X-551-69-73
PREPRINT

NASA TM X- 63570

LEAST SQUARES AND PSEUDOINVERSION

EUGENE J. LEFFERTS

SEPTEMBER 1, 1968

GSFC

GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

X-551-69-73

LEAST SQUARES

AND

PSEUDOINVERSION

Eugene J. Lefferts

Mission and Systems Analysis Branch
Mission and Trajectory Analysis Division

September 1, 1968

GODDARD SPACE FLIGHT CENTER
Greenbelt, Maryland 20771

LEAST SQUARES AND PSEUDOINVERSION

Eugene J. Lefferts
Mission and Systems Analysis Branch
Mission and Trajectory Analysis Division

SUMMARY

The purpose of post flight orbit and trajectory analysis is to establish those parameters and procedures which contribute to the uncertainty in the computed orbit. The two main techniques for post flight analyses are differential correction procedures and regression analyses.

The differential correction procedure is a method by which an orbit is corrected by interatively performing a least squares fit to the differences between computed and measured observations. The actual measured quantities must be corrected as a result of station location uncertainties, bias in the data, and so on.

Regression analysis is performed upon the residuals from the differential correction procedure. Models of station and measurement errors are obtained by means of a least squares fit to these residuals.

Inherent in both of these procedures is the need for inverting the matrix associated with the least squares fit. In most applications these matrices tend to be poorly conditioned and the resulting inversion process becomes prone to computational errors. Thus an investigation of these errors is a prerequisite to successful post flight analyses.

In order to determine the extent of errors introduced by computational procedures, various inversion and generalized inversion (pseudoinversion) methods were applied in the determination of a least square polynomial fit to data generated by a polynomial. The use of the generalized inverse and some of its computational advantages are demonstrated. In particular it is shown that if the formation of the normal equations is avoided one can obtain equivalent accuracy using single precision arithmetic as one would obtain using double precision arithmetic with conventional routines. An added advantage of the generalized inversion routines over conventional techniques is that one always receives a positive indication of when computational difficulties are encountered. Using conventional techniques the identification of polynomial fits of order higher than seven are doubtful for single precision arithmetic. This order can be extended to eleven with double precision methods. In contrast generalized inversions routines give meaningful results for orders as high as the twentieth.

CONTENTS

ILLUSTRATIONS

ILLUSTRATIONS (Continued)

TABLES

ACKNOWLEDGMENT

The Mission Analysis Section of the Bendix Field Engineering Corporation, Programming and Analysis Support Office, is gratefully acknowledged for their assistance in the preparation of this paper.

LEAST SQUARES AND PSEUDOINVERSION

INTRODUCTION

Inherent in most orbit determination and regression analyses is the need for inverting the matrix that is associated with the normal equations. In most applications, such matrices tend to be poorly conditioned, and the inversion process becomes computationally difficult to perform with accuracy in limited precision computers. The source of this difficulty and some of its effects are discussed in this paper.

MATHEMATICAL MODEL

The mathematical model for the problems to be discussed is given by the linear system

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

where A is an n × m matrix, Y is an n vector, and X is an m vector. In general, the system is overdefined; that is, n is much greater than m. It is desired to obtain a solution, $\vec{X} = \vec{X}_0$, such that the scalar function

$$G(\vec{X}) = (A\vec{X} - \vec{Y})^T W(A\vec{X} - \vec{Y}) \tag{2}$$

is minimized over all choices of $\vec{X}$. In this equation, W is an arbitrarily preassigned m × m positive definite matrix of weights. By means of the calculus, a necessary condition for an extremem of $G(\vec{X})$ is obtained by requiring the vanishing of the gradient of G with respect to $\vec{X}$. Thus:

$$\nabla_{\vec{X}} G(\vec{X}) = 2A^T W(A\vec{X} - \vec{Y}) = 0. \tag{3}$$

This gives rise to the normal equations

$$A^T WA\vec{X} = A^T W\vec{Y}, \tag{4}$$

with a solution given by

$$\vec{X}_0 = (A^T WA)^{-1} A^T W\vec{Y}, \tag{5}$$

providing that the indicated inversion can be performed.

In most practical applications, the vector $\vec{Y}$ represents a physical measurement, $\vec{\beta}$, corrupted by noise, $\vec{\eta}$. This noise is generally postulated to be characterized by the statistical properties

$$\vec{Y} = \vec{\beta} + \vec{\eta},$$

$$E\{\vec{Y}\} = \vec{\beta}, \tag{6}$$

$$E\{(\vec{Y} - \vec{\beta})\ (\vec{Y} - \beta)^T\} = E\{\vec{\eta}\,\vec{\eta}^T\} = Q, \tag{7}$$

where Q is an $m \times m$ positive definite matrix, called the covariance matrix, associated with observations. The covariance matrix associated with the solution $\vec{X}_0$ is given by P, where

$$E\{\vec{X}_0\} = \left\{E\ (A^T WA)^{-1} A^T W(\vec{\beta} + \eta)\right\}$$

$$= (A^T WA)^{-1} A^T W\, E\{\vec{\beta} + \vec{\eta}\}$$

$$= (A^T WA)^{-1} A^T W\vec{\beta}$$

and

$$P = E\left\{(\vec{X}_0 - E\{\vec{X}_0\})\ (\vec{X}_0 - E\{\vec{X}_0\})^T\right\}$$

$$= E\left\{(A^T WA)^{-1} A^T W\vec{\eta}\,\vec{\eta}^T WA(A^T WA)^{-1}\right\}$$

$$= (A^T WA)^{-1} A^T W E\{\vec{\eta}\,\vec{\eta}^T\} WA(A^T WA)^{-1}$$

$$= (A^T WA)^{-1} A^T WQWA(A^T WA)^{-1}.$$

Two particular cases in practice are considered. In one case, W is assumed to be the identity matrix, and the solution and covariance matrix are given by

$$\vec{X}_0 = (A^T A)^{-1} A^T \vec{Y}, \tag{9}$$

$$P = (A^T A)^{-1} A^T Q A(A^T A)^{-1}. \tag{10}$$

This is the so called least squares solution. The second case is when $W = Q^{-1}$. In this case, the solution and covariance matrix are defined by

$$\vec{X}_0 = (A^T Q^{-1} A)^{-1} A^T Q^{-1} \vec{Y}, \tag{11}$$

$$P = (A^T Q^{-1} A)^{-1}. \tag{12}$$

PROBLEM STATEMENT

It would appear that the problem is completely solved and that nothing further need be considered. Unfortunately, this is not the case. In most practical applications, implementation of Equations (9) through (12) may lead to drastically poor results.

As an example, consider the set of data y_i, x_i that is generated by the quadratic[1]

$$y = 40x + 10x^2. \tag{13}$$

We will assume that y is to be fitted by the polynomial

$$y = a_0 + a_1 x + \ldots + a_n x^n,$$

[1]Rice, W. and Lefferts, E., "Interference Reduction Techniques for Nonlinear Devices," Quarterly Progress Report No. 1, Contract No. DA-36-039-AMC-02208 CEI, ER-13171, Martin-Marietta Corp., Baltimore, Maryland.

where the coefficients are obtained by a least squares fit to noiseless data. Using a single precision routine (SHARE 7.0.027), we obtain the results that are listed in Table 1.

As shown in the table, even for this simple example, complete recovery of the coefficients is steadily worsened as the number of terms is increased. This poor recovery for such small dimension matrices may be surprising, and one may be tempted to attribute the results to the inadequacy of the program employed. However, this is not the case: poor recovery is characteristic of most least squares programs. This is amply demonstrated in Table 2, where three such programs are compared for a seventh degree fit to the above quadratic, Equation (13).

The difficulties encountered in the above example could be eliminated by employing a higher degree of precision in the computer. But the use of double precision is not a panacea. Besides giving false confidence in the computer results, double precision only displaces the problem since it reappears as the dimension of the vector to be recovered increases. In double precision using conventional inversion routines, this limit is reached at about dimension fourteen.

MATRIX CONDITIONING

The main difficulty encountered in the solution of the normal equations for a least squares fit lies in the poor conditioning of the associated matrix to be inverted. The concept of a poorly conditioned matrix is one that requires some clarification. For nonsingular matrices, such a concept is developed by Forsythe[2].

The norm of a vector, $\vec{Y}$, is given by

$$||\vec{Y}|| = (\vec{Y}^T\vec{Y})^{1/2} = (\vec{Y}_1^2 + \vec{Y}_2^2 + \ldots + \vec{Y}_n^2)^{1/2}.$$

For any square matrix, A, the spectral norm $||A||$ is defined by

$$||A|| = \underset{||\vec{X}||=1}{MAX} ||A\vec{X}||.$$

[2]Forsythe, George E., "Today's Computational Methods of Linear Algebra" Siam Review, July 1967, Vol. 9, No. 3, pp 489-515.

Table 1

Single Precision Recovery of a Quadratic

Coefficient	True Value	Recovery by SHARE Program 7.0.027				
		Second Order	Third Order	Fourth Order	Fifth Order	Sixth Order
a_0	0	-4.02×10^{-6}	-3.357×10^{-5}	-1.373×10^{-4}	-3.508×10^{-4}	-6.74×10^{-4}
a_1	40	40.000056	40.000710	40.003935	40.013277	40.035749
a_2	10	9.9999401	9.9981028	9.9813397	9.8990217	9.5794269
a_3	0		1.262×10^{-3}	2.88×10^{-2}	.2745	1.80559
a_4	0			-1.4097×10^{-2}	-.3067	-3.4891
a_5	0				.1204	3.1014
a_6	0					-1.032

Table 2

Comparison of Seventh Order Fits to a Quadratic

Coefficient	True Value	Recovery by SHARE Program 7.0.027	Recovery by SHARE Program 7.0.002	Recovery by SHARE Program 9.4.015
a_0	0	3.75×10^{-4}	2.98×10^{-3}	1.60×10^{-2}
a_1	40	40.02	39.85	+39.16
a_2	10	9.75	12.053	+22.10
a_3	0	11.55	-11.37	-36.00
a_4	0	-4.24	30.99	+78.00
a_5	0	6.68	-44.20	-11.00
a_6	0	-4.38	31.60	-42.00
a_7	0	1.25	-8.94	26.00

For a nonsingular matrix, A, the condition of A is defined by

$$\text{Cond. } A = ||A|| \cdot ||A^{-1}||.$$

One of the main results in using the concept of the condition of a matrix is that it can be applied to reflect the variations in the solution that are due to perturbations either in the matrix A or its nonhomogeneous side. Let a system be given by

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

If B is perturbed to $\vec{B} + \delta\vec{B}$, then $\vec{X}$ is perturbed to $\vec{X} + \delta\vec{X}$. The relative change in $\vec{X}$ is given by

$$\frac{||\delta\vec{X}||}{||\vec{X}||} \leq \text{cond. } A \frac{||\delta\vec{B}||}{||\vec{B}||}.$$

If A is subjected to a change, δA, then we have

$$\frac{||\delta\vec{X}||}{||\vec{X} + \delta\vec{X}||} \leq \text{cond. } A \frac{||\delta A||}{||A||}.$$

Unfortunately, the condition of a matrix is not easily determined, and one must rely on other procedures to determine how good an inverse actually is. Such a procedure has been given by D. Morrison[3]. His quick version, which may be computed by hand, is given as follows. Let $A = (a_{ij})$ and $A^{-1} = (a^{ij})$. Let the exponents of the diagonal terms of A be $E_1, E_2 \ldots, E_n$ and let the exponents of the diagonal terms of A^{-1} be $F_1, F_2 \ldots, F_n$.

Then if

$$L = \underset{i}{MAX} (E_i + F_i),$$

the number of digits lost in inversion is L, and the number of good digits in single precision is given by 8 - L.

PSEUDOINVERSION

For poorly conditioned matrices, many authors have suggested the use of a pseudoinverse rather than a conventional inverse. Pseudoinversion has an added advantage in that it works even when the matrix to be inverted is singular.

The pseudoinverse, or generalized inverse, performs the same function for singular matrices as an inverse performs for nonsingular ones. The generalized inverse A^+ may be defined for all matrices, including rectangular ones, in terms of two postulates.

Postulate 1: $A^+ AA^+ = A^+$.

Postulate 2: $AA^+ A = A$.

---

[3]Morrison, D. D., "How Bad is a Matrix," Space Technology Laboratories, Inc., Interoffice Correspondence, May 19, 1965.

Unfortunately, the generalized inverse is not uniquely defined by postulates 1 and 2. However, there are many ways of adding constraints to insure the uniqueness. One such set of constraints defines the Penrose Pseudoinverse (see Appendix A), which is the one used in this paper. These added constraints, or restrictions, are given by two axioms.

Axiom 1: $(AA^+)^T = AA^+$.

Axiom 2: $(A^+A)^T = A^+A$.

For the Penrose pseudoinverse, we will use the symbol $A^\#$.

The main utility of the Penrose pseudoinverse lies in the following result. Let the system of equations be given in the form

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

and let

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

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

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

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

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

Thus, the solution given by the Penrose pseudoinverse is a least squares solution. If there exists more than one least squares solution, the Penrose pseudoinverse gives the shortest vector in norm which is a solution.

In terms of the generalized inverse, Equations (5), (8), (9), (10), (11), and (12) may be rewritten as follows:

$$\vec{X}_0 = (A^T WA)^\# A^T W\vec{Y} \qquad (14)$$

$$P = (A^T WA)^{\#} A^T WQWA(A^T WA)^{\#} \tag{15}$$

$$\vec{X}_0 = (A^T A)^{\#} A^T \vec{Y} \tag{16}$$

$$P = (A^T A)^{\#} A^T QA(A^T A)^{\#} \tag{17}$$

$$\vec{X}_0 = (A^T Q^{-1} A)^{\#} A^T Q^{-1} \vec{Y} \tag{18}$$

$$P = (A^T Q^{-1} A)^{\#} \tag{19}$$

COMPUTATIONAL EXPERIMENTS

To determine whether there are computational advantages in using a pseudo-inverse, a comparison was made between inverse derived by the Gauss-Jordon technique and a pseudcinverse derived with the use of an Andree algorithm. (A description of the Andree algorithm appears in Appendices B and C.)

The symmetric and positive definite test matrix A was defined as follows:

$$A = BB^T,$$

where

$$B = (b_{ij})$$

and

$$b_{ij} = \frac{(-1)^j (i-1)!}{(i-j)!\,(j-1)!}, \quad i \geq j$$

$$b_{ij} = 0, \quad i < j .$$

The inverse matrix A^{-1} was given by

$$A^{-1} = B^T B .$$

The comparison was performed on a Univac 1108 computer in double precision. All elements of A were integers, so no round-off was generated upon their entry into the computer. The results were then tabulated and summarized as represented in Table 3 and Figures 1 and 2.

The comparison of the two inverses shows that the Gauss-Jordon inverse provides slightly better results than the pseudoinverse. At the point where there was a computational loss of rank, the pseudoinverse bears no resemblance to the conventional inverse. In general, the pseudoinverse is not a continuous function of its elements, and it changes considerably when its rank is reduced. For example, consider:

$$A = \begin{pmatrix} 1 & 1 \\ 1 & a \end{pmatrix}.$$

Then,

$$A^{\#} = \frac{\begin{pmatrix} a & -1 \\ -1 & 1 \end{pmatrix}}{|A|} = A^{-1} \text{ for } a \neq 1$$

and

$$A^{\#} = \begin{pmatrix} 1/4 & 1/4 \\ 1/4 & 1/4 \end{pmatrix} \text{ for } a = 1.$$

Therefore, rather than compare pseudoinverses to inverses, it is more fruitful to compare recoverable solutions to a set of equations.

The second computational experiment was a comparison between the solutions of a least squares polynomial fit to data generated by the equation

$$y = 1 + 10x + x^2.$$

The solutions were obtained using the Gauss-Jordon inverse and the Andree algorithm pseudoinverse of the normal equations. The results are as follows.

Table 3

Loss of Significant Digits and Symmetry in Matrix Inversion

n	Lost Digits Predicted	Lost Digits		Symmetry	
		Gauss-Jordon Method	Andree Algorithm Method	Gauss-Jordon Method	Andree Algorithm Method
4	1	1	2	15	16
5	2	2	3	14	16
6	3	2	3	13	16
7	3	4	4	12	16
8	4	5	5	11	16
9	5	6	6	10	16
10	6	7	7	10	16
11	7	8	8	9	16
12	8	9	9	7	16
13	9	9	—	7	16
14	10	11	—	6	16
15	11	12	—	5	16
16	12	13	—	4	16
17	12	14	—	3	16
18	13	14	—	3	16
19	14	15	—	2	16
20	15	16	—	1	16

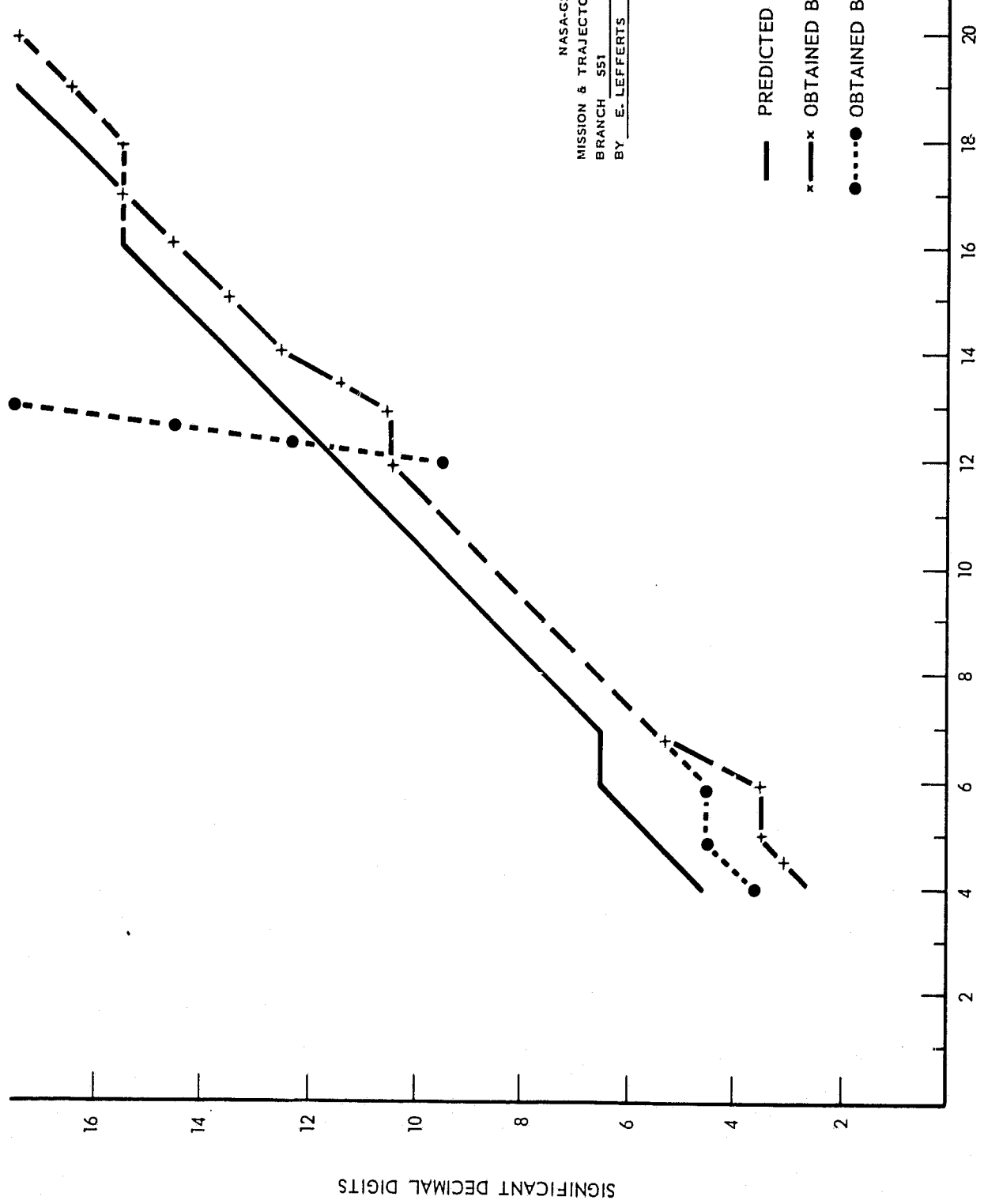

Figure 1. Loss of Significant Decimal Digits in Matrix Inversion

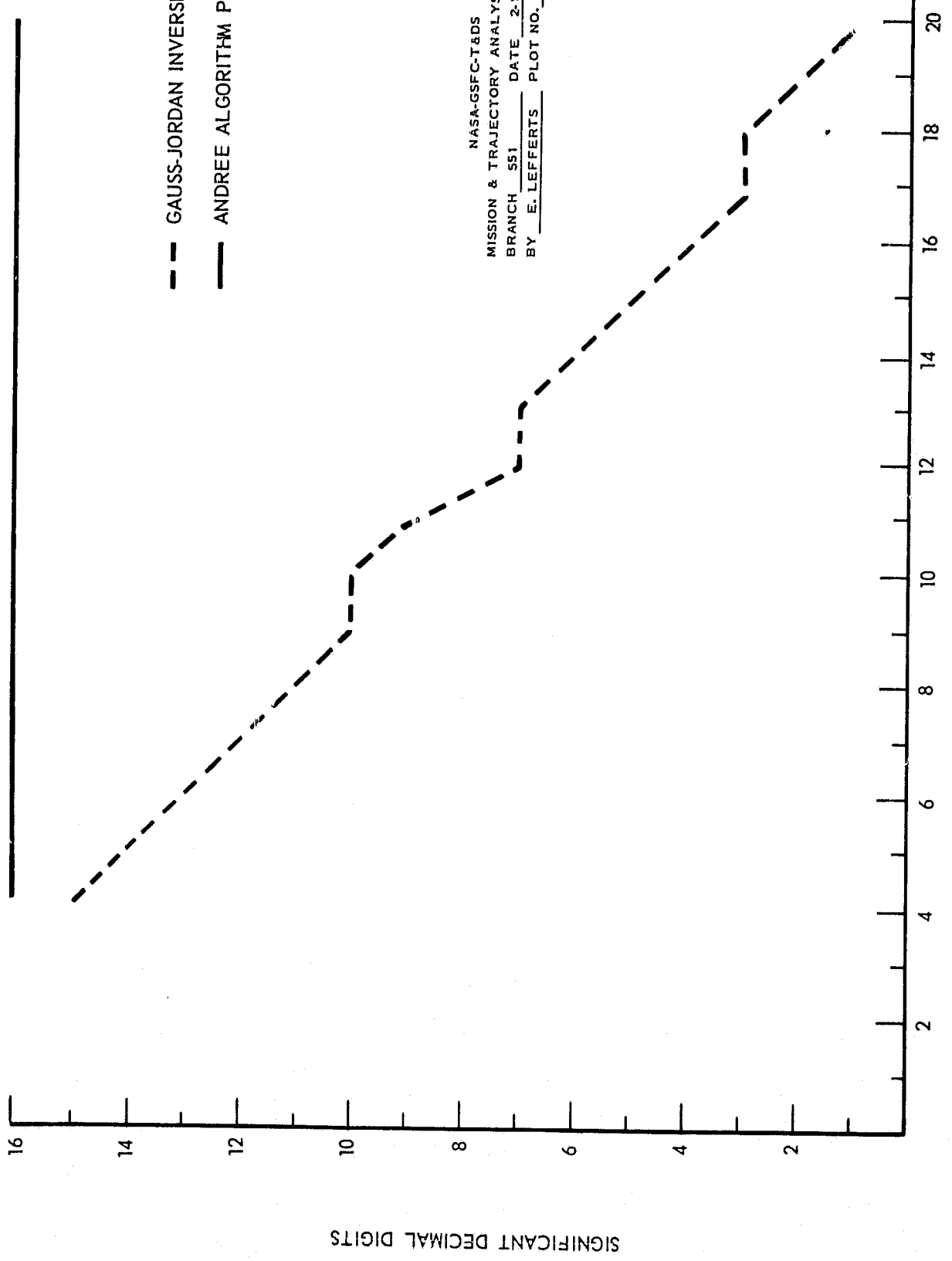

Figure 2. Number of Significant Decimal Digits of Symmetry Versus Dimension

The Gauss-Jordon fit is slightly better for polynomials up to the tenth order. For fits from the eleventh order to the twentieth, which is as far as the investigation was carried, the Gauss-Jordon recovery is markedly inferior. It is interesting to note that from the eleventh order polynomial onward the computational rank of the pseudoinverse is consistently less than maximal. Table 4 and Figure 3 list and chart the errors in the recovery for both procedures.

The loss of rank in the Andree solution may be caused by the loss of positive definiteness in the formation of the matrix of the normal equations, or it could be due directly to computational errors in the inversion algorithm. To investigate which of these is the source of trouble, a second pseudoinverse algorithm, based upon a Gram-Schmidt orthogonalization procedure, was investigated. (Appendices B and D provide a description of the Gram-Schmidt algorithm.) The Gram-Schmidt algorithm computes the pseudoinverse of a matrix directly, without the need of forming the normal equations. However, it can be applied to the normal equations to enable comparison with other procedures.

When applied to the rectangular matrix, the Gram-Schmidt pseudoinverse was found to be markedly superior to both the Gauss-Jordon inverse and the Andree algorithm pseudoinverse. (See Table 4.) In fact it could do as well in single precision as the other two inverses could do in double precision. When the Gram-Schmidt method was applied to the normal equations, it was found to be inferior to the Andree algorithm. Thus, it appears that the formation of the normal equations is extremely critical in least squares solutions.

One of the main advantages in the pseudoinversion routines is the manner in which the computational rank is controlled. In the Andree algorithm, a series of congruent transformations is made. In each step, the reduction is forced to pivot about the largest diagonal element. If at any step of the process the largest remaining diagonal element is less than the product of a preassigned constant, K, and the first pivotal element, then it and all remaining diagonal elements are set to zero. Thus, the rank is depressed. The constant K is chosen to provide n significant figures in the ratio between the pivotal element under consideration and the largest pivotal element. Even when K is set to zero, the rank is reduced if any diagonal element becomes negative.

In the Gram-Schmidt orthogonalization process, the rank is reduced by comparing the size of the angle between any column vector and its projection on the independent column space already determined. This procedure does not preserve the non-negativeness of the matrix when applied to the normal equations.

A comparison was made in which the rank controlling factor was varied. The constant K was chosen as

Table 4

Comparison of Error in Eighth, Eleventh, and Sixteenth Order Fits

n	a_n	Eighth Order Fit $(a_n - b_n)$			Eleventh Order Fit $(a_n - b_n)$			Sixteenth Order Fit $(a_n - b_n)$		
		Gauss-Jordon Method	Andree Algorithm Method	Gram-Schmidt Method	Gauss-Jordon Method	Andree Algorithm Method	Gram-Schmidt Method	Gauss-Jordon Method	Andree Algorithm Method	Gram-Schmidt Method
0	1	-6.6×10^{-10}	2.38×10^{-9}	-1.4×10^{-14}	-9.8E - 7	3.1E - 8	-1.9E - 14	3.4E - 6	2.3E - 7	9.0E - 16
1	10	1.2×10^{-7}	-1.5×10^{-7}	-3.4×10^{-14}	9.2E - 5	-2.9E - 6	7.0E - 14	5.0E - 3	-1.3E - 5	-6.7E - 12
2	1	-1.3×10^{-6}	2.4×10^{-6}	8.1×10^{-12}	-3.2E - 3	8.6E - 5	7.3E - 12	5.8E - 3	1.6E - 4	1.5E - 10
3	0	-1.14×10^{-5}	-1.72×10^{-5}	-3.6×10^{-12}	4.8E - 2	1.4E - 3	1.5E - 10	1.05E - 1	-1.7E - 3	-4.6E - 9
4	0	2.29×10^{-5}	$5.[illegible] \times 10^{-5}$	1.96×10^{-10}	-1.8E - 1	7.8E - 3	-1.2E - 10	-8.1E - 1	8.0E - 2	5.2E - 8
5	0	-8.3×10^{-5}	$[illegible] \times 10^{-4}$	-3.64×10^{-10}	8.1E - 1	9.8E - 3	2.1E - 9	-1.26E + 1	-7.4E - 2	-2.4E - 7
6	0	3.4×10^{-5}	1.61×10^{-4}	4.07×10^{-10}	2.3E - 0	5.1E - 2	-5.6E - 9	7.4E + 1	6.6E - 1	1.7E - 6
7	0	-5.4×10^{-5}	-1.08×10^{-4}	-2.33×10^{-10}	4.4E - 0	3.2E - 2	3.5E - 9	9.4E + 1	6.8E - 1	-1.2E - 4
8	0	9.5×10^{-6}	1.72×10^{-5}	3.00×10^{-11}	-1.20E - 1	4.9E - 3	-7.5E - 9	6.4E + 1	-3.9E - 2	2.3E - 5
9	0				3.56E - 0	2.7E - 2	9.6E - 9	2.4E + 1	-1.9E - 1	9.1E - 5
10	0				-2.2E - 0	4.2E - 2	-1.4E - 8	-1.8E + 2	-2.6E - 1	1.7E - 4
11	0				1.99E - 1	4.4E - 4	2.1E - 9	2.1E + 2	-4.2E - 1	-3.6E - 4
12	0							-2.8E + 2	1.0E - 1	2.0E - 4
13	0							8.1E + 1	-3.5E - 2	-1.9E - 4
14	0							1.5E + 2	-3.9E - 2	6.4E - 5
15	0							1.5E + 1	3.9E - 3	-2.0E - 5
16	0							2.2E - 0	4.6E - 2	3.6E - 6

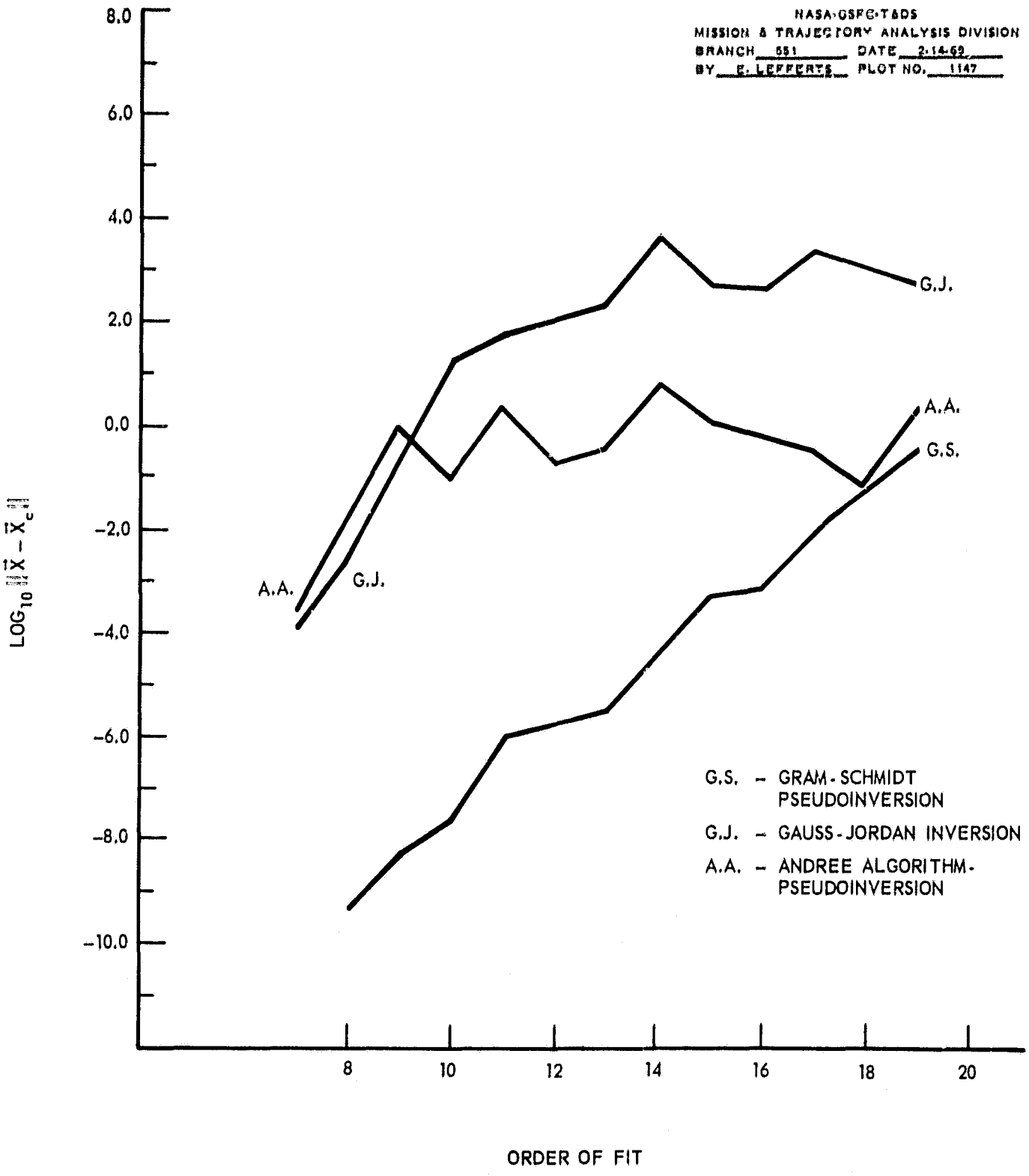

Figure 3. Error in Solution Vector Versus Order of Fit

$$K = 2^{-54} \cdot 10^{n},$$

and n was varied from 0 to 15. For n = 0, the rank of the matrix of normal equations was depressed by two in the Angree algroithm, while the Gram-Schmidt method gave maximal rank. This is due probably to the fact that non-negativeness of the matrix is forced as a constraint in the Angree algorithm. As n was increased, the computational rank was decreased.

As the rank is decreased, the recovery is improved and then worsened. (See Figure 4.) The initial improvement is due to the cancellation of computational errors. The subsequent worsening is probably due to the rejection of observable information.

In Figure 5 the computational rank is plotted against the parameter n. In Figure 6 the norm of the error in the recovery vector is plotted versus the parameter n. Observe that the recovery given by the Andree algorithm corresponding to a given rank is independent of n, while the recovery in the Gram-Schmidt process is different even though the rank stays fixed. This is due to the manner in which the Gram-Schmidt process was implemented. Procedures are available to insure the uniqueness of the recovery for a given rank.

Observe from Figures 5 and 6 that for n = 0, 1, 2, and 3 the two Gram-Schmidt recoveries are of maximum rank, but the ratio of error vectors is 10^8. This error difference can be due only to the formation of the normal equations.

CONCLUSIONS AND RECOMMENDATIONS

On the basis of the preliminary studies described in this report, practical solutions to least squares problems are somewhat in doubt due to the poor conditioning of the associated matrices. Even though these results were demonstrated for the impractical problem of high degree polynomial fitting, the conclusions are valid for many other problems. For example, recovery of station parameters based upon a regression analysis from one or two satellite passes leads to matrices which are in many cases even more poorly conditioned.

The results of this investigation lead to the conclusion that the formation of the normal equations is extremely critical and should be avoided where possible. Unfortunately, due to the large number of data points which must be processed in practical problems, this leads to very large matrices with the incumbent storage limitations of modern computers. Thus, in practice, one is forced to treat the normal system of equations. Here, the pseudoinverse of Penrose leads

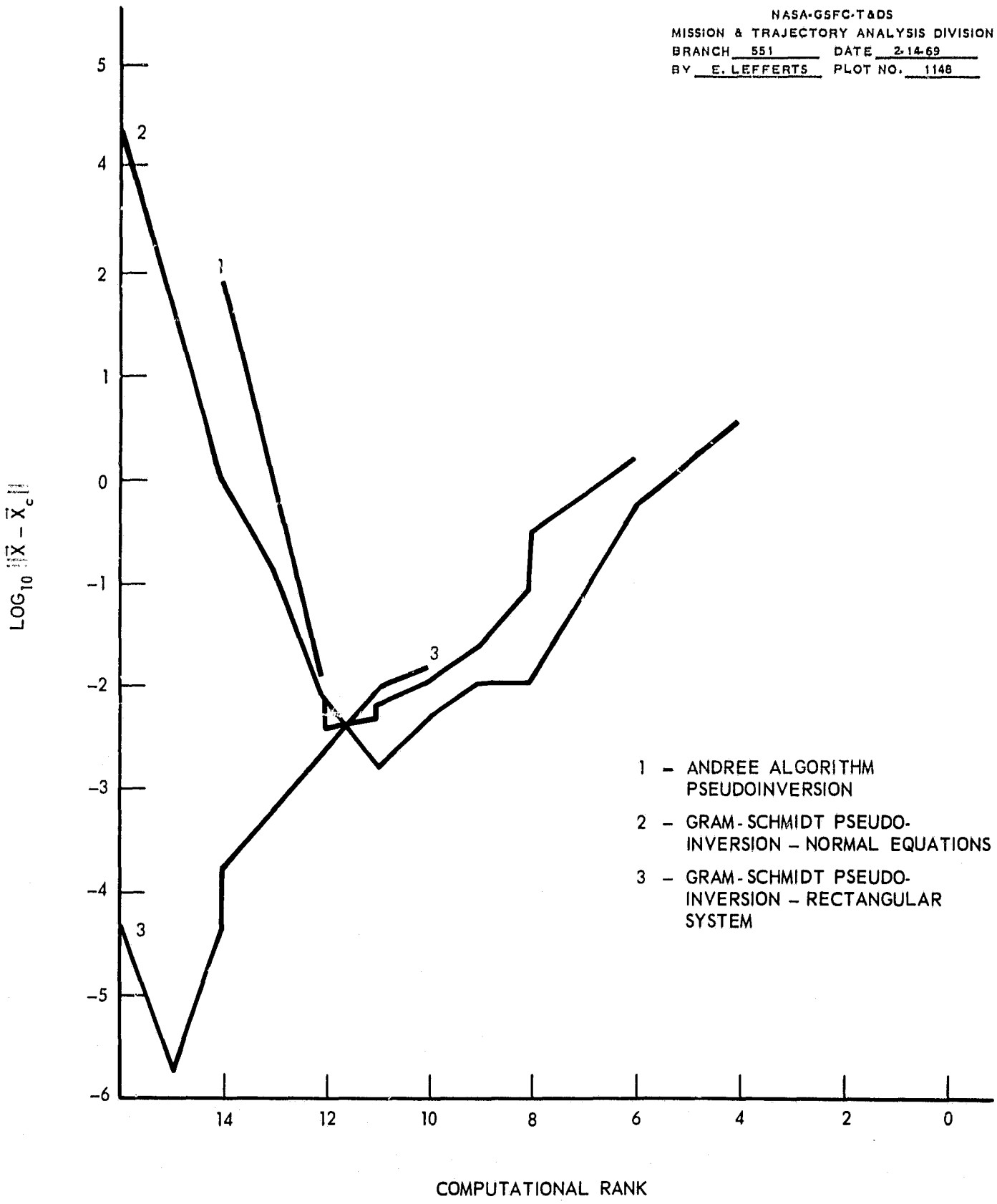

Figure 4. Error in Solution Vector Versus Computational Rank

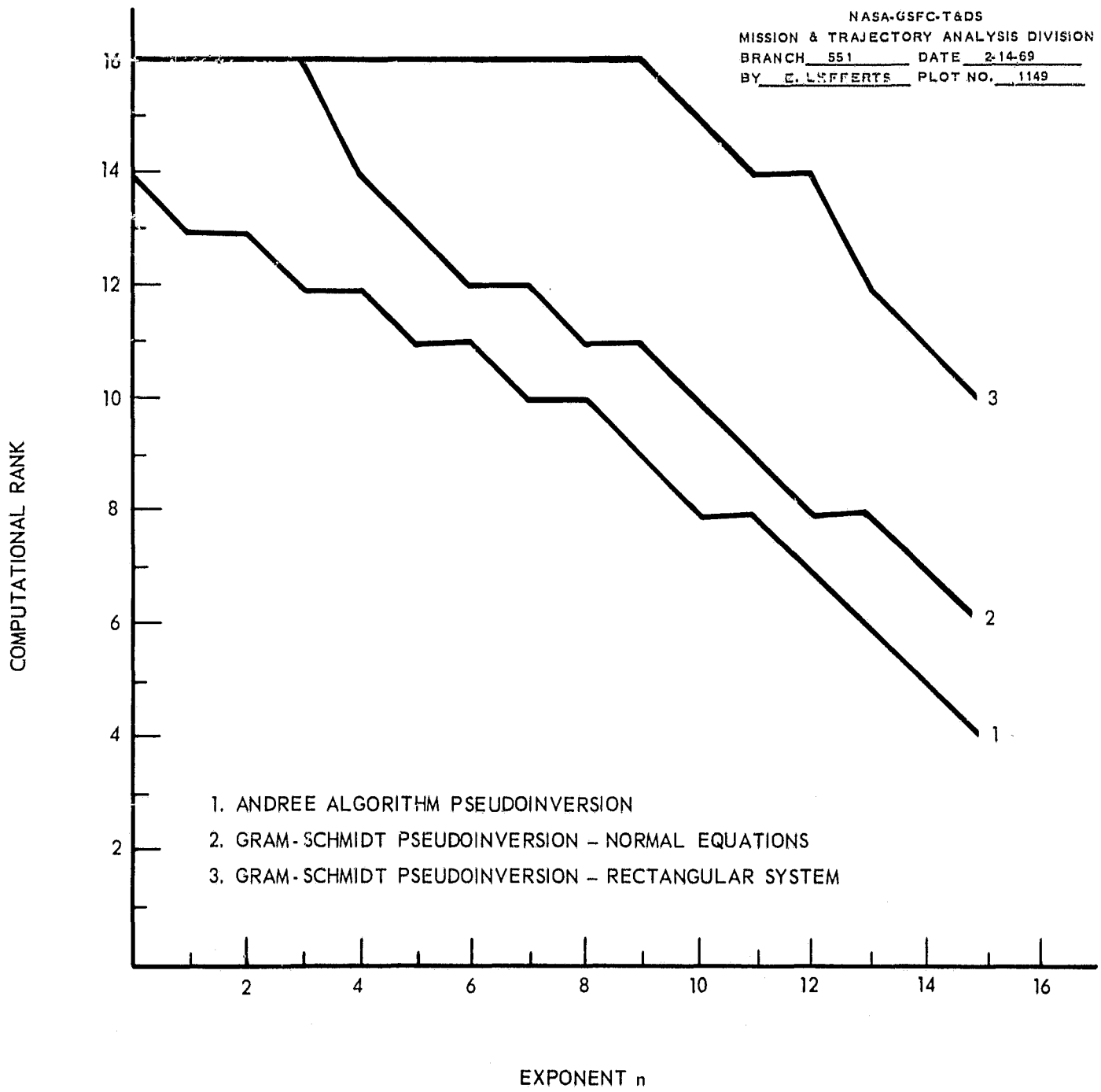

Figure 5. Computational Rank Versus Control Parameter

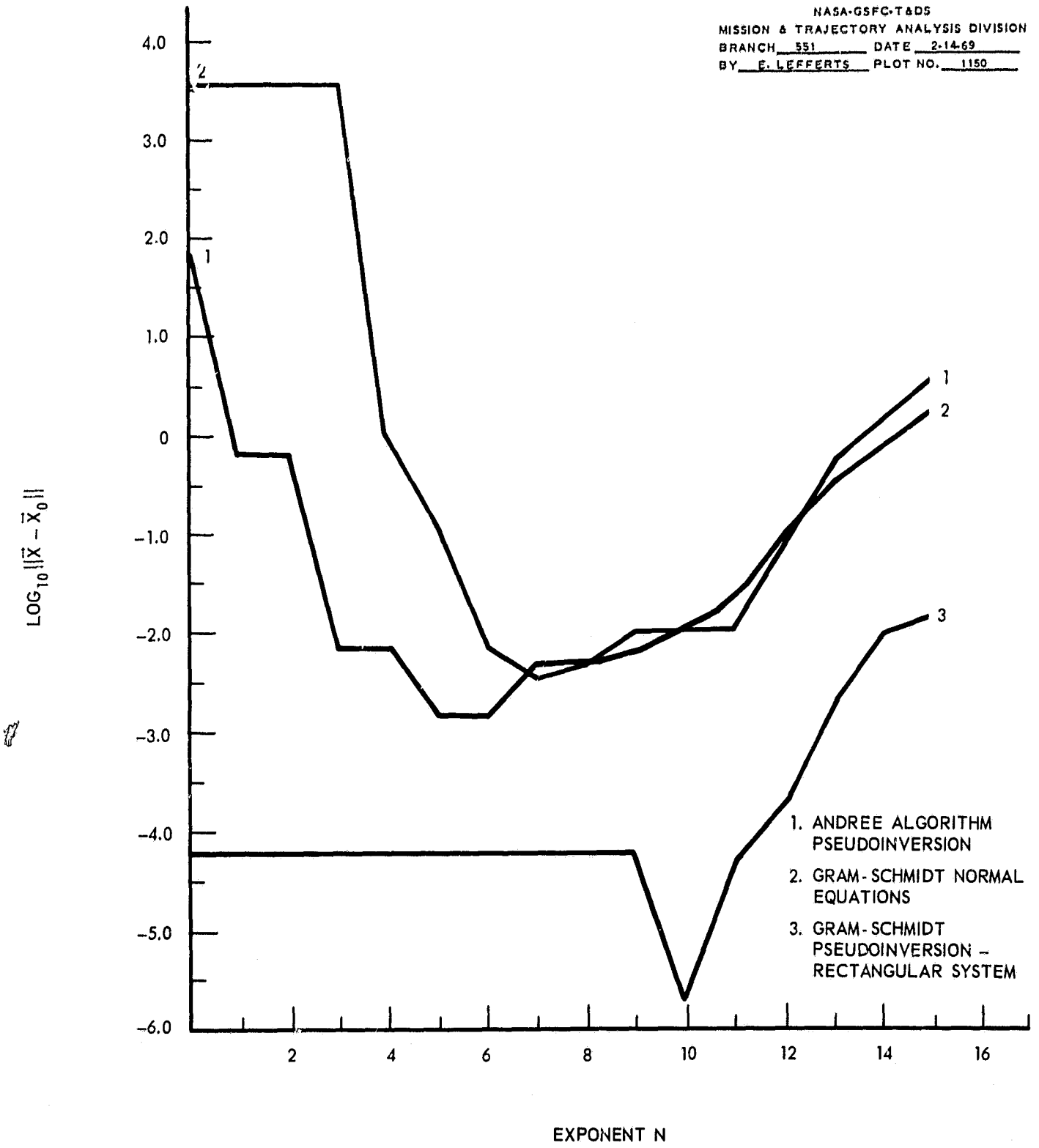

Figure 6. Error in Solution Vector Versus Control Parameter

to solutions of greater confidence. Inherent in the use of pseudoinversion routines is that an indication of computational difficulty is provided. This is evident in the reduction of computational rank from maximal.

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 = diag (λ_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) $AA^\#A = S^T DSS^T D^\# SS^T DS^\# = S^T DD^\# DS = S^T DS = A.$

(2) $A^\#AA^\# = S^T D^\# SS^T DSS^T D^\# S = S^T D^\# DD^\# S = S^T D^\# S = A^\#.$

(3) $(AA^\#)^T = (S^T DSS^T D^\# S)^T = (S^T DD^\# S)^T = S^T (DD^\#)^T S = S^T DD^\# S$

$= S^T DSS^T D^\# S = AA^\#.$

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

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

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

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

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

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,

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

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

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

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

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

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

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

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

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,NP,EPS)
      DOUBLE PRECISION V,B,T,W,X,U,P,DR,DSQRT
      DIMENSION V(30,30),B(30,30),R(30),T(30,30),W(30,30),X(30,30)
      DIMENSION U(30,10)
      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
      75 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(J,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
         LEF = 0
         GO TO 66
```

```
      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 = JOE
         RETURN
         END
```

END

APPENDIX D

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,DET,DET1,DET2,TOL,DSQRT
      DO 10 I = 1,NC
      DO 5 J = 1,NC
    5 U(I,J) = 0.
   10 U(I,I) = 1.
      FAC = DET(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
      DET1 = DET(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
      DET2 = DET(MR,NR,A,J,J)
      IF ((DET2/DET1)-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 = DET(NC,NC,U,J,J)
      FAC = 1./DSQRT(FAC)
      NR1 = NR1-1
      GO TO 75
   70 AFLAG(J) = 1.
      FAC = 1./DSQRT(DET2)
   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
```

```
FN IV G LEVEL 1, MOD 1              GINV2              DATE = 68180        09/27/23

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

```
IN IV G LEVEL 1, MOD 1              LOT                    DATE = 68180          09/20/23

      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
```