ be defined as

$$\vec{V} = \frac{-\alpha \vec{X}}{||A\vec{X}||^2},$$

$$\vec{Y} = \vec{X}_0 + \vec{V}.$$

Then,

$$||A\vec{Y} - \vec{B}||^2 - ||A\vec{X}_0 - \vec{B}||^2 = ||A\vec{X}_0 - \vec{B} - \frac{A\vec{X}\alpha}{||A\vec{X}||^2}||^2 - ||A\vec{X}_0 - \vec{B}||^2$$

$$= \frac{-2\alpha \vec{X}^T A^T (A\vec{X}_0 - \vec{B})}{||A\vec{X}||^2} + \alpha^2 \frac{||A\vec{X}||^2}{||A\vec{X}||^4}$$

$$= \frac{-2\alpha^2}{||A\vec{X}||^2} \frac{+\alpha^2}{||A\vec{X}||^2} = \frac{-\alpha^2}{||A\vec{X}||^2},$$

which is a contradiction.

THEOREM A-7: In the system of equations

$$A\vec{X} = \vec{B},$$

where

$$\vec{X}_0 = A^{\#}\vec{B}$$

and

$$\vec{X} \neq \vec{X}_0,$$

either

$$||A\vec{X} - \vec{B}|| > ||A\vec{X}_0 - \vec{B}||$$

or

$$||A\vec{X} - \vec{B}|| = ||A\vec{X}_0 - \vec{B}|| \text{ and } ||\vec{X}|| > ||\vec{X}_0||.$$

Proof: First we must show that

$$\vec{X}^T A^T (A\vec{X}_0 - \vec{B}) = 0$$

as follows:

$$\begin{aligned}
\vec{X}^T A^T (AA^{\#} \vec{B} - \vec{B}) &= \vec{X}^T A^T AA^{\#} \vec{B} - \vec{X}^T A^T \vec{B} \\
&= \vec{X}^T A^T (AA^{\#}) \vec{B} - \vec{X}^T A^T \vec{B} \\
&= \vec{X}^T A^T (AA^{\#})^T \vec{B} - \vec{X}^T A^T \vec{B} \\
&= \vec{X}^T A^T A^{\#T} A^T \vec{B} - \vec{X}^T A^T \vec{B} \\
&- \vec{X}^T A^T \vec{B} - \vec{X}^T A^T \vec{B} \\
&= 0.
\end{aligned}$$

Thus, from theorem A-6,

$$||A\vec{X} - \vec{B}|| \geq ||A\vec{X}_0 - \vec{B}||.$$

Assume the equality holds:

$$||A\vec{X} - \vec{B}|| = ||A\vec{X}_0 - \vec{B}||.$$

If

$$\vec{X} = \vec{X}_0 + \vec{Y},$$

then, by hypothesis,

$$||A\vec{X} - \vec{B}||^2 - ||A\vec{X}_0 - \vec{B}||^2 = 2\vec{Y}^T A^T(A\vec{X}_0 - \vec{B}) + ||A\vec{Y}||^2$$

$$= ||A\vec{Y}||^2 = 0.$$

Now,

$$||\vec{X}_0||^2 = ||\vec{X} - (\vec{X} - \vec{X}_0)||^2 = ||\vec{X}||^2 + ||\vec{X} - \vec{X}_0||^2 - 2\vec{X}^T(\vec{X} - \vec{X}_0)$$

$$= ||\vec{X}||^2 + ||\vec{Y}||^2 - 2(\vec{X}_0 + \vec{Y})^T\vec{Y}$$

$$= ||\vec{X}||^2 + ||\vec{Y}||^2 - 2\vec{B}^T A^{\#T}\vec{Y} - 2||\vec{Y}||^2$$

$$= ||\vec{X}||^2 - ||\vec{Y}||^2 - 2\vec{B}^T A^{\#T}\vec{Y}$$

$$= ||\vec{X}||^2 - ||\vec{Y}||^2 - 2\vec{B}^T A^{\#T} A^T A^{\#T}\vec{Y}$$

$$= ||\vec{X}||^2 - ||\vec{Y}||^2 - 2\vec{B}^T A^{\#T}(A^{\#}A)^T\vec{Y}$$

$$= ||\vec{X}||^2 - ||\vec{Y}||^2 - 2\vec{B}^T A^{\#T} A^T(A\vec{Y}).$$

Since $||A\vec{Y}|| = 0$, it follows that

$$||\vec{X}_0||^2 = ||\vec{X}||^2 - ||\vec{Y}||^2.$$

APPENDIX B

PSEUDOINVERSE ALGORITHMS

There exist many algorithms for the generation of the pseudoinverse of Penrose. Unfortunately, not much information is available in terms of their efficiency and computer requirements. Such a comparison is presently being made and will be the subject of a future report. In the interim, we will present two such algorithms which have been used extensively by the author. These algorithms possess the desirable property of allowing a limited control of the computational rank.

In the first method, the pseudoinverse is formed by means of the Andree algorithm. This routine computes the pseudoinverse by means of the equation

$$A^{\#} = (A^T A)^{\#} A^T$$

Actually, the pseudoinverse of

$$B = (A^T A)$$

is formed. This routine insures the symmetry of $B^{\#}$ and also imposes the requirement that $B^{\#}$ be non-negative. Thus, if computational errors leading to the loss of the positive definiteness of B exist, they are partially alleviated by this scheme.

The second algorithm, which is basically a Gram-Schmidt orthogonalization procedure, can operate directly on the rectangular matrix A. There is no need to form the matrix associated with the normal equations, thus one major source of computational errors is eliminated. The penalty paid for this lies in the larger computer storage required to handle the rectangular matrix.

ANDREE ALGORITHM

The algorithm used in this paper is a modification of the Andree algorithm by T. S. Englar[1]. The steps of this algorithm are as follows:

[1]Kalman, R. D. and Englar, T. S., "An Automatic Synthesis Program for Optical Filters and Control Systems," NASA, July 1963.

1. Compute A^TA or AA^T, whichever has the smallest dimension. Call this resulting matrix B. It will be sufficient to compute $B^{\#}$, since $A^{\#}$ is given by

$$A^{\#} = (A^TA)^{\#} A^T \text{ or } A^{\#} = A^T(AA^T)^{\#}.$$

2. Compute a non-singular matrix, $S = (s_{ij})$, such that

$$SBS^T = E,$$

where $E = (e_{ij})$ is a diagonal matrix with elements either zero or one.

3. If $E = I$ then B is invertible, and

$$B^{-1} = S^TS.$$

If $E \neq I$ then we define the matrix $U = (u_{ij})$ by the following:

For $i \neq j$, $u_{ij} = -s_{ij}$ if $e_{ii} = 0$,

$= 0$ if $e_{ii} = 1$.

For $i = j$, $u_{ii} = 0$ if $e_{ii} = 0$,

$= 1$ if $e_{ii} = 1$.

4. Compute

$$C = U^TBU.$$

Delete the rows and columns of C corresponding to $C_{ii} = 0$, and call the resulting matrix D. Observe that D is a non-singular matrix of rank m. Compute D^{-1} by means of the Andree algorithm as in step 2. Compute $B^{\#}$ by means of

$$B^{\#} = U \begin{pmatrix} D^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^{T}.$$

5. Compute $A^{\#}$ either by means of

$$A^{\#} = B^{\#} A^{T}$$

or

$$A^{\#} = A^{T} B^{\#}.$$

In the computation of step 2, the generation of the matrix S is done in at most 2n steps. The reduction is based upon pivoting in each step about the largest of the remaining diagonal elements. If after any step of iteration the largest of the remaining diagonal elements is less than the product of a preassigned constant, K, and the first pivotal element, then the rows and columns containing these elements are set to zero thus reducing the rank of the matrix.

GRAM-SCHMIDT PROCEDURE

This algorithm[2] permits one to pseudoinvert a rectangular matrix, A, directly. The algorithm is based upon partitioning A in the form

$$A = (R , RU),$$

where all columns of R are linearly independent. The pseudoinverse $A^{\#}$ is given by

$$A^{\#} = \begin{pmatrix} (I + UU^{T})^{-1} R^{\#} \\ U^{T} (I + UU^{T})^{-1} R^{\#} \end{pmatrix}$$

[2]Rust, B., Burrus, W. R. and Schneeberger, C., "A Simple Algorithm for computing the Generalized Inverse of a Matrix," Communications of the ACM, Vol. 9, No. 5, May 1966, pp 381-387.

This representation may be checked by substitution into the axioms.

If $(\vec{a}_1, \vec{a}_2, \ldots \vec{a}_n)$ is any set of linearly independent vectors, then it can be replaced by an orthonormal set $(\vec{q}_1, \vec{q}_2, \ldots \vec{q}_n)$ in the following manner:

$$\vec{q}_1 = \frac{\vec{a}_1}{||\vec{a}_1||} \qquad \vec{c}_2 = \vec{a}_2 - (\vec{a}_2^T \vec{q}_1)\,\vec{q}_1 .$$

$$\vec{q}_2 = \frac{\vec{c}_2}{||\vec{c}_2||} \qquad \vec{c}_3 = \vec{a}_3 - (\vec{a}_3^T \vec{q}_1)\,\vec{q}_1 - (\vec{a}_3^T \vec{q}_2)\,\vec{q}_2 .$$

$$\vec{q}_3 = \frac{\vec{c}_3}{||\vec{c}_3||}$$

$$\vec{c}_n = \vec{a}_n - \sum_{i=1}^{n-1} (\vec{a}_n^T \vec{q}_i)\,\vec{q}_i .$$

$$\vec{q}_n = \frac{\vec{c}_n}{||\vec{c}_n||}$$

Observe that the resulting matrix Q is such that

$$Q^T Q = I .$$

Let us apply this transformation to A = (R, RU) and keep track of the transformation by applying it simultaneously to the identity matrix partitioned as

$$I = \begin{pmatrix} I_k & 0 \\ 0 & I_{n-k} \end{pmatrix} .$$

A will transform into

$$A = (R \quad , \; RU) \longrightarrow (Q \quad , \; 0)$$

$$\begin{pmatrix} I_k & 0 \\ 0 & I_{n-k} \end{pmatrix} \longrightarrow \begin{pmatrix} Z & X \\ 0 & I_{n-k} \end{pmatrix}$$

Thus, we must have

$$(R \quad , \quad RU)\begin{pmatrix} Z & X \\ 0 & I_{n-k} \end{pmatrix} = (RZ \; , \; RX + RU) = (Q \; , \; 0)$$

or

$$RZ = Q \qquad R = QZ^{-1}$$

$$RX + RU = 0 \qquad X = -U$$

and $R^{\#}$ is given by

$$R^{\#} = Z\,Q^T$$

since

$$R^{\#} = (R^T R)^{-1} R^T = (Z^{-1^T} Q^T Q Z^{-1})\, Z^{-1^T} Q^T = ZQ^T .$$

Thus, the matrices $R^{\#}$ and U are generated. It is still necessary to compute the matrices $(I + UU^T)^{-1}$ and $U^T (I + UU^T)^{-1}$. The first of these expressions can be written as

$$(I + UU^T)^{-1} = I - U(U^TU + I)^{-1} U^T .$$

If both sides are post-multiplied by $(I + UU^T)$, the identity follows. The second expression can be written as

$$U^T(I + UU^T)^{-1} = (U^TU + 1)^{-1}U^T,$$

since

$$\begin{aligned}
U^T(I + UU^T)^{-1} &= U^T - U^TU(U^TU + I)^{-1}U^T \\
&= \left[I - U^TU(U^TU + I)^{-1}\right]U^T \\
&= \left[I + (U^TU + I)^{-1} - (U^TU + I)^{-1} - U^TU(U^TU + I)^{-1}\right]U^T \\
&= \left[I + (U^TU + I)^{-1} - (I + U^TU)\,(I + U^TU)^{-1}\right]U^T \\
&= \left[I + (U^TU + I)^{-1} - I\right]U^T \\
&= (U^TU + I)^{-1}U^T.
\end{aligned}$$

If the Gram-Schmidt orthogonalization process is now applied to the matrix

$$\begin{pmatrix} -U \\ \\ I_{n-k} \end{pmatrix}$$

this will transform to

$$\begin{pmatrix} -U \\ \\ I_{n-k} \end{pmatrix} P = \begin{pmatrix} -UP \\ \\ P \end{pmatrix},$$

where we have

$$\begin{pmatrix} -UP \\ P \end{pmatrix}^T \begin{pmatrix} -UP \\ P \end{pmatrix} = P^T U^T U P + P^T P = I,$$

or this becomes

$$P^T (U^T U + I) P = I$$

$$(U^T U + I) = P^{T-1} P^{-1}$$

$$(U^T U + I)^{-1} = P P^T.$$

Also,

$$I - U(U^T U + I)^{-1} U^T = I - U P P^T U^T$$

$$= I - (UP)(UP)^T$$

and

$$(U^T U + I)^{-1} U^T = P(UP)^T.$$

Thus, $A^{\#}$ takes the form

$$A^{\#} = \begin{pmatrix} \left[I - (UP)(UP)^T \right] R^{\#} \\ P(UP)^T R^{\#} \end{pmatrix}$$

APPENDIX C

HOUSEHOLDER'S TRIANGULATION METHOD FOR OBTAINING THE LEAST SQUARES SOLUTION

The following is a description of the triangulation scheme used in this study for obtaining the least squares solution directly from the data matrix without forming the normal equations. This procedure, originally described by Householder, has been developed into a workable algorithm for least squares problems by Golub.[1] The subroutine and its description was taken from the IBM system 360 scientific subroutine package.[2]

In our linear system,

$$A\vec{X} = \vec{B} \,, \tag{1}$$

both matrix A and vector $\vec{B}$ are premultiplied by an orthogonal matrix Q which transforms A into upper triangular form. All elements below the main diagonal will then be equal to zero; that is,

$$QA = R = \begin{pmatrix} \tilde{R} \\ \cdots \\ 0 \end{pmatrix}, \tag{2}$$

$$Q\vec{B} = \vec{C} = \begin{pmatrix} \vec{C}_1 \\ \vec{C}_2 \end{pmatrix}, \tag{3}$$

where $\tilde{R}$ is an n x n upper triangular matrix and $\vec{C}_1$ represents the first n elements of $Q\vec{B}$. Since the Euclidean norm is preserved under an orthogonal transformation,

$$||A\vec{X} - \vec{B}|| = ||R\vec{X} - \vec{C}|| \,, \tag{4}$$

and the least squares solution is clearly

$$\vec{X}_0 = \tilde{R}^{-1} \vec{C}_1 \tag{5}$$

provided $\tilde{R}^{-1}$ exists.

An effective way to realize this decomposition is via Householder transformations. The algorithm is a recursive n-step procedure defined by the following recursion formulas:

$$A^{(1)} = (a_{ij}^{(1)}) = A \,, \tag{6}$$

---

[1] Golub, G., "Numerical Methods for Solving Linear Least Squares Problems," Numerische Mathematic, Vol. 7, No. 3 (1965), pp. 206-216.

[2] IBM System 360 Publication No. H20-0205-2, 1967, pp. 191-194.

$$A^{(k+1)} = (a_{ij}^{(k+1)}) = P^{(k)} A^{(k)}, \quad k = 1, 2, \cdots, n \ . \tag{7}$$

In order to get an upper triangular matrix $A^{(n+1)}$, every matrix $P^{(k)}$ $(k = 1, 2, \cdots, n)$ should be defined so that it is symmetric and orthogonal, and so that it transforms all elements of the k^{th} column of $A^{(k)}$ below the main diagonal to zero. All of these restrictions to $P^{(k)}$ are satisfied, setting

$$P^{(k)} = I - \beta_k \vec{U}^{(k)} \vec{U}^{(k)T} \tag{8}$$

with

$$\beta_k = \frac{1}{\sigma_k (\sigma_k + a_{kk}^{(k)})} \tag{9}$$

and

$$\sigma_k = \pm \sqrt{\sum_{i=k}^{m} (a_{ik}^{(k)})^2} \quad \text{with} \begin{cases} + \text{ for } a_{kk}^{(k)} \geq 0 \\ - \text{ for } a_{kk}^{(k)} < 0 \end{cases} \tag{10}$$

where I is the identity matrix.

$\vec{U}^{(k)T}$ denotes the transpose of column vector $\vec{U}^{(k)} = (u_i^{(k)})$, the m components of which are defined as follows:

$$u_i^{(k)} = 0 \text{ for } i < k \ , \tag{11}$$

$$u_k^{(k)} = \sigma_k + a_{kk}^{(k)} \ , \tag{12}$$

$$u_i^{(k)} = a_{ik}^{(k)} \text{ for } i > k \ . \tag{13}$$

Neither matrices $P^{(k)}$ $(k = 1, 2, \cdots, n)$ nor matrix $Q = P^{(n)} P^{(n-1)} \cdots P^{(1)}$ are computed explicitly, since from (7) and (8):

$$A^{(k+1)} = A^{(k)} - \vec{U}^{(k)} \vec{Y}^{(k)T} \tag{14}$$

with

$$\vec{Y}^{(k)T} = \beta_k \vec{U}^{(k)T} A^{(k)} \ . \tag{15}$$

Writing the components of row vector $\vec{Y}^{(k)^T}$ explicitly·

$$y_j^{(k)} = 0 \text{ for } j<k \,, \tag{16}$$

$$y_k^{(k)} = 1 \,, \tag{17}$$

$$y_j^{(k)} = \beta_k \sum_{i=k}^{m} u_i^{(k)} a_{ij}^{(k)} \text{ for } j>k \,. \tag{18}$$

Using (14), (16), (17), and (18), the explicit transformation formulas for matrix $A^{(k)}$ appear as follows:

$$a_{kk}^{(k+1)} = -\sigma_k \,, \tag{19}$$

$$a_{ik}^{(k+1)} = 0 \,, \; i = k+1, k+2, \cdots, m \,, \tag{20}$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - u_i^{(k)} y_j^{(k)} \,, \; j = k+1, k+2, \cdots, n$$
$$i = k, k+1, \cdots, m \,. \tag{21}$$

All other elements of matrix $A^{(k)}$ do not change. Naturally the same transformation is performed on vector $\vec{B}$.

Using (6) and (7) :

$$\vec{B}^{(1)} = (b_i^{(1)}) = \vec{B} \,, \tag{22}$$

$$\vec{B}^{(k+1)} = (b_i^{(k+1)} = P^{(k)} \vec{B}^{(k)} \,, \; k = 1, 2, \cdots, n \,. \tag{23}$$

Using (8) and (23) :

$$\vec{B}^{(k+1)} = \vec{B}^{(k)} - \vec{U}^{(k)} \vec{Z}^{(k)^T} \tag{24}$$

with

$$Z^{(k)^T} = \beta_k \vec{U}^{(k)^T} \vec{B}^{(k)} \,, \tag{25}$$

or explicitly:

$$b_i^{(k+1)} = b_j^{(k)} - u_i^{(k)} Z^{(k)} \quad , \quad i = 1, 2, \cdots, m \tag{26}$$

with

$$Z^{(k)} = \beta_k \sum_{i=k}^{m} u_i^{(k)} b_i^{(k)} \quad . \tag{27}$$

In order to keep roundoff errors as small as possible, column interchange is performed in such a way that, at the k^{th} stage, the column of $A^{(k)}$ is chosen to be reduced next which will maximize $| a_{kk}^{(k+1)} |$. Using (19) and (10), the index of this column is determined by giving the maximum overall j of:

$$s_j^{(k)} = \sum_{i=k}^{m} (a_{ij}^{(k)})^2 \quad , \quad j = k, k+1, \cdots, n \quad . \tag{28}$$

After $A^{(k+1)}$ has been computed, it is possible to compute $s_j^{(k+1)}$ as follows:

$$s_j^{(k+1)} = s_j^{(k)} - (a_{kj}^{(k+1)})^2 \quad , \quad j = k+1, k+2, \cdots, n \quad , \tag{29}$$

since the orthogonal transformations leave the column lengths invariant.

After having computed matrices $A^{(n+1)}$ (upper triangular matrix) and $\vec{B}^{(n+1)}$, the computation of the n by 1 solution vector $\vec{X}_0 = (x_i)$ is performed by back substitution according to the following formulas:

$$x_n = \frac{1}{a_{nn}^{(n+1)}} b_n^{(n+1)} \quad , \tag{30}$$

$$x_k = \frac{1}{a_{kk}^{(n+1)}} \left(b_k^{(n+1)} - a_{k,k+1}^{(n+1)} x_{k+1} - a_{k,k+2}^{(n+1)} x_{k+2} - \cdots \right.$$

$$\left. - a_{k,n-1}^{(n+1)} x_{n-1} - a_{kn}^{(n+1)} x_n \right) , \quad \begin{matrix} k = n - j + 1 \\ j = 2, 3, \cdots, n. \end{matrix} \tag{31}$$

After vector $\vec{X}_0$ has been computed, back interchange of elements is performed according to column interchanges in matrices $A^{(k)}$ in order to get the correct sequence of components in the solution vector $\vec{X}_0$.

The only case in which the whole procedure can fail occurs when, at any stage k, no column with nonzero parameter σ_k can be found; that is, no nonzero main diagonal element can be generated. In this case, the rank of matrix A is less than n. With respect to roundoff errors, the test is made in the following way.

At first, all elements $s_j^{(1)}$ are computed according to (28), and the maximum overall j (i.e., σ_1^2) is determined. With the relative tolerance ϵ given by input, the following absolute tolerance is generated:

$$\text{tol} = \sigma_1 \epsilon \quad . \tag{32}$$

If at any stage k (=1, 2, $\cdots$, n+1) the square root of the maximum overall j = k+1, $\cdots$, n of $s_j^{(k+1)}$ is not greater than $\sigma_1 \epsilon$, the rank of matrix A is declared to be k < n, and no solution can be found.

APPENDIX D

ANDREE ALGORITHM SUBROUTINE

The Andree algorithm for calculating the Penrose pseudoinverse of a matrix of equations has been converted into a FORTRAN subroutine. The listing for this subroutine appears on the following pages.

```
      SUBROUTINE ANDREE(V,N,NR,EPS)
      DOUBLE PRECISION V,B,T,W,X,U,P,RR,DSQRT
      DIMENSION V(30,30),B(30,30),R(30),T(30,30),W(30,30),X(30,30)
      DIMENSION U(30,30)
      BILL = EPS
      NB = 56
      IND = 0
      LEE = 0
      DO 333 I = 1,N
      DO 333 J = 1,N
  333 B(I,J) = V(I,J)
  666 DO 98 I = 1,N
      R(I) = 0.
      DO 97 J = 1,N
   97 T(I,J) = 0.
   98 T(I,I) = 1.
   35 LEE = LEE+1
      IF (LEE-N) 77,77,78
   77 P = 0.
      K = 0
      DO 22 I = 1,N
      IF (R(I)) 22,124,22
  124 IF (V(I,I)) 82,82,83
   82 DO 84 J = 1,N
      V(I,J) = 0.
   84 V(J,I) = 0.
      GO TO 22
   83 IF (P-V(I,I)) 21,22,22
   21 P = V(I,I)
      K = I
      IF (LEE.EQ.1) DMAX = P
   22 CONTINUE
      IF (P-10.**BILL*DMAX/2.**NB) 28,28,7
    7 R(K) = K
      P = DSQRT(V(K,K))
      DO 19 I = 1,N
      V(I,K) = V(I,K)/P
      T(I,K) = T(I,K)/P
      V(K,I) = V(K,I)/P
   19 T(K,I) = T(K,I)/P
      V(K,K) = 1.
      T(K,K) = 1./P
      DO 25 I = 1,N
      IF (I-K) 26,125,26
  125 DO 126 J = 1,N
      IF (I-J) 127,128,127
  127 W(I,J) = 0.
      X(I,J) = T(I,J)
      GO TO 126
  128 W(I,I) = 1.
      X(I,I) = T(I,I)
  126 CONTINUE
      GO TO 25
   26 DO 10 J = 1,N
      W(I,J) = V(I,J)-V(I,K)*V(K,J)
      X(I,J) = T(I,J)-V(I,K)*T(K,J)
```

```
 10 CONTINUE
 25 CONTINUE
    DO 24 I = 1,N
    DO 24 J = 1,N
    V(I,J) = W(I,J)
 24 T(I,J) = X(I,J)
    GO TO 35
 28 DO 30 I = 1,N
    IF (R(I)) 30,34,30
 34 DO 33 J = 1,N
    V(I,J) = 0.
 33 V(J,I) = 0.
 30 CONTINUE
 78 BILL = 0.
    IF (IND) 38,39,38
 39 JOE = N
    DO 40 I = 1,N
    IF (V(I,I)) 40,42,40
 42 JOE = JOE-1
 40 CONTINUE
    IF (JOE-N) 43,44,43
 44 DO 45 I = 1,N
    DO 45 II = 1,N
    RR = 0.
    DO 46 J = 1,N
 46 RR = RR+T(J,I)*T(J,II)
    V(I,II) = RR
 45 CONTINUE
    GO TO 36
 43 DO 47 I = 1,N
    IF (R(I)) 48,49,48
 49 DO 60 J = 1,N
 60 U(I,J) = -T(I,J)
    U(I,I) = 0.
    GO TO 47
 48 DO 61 J = 1,N
 61 U(I,J) = 0.
    U(I,I) = 1.
 47 CONTINUE
    DO 50 I = 1,N
    DO 50 II = 1,N
    RR = 0.
    DO 51 J = 1,N
 51 RR = RR+U(J,I)*B(J,II)
    W(I,II) = RR
 50 CONTINUE
    DO 52 I = 1,N
    DO 52 II = 1,N
    RR = 0.
    DO 53 J = 1,N
 53 RR = RR+W(I,J)*U(J,II)
    V(I,II) = RR
 52 CONTINUE
    IND = 1
    LEE = 0
    GO TO 666
```

```
38 DO 54 I = 1,N
   DO 54 II = 1,N
   RR = 0.
   DO 55 J = 1,N
55 RR = RR+T(J,I)*T(J,II)
   W(I,II) = RR
54 CONTINUE
   DO 56 I = 1,N
   DO 56 II = 1,N
   RR = 0.
   DO 57 J = 1,N
57 RR = RR+U(I,J)*W(J,II)
56 T(I,II) = RR
   DO 58 I = 1,N
   DO 58 II = 1,N
   RR = 0.
   DO 59 J = 1,N
59 RR = RR+T(I,J)*U(II,J)
   V(I,II) = RR
58 CONTINUE
36 NR = JDE
   RETURN
   END
```

APPENDIX E

GRAM-SCHMIDT PSEUDOINVERSION SUBROUTINE

The Gram-Schmidt procedure for calculating the Penrose pseudoinverse of a matrix of equations has been converted into a FORTRAN subroutine. The listing for this subroutine appears on the following pages.

```
      SUBROUTINE GINV2(A,U,AFLAG,ATEMP,MR,NR,NC,NR1,EPS)
      DOUBLE PRECISION A(MR,NC),U(NC,NC),AFLAG(NC),ATEMP(NC)
      DOUBLE PRECISION FAC,DOT,DOT1,DOT2,TOL,DSQRT
      DO 10 I = 1,NC
      DO 5 J = 1,NC
    5 U(I,J) = 0.
   10 U(I,I) = 1.
      FAC = DOT(MR,NR,A,1,1)
      FAC = 1./DSQRT(FAC)
      DO 15 I = 1,NR
   15 A(I,1) = A(I,1)*FAC
      DO 20 I = 1,NC
   20 U(I,1) = U(I,1)*FAC
      AFLAG(1) = 1.
      N = 56
      NR1 = NC
      TOL = (10.**EPS*.5**N)**2
      DO 100 J = 2,NC
      DOT1 = DOT(MR,NR,A,J,J)
      JM1 = J-1
      DO 50 L = 1,2
      DO 30 K = 1,JM1
   30 ATEMP(K) = DOT(MR,NR,A,J,K)
      DO 45 K = 1,JM1
      DO 35 I = 1,NR
   35 A(I,J) = A(I,J)-ATEMP(K)*A(I,K)*AFLAG(K)
      DO 40 I = 1,NC
   40 U(I,J) = U(I,J)-ATEMP(K)*U(I,K)
   45 CONTINUE
   50 CONTINUE
      DOT2 = DOT(MR,NR,A,J,J)
      IF ((DOT2/DOT1)-TOL) 55,55,70
   55 DO 60 I = 1,JM1
      ATEMP(I) = 0.
      DO 60 K = 1,I
   60 ATEMP(I) = ATEMP(I)+U(K,I)*U(K,J)
      DO 65 I = 1,NR
      A(I,J) = 0.
      DO 65 K = 1,JM1
   65 A(I,J) = A(I,J)-A(I,K)*ATEMP(K)*AFLAG(K)
      AFLAG(J) = 0.
      FAC = DOT(NC,NC,U,J,J)
      FAC = 1./DSQRT(FAC)
      NR1 = NR1-1
      GO TO 75
   70 AFLAG(J) = 1.
      FAC = 1./DSQRT(DOT2)
   75 DO 80 I = 1,NR
   80 A(I,J) = A(I,J)*FAC
      DO 85 I = 1,NC
   85 U(I,J) = U(I,J)*FAC
  100 CONTINUE
      DO 130 J = 1,NC
      DO 130 I = 1,NR
      FAC = 0.
      DO 120 K = J,NC
```

```
120 FAC = FAC+A(I,K)*U(J,K)
130 A(I,J) = FAC
    RETURN
    END
```

```
      DOUBLE PRECISION FUNCTION DOT(MR,NR,A,J,K)
      DOUBLE PRECISION A(MR,1),X
      X = 0.D0
      DO 50 I = 1,NR
      X = X+A(I,J)*A(I,K)
   50 CONTINUE
      DOT = X
      RETURN
      END
```

APPENDIX F

HOUSEHOLDER'S ALGORITHM SUBROUTINE

The Householder algorithm for obtaining a least squares solution is available as a FORTRAN subroutine in the IBM system 360 scientific subroutine package. The listing appears on the following pages.

```
      SUBROUTINE DLLSQ(A,B,M,N,L,X,IPIV,EPS,IER,AUX)
C
      DIMENSION A(1),B(1),X(1),IPIV(1),AUX(1)
      DOUBLE PRECISION A,B,X,AUX,PIV,H,SIG,BETA,TOL
      DOUBLE PRECISION DSQRT
C     ERROR TEST
      IF(M-N)30,1,1
C
C     GENERATION OF INITIAL VECTOR S K   K 1,2,...,N  IN STORAGE
C     LOCATIONS AUX K   K 1,2,...,N
    1 PIV=0.D0
      IEND=0
      DO 4 K=1,N
      IPIV(K)=K
      H=0.D0
      IST=IEND+1
      IEND=IEND+M
      DO 2 I=IST,IEND
    2 H=H+A(I)*A(I)
      AUX(K)=H
      IF(H-PIV)4,4,3
    3 PIV=H
      KPIV=K
    4 CONTINUE
C
C     ERROR TEST
      IF(PIV)31,31,5
C
C     DEFINE TOLERANCE FOR CHECKING RANK OF A
    5 SIG=DSQRT(PIV)
      TOL=SIG*ABS(EPS)
C
C
C     DECOMPOSITION LOOP
      LM=L*M
      IST=-M
      DO 21 K=1,N
      IST=IST+M+1
      IEND=IST+M-K
      I=KPIV-K
      IF(I)8,8,6
C
C     INTERCHANGE K-TH COLUMN OF A WITH KPIV-TH IN CASE KPIV.GT.K
    6 H=AUX(K)
      AUX(K)=AUX(KPIV)
      AUX(KPIV)=H
      ID=I*M
      DO 7 I=IST,IEND
      J=I+ID
      H=A(I)
      A(I)=A(J)
    7 A(J)=H
C
C     COMPUTATION OF PARAMETER SIG
    8 IF(K-1)11,11,9
    9 SIG=0.D0
```

```
      DO 10 I=IST,IEND
   10 SIG=SIG+A(I)*A(I)
      SIG=DSQRT(SIG)
C
C     TEST ON SINGULARITY
      IF(SIG-TOL)32,32,11
C
C     GENERATE CORRECT SIGN OF PARAMETER SIG
   11 H=A(IST)
      IF(H)12,13,13
   12 SIG=-SIG
C
C     SAVE INTERCHANGE INFORMATION
   13 IPIV(KPIV)=IPIV(K)
      IPIV(K)=KPIV
C
C     GENERATION OF VECTOR UK IN K-TH COLUMN OF MATRIX A AND OF
C     PARAMETER BETA
      BETA=H*SIG
      A(IST)=BETA
      BETA=1.D0/(SIG*BETA)
      J=N+K
      AUX(J)=-SIG
      IF(K-N)14,19,19
C
C     TRANSFORMATION OF MATRIX A
   14 PIV=0.D0
      ID=0
      JST=K+1
      KPIV=JST
      DO 18 J=JST,N
      ID=ID+M
      H=0.D0
      DO 15 I=IST,IEND
      II=I+ID
   15 H=H+A(I)*A(II)
      H=BETA*H
      DO 16 I=IST,IEND
      II=I+ID
   16 A(II)=A(II)-A(I)*H
C
C     UPDATING OF ELEMENTS J STORED IN LOCATION AUX J
      II=IST+ID
      H=AUX(J)-A(II)*A(II)
      AUX(J)=H
      IF(H-PIV)18,18,17
   17 PIV=H
      KPIV=J
   18 CONTINUE
C
C     TRANSFORMATION OF RIGHT HAND SIDE MATRIX B
   19 DO 21 J=K,LM,M
      H=0.D0
      IEND=J+M-K
      II=IST
      DO 20 I=J,IEND
```

```
      H=H+A(II)*B(I)
   20 II=II+1
      H=BETA*H
      II=IST
      DO 21 I=J,IEND
      B(I)=B(I)-A(II)*H
   21 II=II+1
C     END OF DECOMPOSITION LOOP
C
C
C     BACK SUBSTITUTION AND BACK INTERCHANGE
      IER=0
      I=N
      LN=L*N
      PIV=1.D0/AUX(2*N)
      DO 22 K=N,LN,N
      X(K)=PIV*B(I)
   22 I=I+M
      IF(N-1)26,26,23
   23 JST=(N-1)*M+N
      DO 25 J=2,N
      JST=JST-M-1
      K=N+N+1-J
      PIV=1.D0/AUX(K)
      KST=K-N
      ID=IPIV(KST)-KST
      IST=2-J
      DO 25 K=1,L
      H=B(KST)
      IST=IST+N
      IEND=IST+J-2
      II=JST
      DO 24 I=IST,IEND
      II=II+M
   24 H=H-A(II)*X(I)
      I=IST-1
      II=I+ID
      X(I)=X(II)
      X(II)=PIV*H
   25 KST=KST+M
C
C
C     COMPUTATION OF LEAST SQUARES
   26 IST=N+1
      IEND=0
      DO 29 J=1,L
      IEND=IEND+M
      H=0.D0
      IF(M-N)29,29,27
   27 DO 28 I=IST,IEND
   28 H=H+B(I)*B(I)
      IST=IST+M
   29 AUX(J)=H
      RETURN
C
C     ERROR RETURN IN CASE M LESS THAN N
```

```
   30 IER=-2
      RETURN
C
C     ERROR RETURN IN CASE OF ZERO-MATRIX A
   31 IER=-1
      RETURN
C
C     ERROR RETURN IN CASE OF RANK OF MATRIX A LESS THAN N
   32 IER=K-1
      RETURN
      END
```