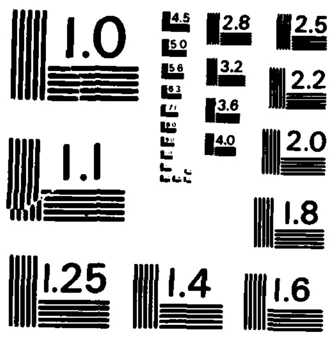
AD-A163 840 A NEW METHOD FOR COMPUTING THE GENERALIZED INVERSES OF A MATRIX AND ITS A..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. C F MURRAY 1/2

UNCLASSIFIED DEC 85 AFIT/GOR/MA/85D-2 F/G 12/1 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

1

A NEW METHOD FOR COMPUTING THE GENERALIZED INVERSES OF A MATRIX AND ITS APPLICATION TO THE LYAPUNOV MATRIX EQUATION

THESIS

Craig F. Murray
Captain, USAF

AFIT/GOR/MA/85D-2

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GOR/MA/85D-2

1

A NEW METHOD FOR COMPUTING THE GENERALIZED INVERSES OF A MATRIX AND ITS APPLICATION TO THE LYAPUNOV MATRIX EQUATION

THESIS

Craig F. Murray
Captain, USAF

AFIT/GOR/MA/85D-2

AFIT/GOR/MA/85D-2

A NEW METHOD FOR COMPUTING THE GENERALIZED INVERSES OF A MATRIX AND ITS APPLICATION TO THE LYAPUNOV MATRIX EQUATION

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Masters of Science in Computer Systems

Craig F. Murray
Captain, USAF

December 1985

Preface

This work covers a unique combination of theory and application in two fields, mathematics and computer science. It address a subject with a wide variety of uses in an equally wide variety of fields. Thus the content and particularly the results of this work should appeal to a large audience.

The subject I'm referring to is the computation of four generalized inverses. The significance of this treatment is its improvement over existing methods for computing the generalized inverses of a matrix. The computational method developed here in theory and with application provides a simple, direct, and unified approach for computing four generalized inverse. Probably more importantly, this treatment emphasizes the properties of each generalized inverse relative to its computation and its potential use. For some readers, hopefully this combination will provide the missing insight needed to recognize a use for this method with the proper generalized inverse over a current approach.

Naturally, with an effort this size, there are a number of people who deserve credit and a word of thanks. I am deeply indebted to my thesis advisor, Dr J. Jones Jr., for his interest, patience, and total assistance in completing this effort. Certainly Dr. Jones personifies all the qualities of a great educator and scholar, especially with his committment and enthusiasm. I also thank Capt Steve Woffinden for his comments which help make my thesis a more understandable document.

Craig F. Murray

Table of Contents

List of Figures

AFIT/GOR/MA/85D-2

Abstract

This ~~work~~ thesis examines a new method for computing four generalized inverses of a matrix. This method, the ST method, is based on the careful selection of a sequence of matrix multiplications and partitionings which provide a new foundation for computing four generalized inverses. Central to this approach is the partitioning of the two submatrices, R and C, where the product of their submatrices will give the generalized inverses of interest. Thus using this new representation, the generalized inverses of a matrix can be computed in a simple and direct manner.

In this work, four generalized inverses are derived and computed in a systematic manner from this representation. These results are strongly tied to the solution of the matrix equation $Ax=b$ where the general solution is given in terms of this new representation and computational technique. The computational technique is presented with an example and also as an algorithm. Included with the algorithm is an analysis of its computer implementation. The use of one generalized inverse is used to find the solution of a Lyapunov matrix equation.

I Introduction

Background

The classical inverse of a matrix, A, is another matrix, usually denoted A^{-1}, which satisfies the equation $AA^{-1} = A^{-1}A = I$. For the matrix A to have an inverse, in this sense, one important restriction must be met. The matrix must be a nonsingular square matrix. In many instances, this restriction cannot be met. For example, in systems theory, data fitting, statistics, and other areas, rectangular and singular matrices arise which can not be inverted using the classical method to solve the respective problem (3; 5; 7:15; 19). Obviously, in these cases the classical inverse can not be computed so what must be done to solve the respective problems? In the case of rectangular or singular matrices another inverse, or more precisely class of inverses can be computed and used to find a possibly more robust solution. This class of inverses is called the generalized inverses of a matrix.

The history of this class of matrices referred to as generalized inverses can probably be succinctly summarized by noting five key developments. The first of these developments is creditted to Fredholm who, it is believed, introduced the concept of a generalized inverse in 1903 (3:4). He is also believed to have introduced the term pseudoinverse to describe his new concept (3:4). Following Fredholm about 20 years later was the developments set forth by E. H. Moore. Moore was the first to define a unique generalized inverse for every finite matrix. Moore called this generalized inverse the general reciprocal (3:5). While the progress in this new field continued following the publication of Moore's work, the next significant development came another 30 years later when Bjerhammar showed the relationship of a generalized inverse to a linear system of equations (3:5). This was extended four years later by Penrose who showed that the generalized inverse, defined by Moore, was unique. In doing this, Penrose defined this unique inverse with the following four equations

$$AXA = A \tag{1.1}$$

$$XAX = X \tag{1.2}$$

$$(AX)^* = (AX) \tag{1.3}$$

$$(XA)^* = (XA) \tag{1.4}$$

where X is the unique generalized inverse of the matrix A (3:7). This unique generalized inverse is popularly referred to as the Moore-Penrose inverse. After Penrose's work, the next major development is really a series of developments starting with the work of Rao who defined a new generalized inverse which did not satisfy all the conditions that Penrose introduced with Eqs (1.1) through (1.4) (23:viii). Instead this inverse satisfied a combinations of Penrose's equations. Other generalized inverses followed which satisfied fewer or different combinations of the four equations and were, at the same time, more robust. The major draw back of these developments was the lack of a direct, unified, and numerically stable method for computing the generalized inverses whose existence were proved theoretically. This problem lead to the fifth major development when a new method, the ST method, was introduced by J. Jones, Jr. in 1984. This latest development will be documented in detail in this work emphasizing its computability.

Characterizing the Generalized Inverses of a Matrix

From the historical notes on the generalized inverses of matrices, it should be apparent that the development of this class of matrices suffered a rather piecemeal and slow growth. This has lead to an equally diverse nomenclature used to denote these inverses. Where the classical inverse of the matrix A was simply denoted as a A^{-1} and defined by the equation $AA^{-1} = A^{-1}A = I$, the same can not be said for the generalized inverse of a matrix. Instead of the single equation found in the classical case, there is a range of equations for the generalized inverses of a matrix where each identifies a particular property a given inverse satisfies. This lead to the wide range of names which are currently used to refer to these inverses. These names include pseudoinverse, Moore-Penrose inverse, reflective generalized inverse, g-inverse, left weak generalized inverse, and right weak generalized inverse. Additionally, Ben-Israel and Greville in their work (3:18), list about 24 other names used to describe an inverse satisfying some

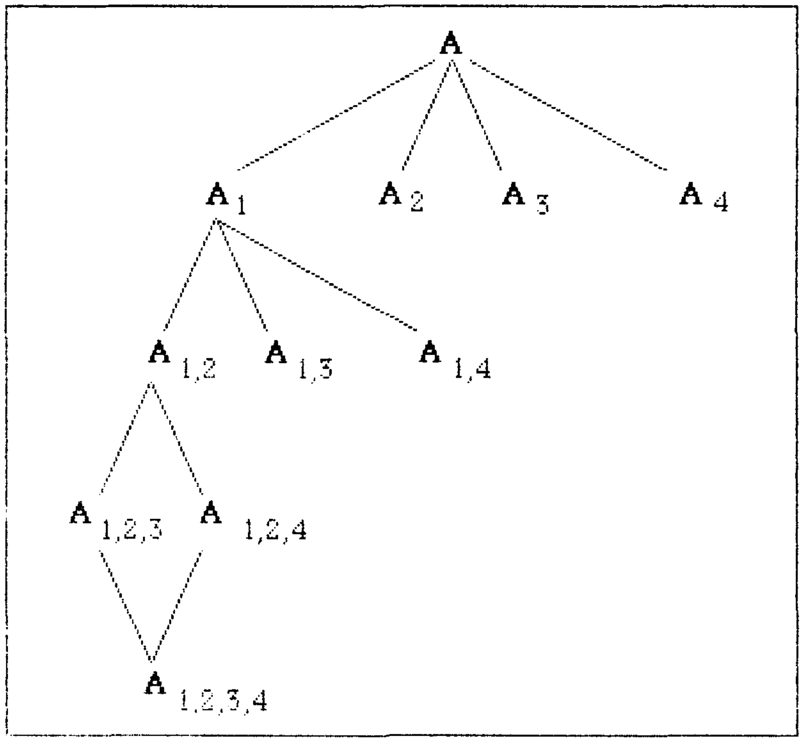


Figure 1. The Generalized Inverses of a Matrix

combination of Penrose's four equations. Thus the initial problem to overcome, when beginning a discussion on the theory of generalized inverses, is choosing a suitable method for describing the inverses of concern. One means of overcoming this problem is by adopting a naming convention which is both simple and direct and also incorporates the properties of a given inverse. Starting with the given matrix A, such a method is illustrated in figure 1 which graphically depicts a partial class of generalized inverses. This method builds from the given matrix A up to the inverse satisfying all four of Penrose's equations, Eqs (1.1) through (1.4). Thus as it travels from the least restricted generalized inverses to the unique generalized inverse, $A_{1,2,3,4}$, the combinations which arise are denoted by the trailing subscripts where each subscript refers to one of the four equations developed by Penrose. The subscripts listed are equated to the conditions satisfied by a particular

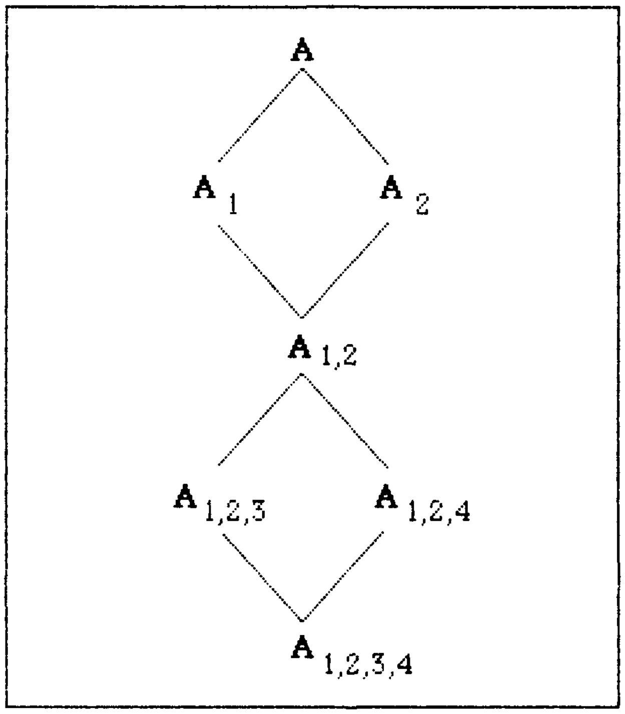


Figure 2. Subset of Inverses

generalized inverse. To prevent confusion, the simple relationship between subscripts and the four equations is illustrated below.

Equation	Corresponding Subscript
$(AXA) = A$	1
$(XAX) = X$	2
$(AX)^* = AX$	3
$(XA)^* = XA$	4

Again, the X in each equation denotes the appropriate inverse. While figure 1 shows most of the possible generalized inverses, this work will concentrate on the subset of these inverses shown in figure 2 of the previous page. Again, the subscripts following each A denote which of Penrose's four equations the inverse satisfies. This nomenclature will be used throughout this work.

Problem

Current methods for computing the generalized inverses of a matrix are cumbersome, and many are also numerically unstable (19:247). None were found which provide a unified method for calculating more than one generalized inverse. A new method for computing these inverses, the ST method, is a direct and numerically stable technique which represents a marked improvement over previous ones in use. This work will review the theoretical basis of the ST method, and go on to define both the characteristics of its computer implementation and how this new method of computation represents an improvement over past methods.

Scope

In this work, the ST method is considered across the field of complex numbers. For the treatment of theory, this means matrices with complex elements will be used. In terms of the technique's computer implementation, this translates into matrices of constant elements.

In describing and defining the technique's computer implementation, three areas will be examined. First the numerical properties of this technique will be treated. This will be followed by a time and space analysis of the algorithm arising from the ST method. Finally, this technique will be compared to other current techniques to establish its claimed improvements.

Approach and Presentation

Chapter II begins the review of the theoretical basis of the ST method for computing the generalized inverses of a matrix with complex elements. The theory begins with a new computation representation and continues to present a systematic development of four generalized inverses. Each theory is developed with as much detail as possible to insure maximum clarity. An example is given early during this treatment to motivate and assist the understanding of the theorems presented.

Chapter III applies the theorems developed in chapter II to the realm of the computer. Here a detailed description of the technique is given in an algorithmic manner versus the theoretical basis of chapter II. In this regard, the algorithm for the ST method is examined to determine its numerical characteristics and analyze its computer time and space requirements. This is followed by an introduction to other techniques for computing the generalized inverses of a matrix, and these are compared to the ST method in terms of numerical stability and computer requirements. Finally, the ST method is adapted to an existing algorithm available in many computer math packages to compute an $A_{1,2,3,4}$ generalized inverse.

Chapter IV concludes the heart of this work by extending the comparisons developed in chapter III to an actual application of the ST method. In this regard, the ST method is used to find the solution of a Lyapunov matrix equation. To do this, additional theorems are developed to show how the ST method is applied to the Lyapunov matrix equation. Finally, an A1,2 generalized inverse is developed in the common solution for a Lyapunov matrix equation.

Chapter V completes this work with a summary of its important points and recommendations for further study in this area.

II Theory of Generalized Inverses

As mentioned earlier, the theory of generalized inverses has been introduced under a variety of names, including the pseudoinverse and the Moore-Penrose inverse. Common to each approach, no matter what the name, is an emphasis on the generalized inverse which satisfies the four equations:

$$AXA = A \tag{2.1}$$

$$XAX = X \tag{2.2}$$

$$(AX)^* = AX \tag{2.3}$$

$$(XA)^* = XA \tag{2.4}$$

Subsequent to such an introduction, this common approach continues by introducing a variety of properties arising from the pseudoinverse or Moore-Penrose inverse satisfying Eqs (2.1) through (2.4), and then gradually moves toward a more complete discussion of the other generalized inverses of a matrix. Such an approach concludes by finally presenting a method for computing the generalized inverses again, emphasizing the one inverse satisfying Eqs (2.1) through (2.4).

The approach to the theory of generalized inverses presented in this work will be different. Instead of the common path taken by many authors in the past, this approach presents a systematic development of four generalized inverses with their respective properties and their computability. Additionally, this discussion of the generalized inverses of a matrix is strongly oriented toward solving the system of equations often denoted as $Ax=b$ where the matrix A does not have a classical inverse. Central to this orientation is an emphasis on computation which will be presented hand in hand with theory. This is intended to provide both an understanding of theory and the application of the theory. Thus when the equivalent of the psuedo- inverse or Moore-Penrose inverse is finally reached, its properties and computability arises more directly. Unfortunately this approach shares one common problem with its predecessors, the problem of nomenclature. Since the means of describing a generalized inverse of a matrix varies, as noted in chapter 1, a few additional comments are devoted to more completely defining the notation used in this work. Other remarks regarding notation used throughout this work are also included at this point to prevent confusion and emphasize the important characteristics of the material which follows. Additionally, since this work is geared to the practical adaptation of a generalized inverse to solving real world problems, a section will be devoted to the basic operations used to compute the generalized inverses, namely elementary row and column operations. For the reader unfamiliar with these operations, this review will be helpful for understanding the computations involved in finding the generalized inverses of concern here. For the reader familiar with row and column operations, their mention here should provide an insight to the directness of the computation to be discussed.

Notation

The remarks concerning the notation used throughout this work deal with both the exact expressions used to denote the generalized inverses of interest and the mathematical expressions used to derive these inverses. Both are discussed below.

As mentioned in chapter 1, the generalized inverses of a matrix have been denoted in a variety of ways. In this work the following representation will be used

$A_{i,j,k,l}$

where the subscripts denote the equations the given generalized inverse satisfies. Thus a generalized inverse satisfies Eqs (2.1) and (2.2) would be denoted as

$A_{1,2}$

A generalized inverse satisfying all four equations would be denoted as

$A_{1,2,3,4}$

This notation provides an easy and direct method for classifying and describing the generalized inverses of a given matrix.

The second notational concern deals with the mathematical expressions used throughout this work. First, the symbol, Ç, is used to denote the field of complex numbers, and $Ç^{mxn}$ denotes the vector space of m x n complex matrices over Ç. Second, matrices are represented by capital letters, and the small letters b, w, x, y, and z denote vectors in this work. Thus, A, would denote a matrix A of unspecified dimensions. Third, an asterisk denotes the conjugate transpose of a matrix. Thus the conjugate transpose of A would be denoted as A^*. The final comment on notation again involves the representation of matrices. When depicting a large matrix with multiple submatrices, the matrix will be enclosed in brackets, and the submatrices denoted as capital letters. In some cases, the submatrices will also be separated by partitioning lines. This practice is only used in a few cases to emphasize the composition of a matrix. Normally, the partitioning of a large matrix will be denoted using the submatrices depicted in capital letters, sufficient spacing, and the enclosing brackets. When numeric values are listed as the components of a matrix the meaning of the above conventions changes slightly. If the value is a zero, then it may represent the real number zero or the zero matrix. While this may sound confusing, the context of such an occurrence will clarify its meaning. Other occurrences of numeric values in matrices do depict individual matrix elements.

Elementary Row and Column Operations

In the next section, theorem 2-1 will provide the basis for computing the generalized inverses of a matrix. To understand how to utilize this result, in a computational sense, a knowledge of elementary row and column operations is required. Further an understanding of these operations will help develop an appreciation for the computationally direct and simple process used to obtain these generalized inverses. Thus as either a review or quick

$$\begin{bmatrix} 1 & 1 & 3 \\ 2 & 2 & 2 \\ 3 & 3 & 1 \end{bmatrix}$$

Original Matrix

$$\begin{bmatrix} 3 & 3 & 1 \\ 2 & 2 & 2 \\ 1 & 1 & 3 \end{bmatrix}$$

Matrix After Interchanging Row 1 and Row 3

Figure 3. Interchanging Rows

overview, the following summary of elementary row and column operations will help, at a minimum, appreciate the computational aspects of the processes at the heart of this work.

Interchanging Rows or Columns. Given a matrix, any complete row of elements may be interchanged with the corresponding elements of any other complete row of the same matrix. The new matrix is equivalent to the original matrix. Using an arbitrary matrix, this operation is demonstrated in figure 3 where the first and third rows are interchanged. The same conceptual operation can also be performed on columns. Now instead of interchanging the corresponding elements of two rows, the corresponding elements of two columns are interchanged. Had this operation been applied to the matrix in figure 3, interchanging columns one and three would not have produced a matrix where the first column consisted of a three, two, and one. Interchanging columns one and two would not have made a noticeable difference since each column is identical.

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$$

Original Matrix

$$\begin{bmatrix} 3 & 3 & 3 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$$

Matrix After Multiplying Row 1 by Scalar k=3

Figure 4 Multiplication by a Scalar

Multiplication of a Row or Column. Given a matrix, the elements of any row in the matrix can be multiplied by a scalar. The resulting matrix is equivalent to the original matrix. Again, using the same arbitrary matrix, this operation is demonstrated below in figure 4 where row 1 is multiplied by scalar k=3. This same operation may be performed on columns as well as rows. Thus had the elements of column 1 been multiplied by k=3 the resulting column would contained 3, 6, and 9 from top to bottom respectively.

Multiply a Row or Column and Add. Using the same matrix from the previous illusrtation, a row is again multiplied by a scalar and the resulting new row is now added to one of the remaining rows of the matrix. As before, the addition is performed

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$$

Original Matrix

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 0 & 0 & 0 \end{bmatrix}$$

Matrix After Multiplying Row 1 by Scalar k=(-3)
and then Adding Row 1 to Row 3

Figure 5. Multiplication by a Scalar and Addition

on only corresponding elements of each row. The results of multiplying row 1 by k=(-3) and then adding this new row to row 3 are shown in figure 5. Note, row 1 is unchanged by this operation and row 3 is eliminated. The same operation can be performed on a column basis. Using column operation and the same original matrix from figure 5, column 1 could be used to eliminate the elements of column 3 by multiplying its elements by k=(-1) and then adding corresponding elements of columns 1 and 3. Had this same operation been performed on the resepective columns of figure 3, only the second element in column three would have been. eliminated.

The Generalized Inverses of a Matrix (10)

The systematic development of the four generalized inverses begins with the construction of a new and equivalent representation of the arbitrary matrix A which has dimensions m rows and n columns. The starting point of this systematic development is then the matrix A augmented with two identity matrices of appropriate dimensions and shown in the following equation

$$\begin{bmatrix} A & I \\ I & 0 \end{bmatrix} \tag{2.5}$$

Using this representation, the the four generalized inverses of interest will be developed in theory and in practice. This representation and consequently the generalized inverses derived from it will also be closely tied to the general solution of a system of equations. This is all accomplished through seven theorems and one example. Theorem 2-1 begins by developing the new representation from Eq (2.5). This is immediately followed by an example which demonstrates how each of the four generalized inverses of interest is computed from this new computational framework. Theorem 2-2 extends this example by introducing how the general solution of a system of equations also arises from this computational framework in terms of the generalized inverses. Theorems 2-3, 2-4, 2-5, and 2-7 develop the existence of each of the generalized inverses computed from this new representation. Finally theorem 2-6 develops the general solution of a system of equations in a manner similar to theorem 2-2 but where the submatrix M is not defined based on the rank of A.

Theorem 2-1. For any given matrix $A \in Ç^{m x n}$, there exist two nonsingular matrices, $R \in Ç^{mxm}$ and $C \in Ç^{nxn}$, such that the following pair of equivalent matrices can be constructed

$$\begin{bmatrix} A & I \\ I & 0 \end{bmatrix} \quad \text{and} \quad \left[\begin{array}{cc|c} I_r & 0 & T \\ 0 & 0 & M \\ \hline S & N & 0 \end{array}\right] \tag{2.6}$$

<u>Proof</u>. For any matrix $A \in Ç^{mxn}$ there exist nonsingular matrices $R \in Ç^{mxm}$ and $C \in Ç^{nxn}$ such that

$$RAC = \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} \tag{2.7}$$

where I_r is an identity matrix of dimension equal to the rank of matrix A. Eq (2.7) is a well established result, and this result is used as it arises in the following matrix multiplications

$$\begin{bmatrix} R & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} A & I \\ I & 0 \end{bmatrix} \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} RA & R \\ I & 0 \end{bmatrix} \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} RAC & R \\ C & 0 \end{bmatrix} \tag{2.8}$$

Now matrices R and C are now defined as follows

$$R \triangleq \begin{bmatrix} T \\ M \end{bmatrix} \quad \text{and} \quad C \triangleq [S \; N] \tag{2.9}$$

Using these new forms of R and C, the desired result is obtained from Eq (2.8) giving the following

$$\begin{bmatrix} RAC & R \\ C & 0 \end{bmatrix} = \left[\begin{array}{c|c|c} I_r & 0 & T \\ \hline 0 & 0 & M \\ \hline S & N & 0 \end{array}\right] \tag{2.10}$$

■

At this point, the definition of matrices R and C in Eq (2.8) may seem somewhat arbitrary, but these new forms will play an important role in computing the various generalized inverses of the given matrix A ϵ $Ç^{mxn}$. This representation, developed in theorem 2-1, will allow the partitioning of A into the matrices S, T, M, and N using a scheme reminiscent of the classical approach to matrix inversion. These new matrices are then the foundation for computing the four generalized inverses of A as discussed above. The following example will help illustrate how the S, T M, and N matrices are obtained, and also how these matrices will form the foundations for computing the generalized inverses of interest.

<u>Example 1</u>. Given the matrix A where

$$A = \begin{bmatrix} 1.000 & 2.000 & 3.000 & 4.000 \\ 5.000 & 10.000 & 15.000 & 20.000 \end{bmatrix} \tag{2.11}$$

Augment A with an identity matrix, beside and below it, to give the representation of Eq (2.12):

$$\begin{bmatrix} A & I \\ I & 0 \end{bmatrix} = \begin{bmatrix} 1.0 & 2.0 & 3.0 & 4.0 & 1.0 & 0.0 \\ 5.0 & 10.0 & 15.0 & 20.0 & 0.0 & 1.0 \\ 1.0 & 0.0 & 0.0 & 0.0 & & \\ 0.0 & 1.0 & 0.0 & 0.0 & & \\ 0.0 & 0.0 & 1.0 & 0.0 & & \\ 0.0 & 0.0 & 0.0 & 1.0 & & \end{bmatrix} \tag{2.12}$$

Now using elementary row and column operations, reduce A to the following form where I is an identity matrix of dimension equal to the rank of A and the identity matrices of Eq (2.5) are not shown:

$$\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$$

From this point on, the identity matrix I, arising from the reduction of A, will be referred to as I_r as introduced in theorem 2-1. The form of Eq (2.12) should look familiar. It was the starting point of theorem 2-1, and it was eventually transformed it to an equivalent representation which exhibited the submatrices S, T, M, and N. This representation, Eq (2.10), will be obtained in this example by using elementary row and column operations to reduce the matrix A to the identity matrix I_r. As A is reduced, the desired submatrices will arise on the two augmented identity matrices similar to the classical method, where the original matrix is augmented with an identity matrix and then original matrix is reduced to an identity matrix. In the classical case, the result of the row operations on the identity matrix produce the classical inverse, but here both row and column operations will produce four new submatrices, two of which will be used to compute the generalized inverse of interest.

The process begins by first performing the desired row operations to eliminate the second row of A as illustrated in Eq (2.13). As the second row of A is eliminated, the results of this row operation begin to define the submatrices of interest. Next,

$$\begin{bmatrix} 1.0 & 2.0 & 3.0 & 4.0 & 0.0 & 0.2 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & -0.2 \\ 1.0 & 0.0 & 0.0 & 0.0 & & \\ 0.0 & 1.0 & 0.0 & 0.0 & & \\ 0.0 & 0.0 & 1.0 & 0.0 & & \\ 0.0 & 0.0 & 0.0 & 1.0 & & \end{bmatrix} \qquad (2.13)$$

performing the required column operations, A is reduced to I_r as shown in Eq (2.14):

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.2 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & -0.2 \\ 1.0 & -2.0 & -3.0 & -4.0 & & \\ 0.0 & 1.0 & 0.0 & 0.0 & & \\ 0.0 & 0.0 & 1.0 & 0.0 & & \\ 0.0 & 0.0 & 0.0 & 1.0 & & \end{bmatrix} \qquad (2.14)$$

The augmented version of A, Eq (2.12), has now been transformed to the equivalent representation exhibiting the submatrices S, T, M, and N. These are illustrated more explicitly in Eq (2.15):

$$\left[\begin{array}{c|c|c} I_r & 0 & T \\ \hline 0 & 0 & M \\ \hline S & N & 0 \end{array}\right] = \left[\begin{array}{c|ccc|cc} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.2 \\ \hline 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & -0.2 \\ \hline 1.0 & -2.0 & -3.0 & -4.0 & & \\ 0.0 & 1.0 & 0.0 & 0.0 & & \\ 0.0 & 0.0 & 1.0 & 0.0 & & \\ 0.0 & 0.0 & 0.0 & 1.0 & & \end{array}\right] \qquad (2.15)$$

where

$$S = \begin{bmatrix} 1.000 \\ 0.000 \\ 0.000 \\ 0.000 \end{bmatrix}$$

$$N = \begin{bmatrix} -2.000 & -3.000 & -4.000 \\ 1.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{bmatrix}$$

$$T = [0.000 \ \ 0.200]$$

$$M = [1.000 \ -0.200]$$

With the S, T, M, and N matrices computed, the next objective of this example is to form the generalized inverses of A using these new submatrices. To do this one simple rule is required. The generalized inverse is equal to the product of the submatrices S and T. Using this rule and the notation introduced earlier, the $A_{1,2}$ generalized inverses will be computed directly from the result in Eq (2.15).

The $A_{1,2}$ generalized inverse is formed from the product of the matrices S and T. Thus, the multiplication in Eq (2.16) directly and simply obtains the $A_{1,2}$ generalized inverse of A:

$$A_{1,2} = ST = \begin{bmatrix} 1.000 \\ 0.000 \\ 0.000 \\ 0.000 \end{bmatrix} [0.000 \ \ 0.200] = \begin{bmatrix} 0.000 & 0.200 \\ 0.000 & 0.000 \\ 0.000 & 0.000 \\ 0.000 & 0.000 \end{bmatrix} \quad (2.16)$$

where $A_{1,2}$ satisfies the first two conditions defined by Penrose:

$$A(ST)A = A \tag{2.17}$$

$$(ST)A(ST) = ST \tag{2.18}$$

While the $A_{1,2}$ generalized inverse was computed directly from Eq (2.15), obtaining the $A_{1,2,3}$ and $A_{1,2,4}$ generalized inverses, as one might expect, requires a little more work. Both inverses are still the product of S and T, but now the form of S and T change depending on which inverse is desired. For $A_{1,2,3}$ each row of T is orthogonalized to the rows of M. For the $A_{1,2,4}$ generalized inverse, each column of S is orthogonalized to the columns of N. When both orthogonalization processes are performed, S and T appear as shown below (To orthogonalize the rows or columns, the Gram-Schmidt

$$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.038 & 0.192 \\ 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & -0.200 \\ 0.033 & -0.077 & -0.176 & -4.000 & & \\ 0.067 & 1.000 & 0.000 & 0.000 & & \\ 0.100 & -0.231 & 1.000 & 0.000 & & \\ 0.133 & -0.308 & -0.706 & 1.000 & & \end{bmatrix}$$

orthogonalization process, was used. A more detailed explanation of this process is offered in chapter 3).

Computing the $A_{1,2,3}$ generalized inverse requires a step back from the fully orthogonalized representation shown above to Eq (2.15). To computed the $A_{1,2,3}$ inverse requires the orthogonalized T submatrix and the original S submatrix from Eq (2.15). When performed, the multiplication appears as illustrated in Eq (2.19):

$$A_{1,2,3} = ST = \begin{bmatrix} 1.000 \\ 0.000 \\ 0.000 \\ 0.000 \end{bmatrix} [0.038 \;\; 0.192] = \begin{bmatrix} 0.038 & 0.192 \\ 0.000 & 0.000 \\ 0.000 & 0.000 \\ 0.000 & 0.000 \end{bmatrix} \quad (2.19)$$

where S is the original S submatrix from Eq (2.15) and T is now an orthogonalized version of the T from Eq (2.15). This $A_{1,2,3}$ generalized inverse satisfies Eqs (2.17) and (2.18) plus the additional Penrose condition detailed in Eq (2.20):

$$(ST)(A)^* = (ST)A \quad (2.20)$$

The process for computing the $A_{1,2,4}$ generalized inverse proceeds in a similar manner. Instead of the orthogonalized version the the T submatrix and the original S submatrix, the opposite of each is used. The resulting multiplication follows in Eq (2.21)

$$A_{1,2,4} = ST = \begin{bmatrix} 0.033 \\ 0.067 \\ 0.100 \\ 0.133 \end{bmatrix} [0.0 \;\; 0.2] = \begin{bmatrix} 0.000 & 0.007 \\ 0.000 & 0.013 \\ 0.000 & 0.020 \\ 0.000 & 0.027 \end{bmatrix} \quad (2.20)$$

Again, along with Eq (2.17) and (2.18), the $A_{1,2,4}$ generalized inverse satisfies Eq (2.22):

$$(ST)^*(A) = (ST)A \quad (2.22)$$

With three out of a possible four choices used, there is only one logical combination of the S and T submatrices left for computing the final generalized inverse of interest, the $A_{1,2,3,4}$ inverse.

The $A_{1,2,3,4}$ inverse is obtained in a similar manner using the S and T submatrices, but now both orthogonalized versions are multiplied together. In this example the result is obtained by

$$A_{1,2,3,4} = ST = \begin{bmatrix} 0.033 \\ 0.067 \\ 0.100 \\ 0.133 \end{bmatrix} [0.038 \;\; 0.192] = \begin{bmatrix} 0.001 & 0.006 \\ 0.003 & 0.013 \\ 0.004 & 0.019 \\ 0.005 & 0.026 \end{bmatrix} \qquad (2.23)$$

The $A_{1,2,3,4}$ inverse satisfies all the conditions satisfied by the lower inverses plus one last condition making it the unique inverse:

$$(A(ST))^* = A(ST) \qquad (2.24)$$

A second more detailed example is provided in appendix A. This example was taken from pages 340 and 341 of Noble's <u>Applied Linear Algerbra</u> text, reference 21. All four generalized inverses are computed and each of the four conditions are tested to verify the results.

Theorem 2-1 defined a new starting point for computing the generalized inverses of a matrix. In this respect, theorem 2-1 provided both a new method of representing a matrix and a new means of partitioning a matrix to a form well suited for computing the generalized inverses of interest. Using this result, example 1 took a 2x4 matrix and reduced it to the form developed in theorem 2-1, the new starting point for computing the generalized inverses of concern. The process of reducing the matrix to the desired form was computationally simple and direct using only elementary row and column operations. From this starting point, the process was expanded to compute four generalized inverses where each inverse satisfies one or more of the four conditions. Combined, theorem 2-1 and example 1 provide a starting point and insight into how the result and process can be put to further use. Theorem 2-2 will begin to expand on this by extending the new representation and computational technique to the framework of the equation $Ax=b$.

Theorem 2-2. The system of equations defined by

$$Ax = b$$

where

$$A \in \mathbb{C}^{mxn}$$
$$x \in \mathbb{C}^{nx1}$$
$$b \in \mathbb{C}^{mx1}$$

has a solution x if and only if $Mb = 0$ and the general solution of $Ax=b$ can be given by

$$x_{\text{general solution}} = STb + Nz, \ \forall z \qquad (2.25)$$

where S, T, M, and N are the matrices defined in theorem 2-1 and z is an arbitrary variable of appropriate dimensions.

Proof. In theorem 2-1, matrices R and C were partitioned as

$$R \triangleq \begin{bmatrix} T \\ M \end{bmatrix} \text{ and } C \triangleq [S\ N] \qquad (2.9)$$

Additionally R and C are nonsingular as defined in theorem 2-1. By using this partitioning along with matrix multiplication, theorem 2-1 provided the following relationship

$$\begin{bmatrix} RAC & R \\ C & 0 \end{bmatrix} = \left[\begin{array}{cc|c} I_r & 0 & T \\ 0 & 0 & M \\ \hline S & N & 0 \end{array}\right] \qquad (2.10)$$

From these matrices and their submatrices, the following chain of implications arise

Ax = b has a solution x,

if and only if RAx = Rb has a solution x,

if and only if RACy = Rb has a solution y and x = Cy,

if and only if $\begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} w \\ z \end{bmatrix} = \begin{bmatrix} T \\ M \end{bmatrix} b$ has a solution $y = \begin{bmatrix} w \\ z \end{bmatrix}$

and $x = [S\ N] \begin{bmatrix} w \\ z \end{bmatrix}$, if and only if $\begin{bmatrix} I_r & W \\ & 0 \end{bmatrix} = \begin{bmatrix} Tb \\ Mb \end{bmatrix}$ has a solution $y = \begin{bmatrix} W \\ z \end{bmatrix}$

and $x = [SW + Nz]$,

if and only if $I_r\, W = Tb$, $0 = Mb$, and $x = [SW+Nz]$

Thus from the above chain of implications, $Mb=0$ is a consistency condition and determines if the the equation

$$x = SW+Nz \tag{2.26}$$

has a solution. Further, Eq (2.26) can be rewritten in the following form to capitalize on the results of theorem 2-1, and obtain the results stated in theorem 2-2:

$$x = S(Tb)+Nz \tag{2.27}$$

■

Corollary 1. By choosing $b=0$ in theorem 2-1, the columns of N form a basis for the null space, $\eta(A)$, where $Ax=b$ has the general solution

$$x_{\text{general solution}} = STb + Nz, \quad \forall z \tag{2.28}$$

Again, z must be of appropriate dimensions and $M^*b=0$ is a consistency condition.

Proof. By choosing $b=0$, the columns of N span $\eta(A)$. Also the columns of A come from the nonsingular matrix C. Therefor the columns of N are linearly independent and hence form a basis for $\eta(A)$.

■

With a growing understanding of the ST method from theorem 2-1 and 2-2, nothing yet has been said about how the product of the S and T submatrices satisfies combinations of the four equations defined by Penrose. The next three theorems plus theorem 2-7 will prove the existence of each generalized inverse of concern in this work relative to the results of theorem 2-1 and theorem 2-2.

Theorem 2-3. From the representation developed in theorem 2-1 and the process described in example 1, the matrix ST is a A_1 generalized inverse of A where $A_1 \in Ç^{nxm}$ and $A \in Ç^{mxn}$. The generalized inverse A_1 satisfies the following conditions:

$$(ST)A(ST) = ST = A_1$$

Proof. Using the R and C matrices defined earlier in Eq (2.7), the matrix multiplication of RAC can be written slightly differently than its form in Eq (2.8). The RAC multiplication can now be extended with the four submatrices S, T, M, and N providing

$$RAC = \begin{bmatrix} Ir & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} T \\ M \end{bmatrix} A \, [S \; N] \qquad (2.29)$$

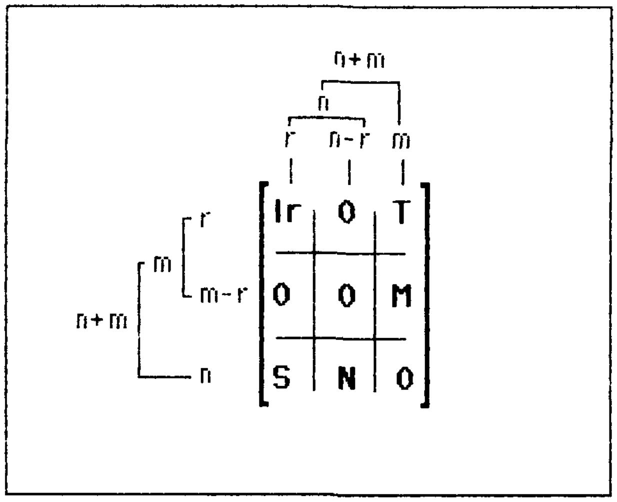


Figure 6. Matrix Dimension Table

The next step is to begin a long chain of multiplications and substitutions which can be deceiving when using matrices without clearly depicted dimensions. Figure 6 depicts the correct dimension of each matrix appearing in the subsequent multiplications. Reference figure 6 if any confusion arises regarding the conformity of a multiplication or substitution. Now applying the multiplications introduced in Eq (2.29)

$$\begin{bmatrix}T\\M\end{bmatrix} A\,[S\ N] = \begin{bmatrix}TA\\MA\end{bmatrix}[S\ N] \tag{2.30}$$

$$\begin{bmatrix}TA\\MA\end{bmatrix}[S\ N] = \begin{bmatrix}TAS & TAN\\MAS & MAN\end{bmatrix} \tag{2.31}$$

Next to see that

$$(ST)A(ST) = ST = A_1$$

holds, the result from Eq (2.31) are substituted back into Eq (2.29) giving

$$\begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} TAS & TAN \\ MAS & MAN \end{bmatrix} \qquad (2.32)$$

This relationship implies

$$TAS = I_r$$

Thus

$$(ST)A(ST) = S(TAS)T = S(I_r)T = ST = A_1$$

■

While the A_1 generalized inverse was not computed in example 1, it will be used here as an intermediate step to reaching the $A_{1,2}$ generalized inverse. Thus the next step will be to show ST also satisfies the second condition introduced in Eq (2.2). Once this is proved, ST, in this case, will be an $A_{1,2}$ generalized inverse by definition since it satisfies Penrose's first two conditions.

Theorem 2-4. The ST matrix obtained in theorem 2-3 is also an A_2 generalized inverse satisfying

$$A(ST)A = A \tag{2.17}$$

Proof. Recalling from above

$$RAC = \begin{bmatrix} Ir & 0 \\ 0 & 0 \end{bmatrix} \tag{2.7}$$

Since R and C are nonsingular, it follows that

$$A = R^{-1} \begin{bmatrix} Ir & 0 \\ 0 & 0 \end{bmatrix} C^{-1} \tag{2.33}$$

Using the new definition

$$\begin{bmatrix} Ir & 0 \\ 0 & 0 \end{bmatrix} \triangleq \beta \tag{2.34}$$

Eq (2.33) can be expressed as

$$A = R^{-1} \begin{bmatrix} Ir & 0 \\ 0 & 0 \end{bmatrix} C^{-1} = R^{-1} \beta C^{-1} \tag{2.35}$$

Now consider the following using the previously defined properties of R and C

$$C\,\beta^*\,R = [S\ N]\,\beta^* \begin{bmatrix} T \\ M \end{bmatrix} \tag{2.36}$$

$$= [S\ N] \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix}^* \begin{bmatrix} T \\ M \end{bmatrix}$$

$$= [S I_r\ 0] \begin{bmatrix} T \\ M \end{bmatrix}$$

$$= S(I_r)T = ST$$

Using these and previous results to substitute for A and ST gives

$$\begin{aligned} A(ST)A &= (R^{-1}BC^{-1})(CB^*R)(R^{-1}BC^{-1}) \\ &= R^{-1}B(C^{-1}C)B^*(RR^{-1})BC^{-1} \\ &= R^{-1}BB^*BC^{-1} \\ &= R^{-1}C^{-1} \\ &= A \end{aligned}$$

Finally combing this result on top of the previous result, the proof of theorem 2-4 follows with

$$\begin{aligned} (ST)A(AT) &= (CB^*R)(R^{-1}BC^{-1})(CB^*R) \\ &= CB^*(RR^{-1})B(C^{-1}C)B^*R \\ &= CB^*R \\ &= ST \end{aligned}$$

■

Once again, since the same ST matrix was used to obtain the generalized inverses A_1 and A_2 in theorem 2-3 and theorem 2-4 respectively, ST satisfies both conditions and is then by definition an $A_{1,2}$ generalized inverse. The same type of argument will be used to prove the existence of the $A_{1,2,3,4}$ generalized inverse. This begins with theorem 2-5 below and concludes with theorem 2-7. Theorem 2-6 interrupts the process to provide the additional results needed to prove the existence of an $A_{1,2,4}$ generalized inverse.

Theorem 2-5. If $MT^*=0$, then the matrix ST is an $A_{1,2,3}$ generalized inverse of the matrix A.

Proof. It must be shown that the following equations hold for any given $A \in Ç^{mxn}$ and $A_{1,2,3} \in Ç^{nxm}$

$$A\ A_{1,2,3}\ A = A \qquad (2.37)$$

$$A_{1,2,3}\ A\ A_{1,2,3} = A_{1,2,3} \qquad (2.38)$$

$$(A\ A_{1,2,3})^* = (A\ A_{1,2,3}) \qquad (2.39)$$

Let $A^- = ST$. It is sufficient to show that AA^- is symmetric since by the results of theorem 2-4 it is already know that

$$A\ A^-A = A$$

$$A^-A\ A^- = A^-$$

Then in order to show any matrix is symmetric, it is sufficient show that

$$(A\,A^{-})(A\,A^{-})^{*} = (A\,A^{-}) \tag{2.40}$$

In other words, A is symmetric if and only if $A = A^{*}$. Now to show that (AA^{-}) is symmetric, begin by showing $(AA^{-})(AA^{-})^{*}$ is symmetric. First note that $(PQ)^{*}= Q^{*}P^{*}$ holds for matrix multiplication of P and Q. Then taking the transpose of $(AA^{-})(AA^{-})^{*}$ as with P and Q gives

$$[(AA^{-})(AA^{-})^{*}]^{*} = ((AA^{-})^{*})^{*}(AA^{-})^{*} = (AA^{-})(AA^{-})^{*} \tag{2.41}$$

Here the transpose of the transpose of $(AA^{-})(AA^{-})^{*}$ remains unchanged. Hence the left hand side of Eq (2.40) is symmetric and consequently so is the right hand side of this equation.

Next, assume that $MT^{*}=0$ holds by hypothesis. By making use of theorem 2-2 which guarantees the existence of a solution x to the equation $Ax=b$ when $MT^{*}=0$ ($MT^{*}=0$ was the consistency condition for a general solution of $x_{\text{general solution}} = STb + Nz$). Apply the results of this theorem to the columns of T^{*}. Let b be the columns of T^{*} where b may vary as only one to all the columns of T^{*}. Then by theorem 2-2, there exists a matrix X such that $AX=b$ holds where b ranges over the columns of T^{*}, A is fixed, and X arises from each solution of $AX=b$. Now $T^{*}=AX$ implies that

$$((T)^{*})^{*} = T = (AX)^{*} = X^{*}A^{*} \tag{2.42}$$

Since $A^{-} = ST$

$$A^{-} = S(X^{*}A^{*}) \tag{2.43}$$

then for $Z = SX^{*}$

$$(SX^{*})A^{*} = ZA^{*} \tag{2.44}$$

Finally it follows from the earlier equation that

$$\begin{aligned}(A\,A^{-})A\,A^{-})^{*} &= A\,A^{-}(A^{-})^{*}A^{*} \\ &= A(A^{-}(A^{-})^{*}A^{*} \\ &= A(STA^{-*}A^{*}\ ) \\ &= ASX^{*}A^{*}A^{-*}A^{*} \\ &= AZ(A^{*}A^{-*}A^{*}) \\ &= AZ(A\,A^{-}A)^{*} \\ &= AZA^{*} \\ &= A(ZA^{*}) \\ &= AA^{-}\end{aligned} \tag{2.45}$$

So from Eqs (2.42) through (2.45), equality holds for

$$(A\,A^{-})(A\,A^{-})^{*}=(A\,A^{-}) \tag{2.46}$$

and by this equality symmetry also holds.

■

Theorem 2-6. Recall the definition of the nonsigular matrices R and C

$$R \triangleq \begin{bmatrix} T \\ M \end{bmatrix} \text{ and } C \triangleq [S\ N] \qquad (2.9)$$

and the result of theorem 2-1, namely:

$$\begin{bmatrix} RAC & R \\ C & 0 \end{bmatrix} = \left[\begin{array}{cc|c} I_r & 0 & T \\ 0 & 0 & M \\ \hline S & N & 0 \end{array}\right] \qquad (2.10)$$

Then the equation Ax = b has a solution if and only if $N^*b = 0$ and the general solution of Ax = b is given by

$$x_{\text{general solution}} = T^*S^*b + M^*z, \quad \forall z \qquad (2.47)$$

Proof. Recalling the matrix multiplication which lead to the form of Eq (2.10) in theorem 2-1, now, the transpose of this sequence of multiplications is taken prior to the actual multiplication. Thus from the original multiplication sequence

$$\begin{bmatrix} R & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} A & I \\ I & 0 \end{bmatrix}\begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} RA & R \\ I & 0 \end{bmatrix}\begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} RAC & R \\ C & 0 \end{bmatrix} \qquad (2.8)$$

and by taking the transpose of Eq (2.8) the following is obtained

$$\begin{bmatrix} C^* & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} A & I \\ I & 0 \end{bmatrix} \begin{bmatrix} R^* & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} C^*A & C^* \\ I & 0 \end{bmatrix} \begin{bmatrix} R^* & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} C^*AR^* & C^* \\ R^* & 0 \end{bmatrix} \tag{2.48}$$

Extending the results of theorem 2-1 to the form of Eq (2.48) provides the following equality

$$\begin{bmatrix} C^*AR^* & C^* \\ R^* & 0 \end{bmatrix} = \left[\begin{array}{c|c|c} I_r & 0 & S^* \\ \hline 0 & 0 & N^* \\ \hline T^* & M^* & 0 \end{array}\right] \tag{2.49}$$

where now

$$R^* = [T^* \; M^*] \tag{2.50}$$

$$C^* = \begin{bmatrix} S^* \\ N^* \end{bmatrix} \tag{2.51}$$

Since R and C are nonsingular matrices by hypothesis, the following series of implications can be made

$A^*x=b$ has a solution x

if and only if $C^*A^*x = C^*b$ has a solution x

if and only if $C^*A^*R^*y = C^*b$ has a solution y where $x = R^*y$

The last implication can be rewritten using Eq (2.49). Thus the last implication becomes

$$\begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} y = C^* b \tag{2.52}$$

where y is now defined as

$$y \triangleq \begin{bmatrix} W \\ Z \end{bmatrix}$$

with x consequently rewritten as

$$x = [T^* \; M^*] \begin{bmatrix} W \\ Z \end{bmatrix}$$

Now the last implication as rewritten in Eq (2.52) can be simplified as follows

$$\begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} y = \begin{bmatrix} S^* \\ N^* \end{bmatrix} b \tag{2.53}$$

$$\begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} W \\ Z \end{bmatrix} = [S^*b \; N^*b] \tag{2.54}$$

$$[W \quad 0] = [S^*b \quad N^*b] \tag{2.55}$$

For the simplification in Eq (2.53) through Eq (2.55) to hold, the following equalities must be true

$$W = S^*b \text{ and } N^*b = 0$$

where

$$x = T^*W + M^*z$$

Finally using the new definition of W provides the desired result

$$x_{\text{general solution}} = T^*S^*b + M^*z, \quad \forall z \qquad (2.56)$$

■

Corollary 2. The columns of M^* form a basis for the null space of A^*, referred to by $\eta(A^*)$.

Proof. The proof for corollary 2 is the similar to that of corollary 1, but follows from the representation of the general solution of x given in theorem 2-6.

■

From theorem 2-6 the last two remaining generalized inverses can be obtained. First, using the results established in theorem 2-6, the $A_{1,2,4}$ generalized inverse will be obtained in the following theorem. The final generalized inverse of interest will be obtained using the definition of an $A_{1,2,3,4}$ generalized inverse established earlier.

Theorem 2-7 (12). If $N^*S=0$ then the matrix ST is an $A_{1,2,4}$ generalized inverse of A.

Proof. To prove ST is an $A_{1,2,4}$ generalized inverse, it must be shown that ST satisfies the following equations

$$AA_{1,2,4}A = A \tag{2.57}$$

$$A_{1,2,4}AA_{1,2,4} = A_{1,2,4} \tag{2.58}$$

$$(A_{1,2,4}A)^* = (A_{1,2,4}A) \tag{2.59}$$

By the results of theorems 2-3 and 2-4 it follows that $A_{1,2,4}$ satisfies Eqs (2.57) and (2.58). To show Eq (2.59) holds it must be shown that $(A_{1,2,4}A)^*$ is symmetric. This proceeds in a manner similar to the proof of theorem 2-5. First let $A^-=ST$ and then show Eq (2.60) is symmetric:

$$(A^-A)^*(A^-A) = (A^-A) \tag{2.60}$$

Using the definition of symmetry mentioned in theorem 2-5, the transpose of the left hand side of Eq (2.60) is taken as follows

$$((A^-A)^*(A^-A))^* = (A^-A)^*((A^-A)^*)^* = (A^-A)^*(A^-A) \tag{2.61}$$

However, this symmetry it does not establish equality. To show equality let $N^*S=0$, and then by using the results of theorem 2-6, there must exist an X such that $A^*X=S$. Using this result and the previous definition of A^- leads to the following

$$(A^-A)^*(A^-A) = (A^*A^{-*})(A^-A)$$

$$= (A^{*}A^{-*})(A^{*}XTA)$$
$$= (A^{*}A^{-*}A^{*})(XTA)$$
$$= (A^{*})(XTA)$$
$$= (A^{*}XT)(A)$$
$$= (A^{-}A)$$

Therefore, the ST matrix also satisfies Eq (2.59) and is thus an $A_{1,2,4}$ generalized inverse.

■

The final inverse, the $A_{1,2,3,4}$ generalized inverse is obtained from theorems 5 and 7 using the defintions illustrated in figure 2 earlier. Therefore, if $MT^{*}=0$ and $N^{*}S=0$, the ST matrix must satisfy Eqs (2.1) through (2.4). Then by definition, the matrix ST is an $A_{1,2,3,4}$ generalized inverse.

III The ST Computation

Chapter 2 reviewed the theory behind the ST method for computing the generalized inverse of a matrix, and briefly introduced this technique through an example. This chapter will examine the computer implementation of this technique in greater detail and also discuss an interesting adaptation of the ST method which will make it more immediately useful and available.

In respect to the ST method, the algorithm behind this computational technique will be examined from several perspectives, including time and space complexities plus numerical accuracy considerations. To rounded out the discussion and put it in the proper perspective, the ST method will be compared to other techniques currently used to computed various generalized inverses. The intent of this comparison is to try to determine what possible advantages or potential disadvantages the ST method may possess in relation to these other techniques.

While determining the potential advantages and disadvantages, the discussion on comparisons will also serve to introduce the adaptaion of the ST method to a well known algorithm. This adaptation is based on a singular value decomposition algorithm. The ST method will be used to short cut a method used for computing an $A_{1,2,3,4}$ inverse based on the results of the singular value decomposition algorithm. This discussion will conclude the chapter with a detailed explanation and an example of this modified computational technique for finding an $A_{1,2,3,4}$ generalized inverse.

The ST Method: A Synthetic Approach

The ST method can be described as a five step technique for finding four generalized inverses of a given matrix. Of these five steps, two steps form the computational heart of the technique. These steps, step 2 and step 4, utilize a modified Gauss-Jordan and the modified Gram-Schmidt orthogonalization algorithms respectively. A brief overview of each step will be given. Following the overview, the computations surrounding step 2, the modified Gauss-Jordan algorithm, and step 4, the modified Gram-Schmidt orthogonalization algorithm, will be examined in detail. Based primarily on steps 2 and 4, the time and space complexity of the ST method will be discussed. Next some of the subtleties involved in implementing the ST method will be addressed from a very general view. Included in this discussion will be numerical accuracy considerations. Finally, this technique will be compared to others used to compute various generalized inverses.

Algorithm Overview. The example in chapter 1 used a new representation defined by theorem 1-1, namely Eq (2.10), as the starting point for computing four generalized inverses. To obtain this representation, step 1 begins by augmenting the matrix of concern,

$$\left[\begin{array}{c|c} \overset{m\times n}{A} & \overset{m\times m}{I} \\ \hline \overset{n\times n}{I} & 0 \end{array}\right]$$

Figure 7. Dimensions of Augment Matrix and Identities

say matrix A, with an identity matrix beside it and an identity matrix below it. If the given matrix A is of dimension mxn, then the identity matrix augmented beside A must be of dimension mxm. The identity matrix augmented below A must have dimensions nxn. Thus the result of step 1 is a square matrix with dimensions (m+n) by (n+m) as depicted in figure 7. This matrix representation was shown, by theorem 2-1, to be similar to the representation which will now be obtained in step 2. Before leaving step 1, note the zero matrix in the lower right-hand corner of figure 7. While it is not involved in the actual computation and is never changed, it has a potential use which will be pointed out later.

Now to obtained the next representation of the augmented A matrix, step 2 utilizes a slightly modified Guass-Jordan algorithm for reducing A to an identity matrix with dimensions equal to the rank of A. Applying this algorithm, the representation of figure 7 becomes the famiilar representation illustrated in figure 8. Important to note from figure 8 is how the submatrices S, T, M, and N arise. First,

$$\left[\begin{array}{c|c|c} I^{rxr} & 0 & T^{rxm} \\ \hline 0 & 0 & M^{(m-r)xm} \\ \hline S^{nxr} & N^{nx(n-r)} & 0 \end{array}\right]$$

Figure 8. Partitioning and Dimensions of Submatrices

these submatrices are the consequence of reducing of A to I_r where r is the rank of A. Second, the partitioning of what was originally the two identity matrices is determined by r. Thus the respective dimensions of these submatrices are S^{nxr}, T^{rxn}, $M^{(m-r)xn}$, and $N^{nx(n-r)}$ as illustrated in figure 8. As mentioned earlier, this representation forms the foundation for computing the four generalized inverses of interest in this work, and its validity is based on the theory reviewed in chapter 2.

In the representation of figure 8, the computation of the first generalized inverse of interest follows directly in step 3. Here the $A_{1,2}$ inverse is computed simply from the product of the S and T submatrices.

Computing the next two inverses, $A_{1,2,3}$ and $A_{1,2,4}$, require the use of the modified Gram-Schmidt orthogonalization algorithm. Again using the representation of figure 8, each row in T is orthogonalized to every row in M. Now taking the product of the original S submatrix and the new T submatrix gives the $A_{1,2,3}$ generalized inverse. This process is defined as step 4A. Step 4B again uses the same orthogonalization process but now it is applied to each column of S. Each column of the submatrix S is orthogonalized to every column of N. The product of the new S and old T submatrices produces the $A_{1,2,4}$ generalized inverse. Since both inverses are directly based on the results of step 3, the two separate computations were labeled as steps 4A and 4B.

The final inverse the $A_{1,2,3,4}$ generalized inverse, follows from both steps 4A and 4B. This step is naturally labeled step 5, and is composed simply of the product of the orthogonalized S and T submatrices.

The relationship of these five steps along with their important characteristics is summarized on the following page in figure 9.

Modified Guass-Jordan Algorithm. The Guass-Jordan algorithm is well documented (22:71; 27:417; 24; 1; 17), and the details of this method will not be discussed. Instead, the aspects of this method with particular importance to the ST method will be addressed. In particular, this includes the addition of a column operation to reduce a given matrix, and a change to what might normally be used as a pivoting scheme for reducing the given matrix. These two issues are discussed below as they pertain to the modifications of the Guass-Jordan algorithm adopted for the ST method.

STEP 1: AUGMENT A

Augment A with an identity matrix beside and below it to form:

$$\left[\begin{array}{c|c} A & I \\ \hline I & 0 \end{array}\right]$$

STEP 2: REDUCE A

Reduce A to an identity matrix with dimension equal to the rank of A to form:

$$\left[\begin{array}{cc|c} I_r & 0 & T \\ 0 & 0 & M \\ \hline S & N & 0 \end{array}\right]$$

STEP 3: FORM $A_{1,2}$ INVERSE

Form $A_{1,2}$ from the product of the submatrices S and T. $A_{1,2}$ satisfies the following conditions:

$$A A_{1,2} A = A$$
$$A_{1,2} A A_{1,2} = A_{1,2}$$

STEP 4A: FORM $A_{1,2,3}$ INVERSE

Form $A_{1,2,3}$ from the product of the submatrices S and T where each row of T is orthogonalized to the rows of the submatrix M. $A_{1,2,3}$ satisfies the following conditions:

$$A A_{1,2,3} A = A$$
$$A_{1,2,3} A A_{1,2,3} = A_{1,2,3}$$
$$(A A_{1,2,3})^* = (A A_{1,2,3})$$

STEP 4B: FORM $A_{1,2,4}$ INVERSE

Form $A_{1,2,4}$ from the product of the submatrices S and T where each column of S is orthogonalized to the columns of the submatrix N. $A_{1,2,4}$ satisfies the following conditions:

$$A A_{1,2,4} A = A$$
$$A_{1,2,4} A A_{1,2,4} = A_{1,2,4}$$
$$(A_{1,2,4} A)^* = (A_{1,2,4} A)$$

STEP 5: FORM $A_{1,2,3,4}$ INVERSE

Form $A_{1,2,3,4}$ from the product of S in step 4B and T in step 4A. $A_{1,2,3,4}$ satifies the following conditions:

$$A A_{1,2,3,4} A = A$$
$$A_{1,2,3,4} A A_{1,2,3,4} = A_{1,2,3,4}$$
$$(A A_{1,2,3,4})^* = (A A_{1,2,3,4})$$
$$(A_{1,2,3,4} A)^* = (A_{1,2,3,4} A)$$

Figure 9 Five Step Summary

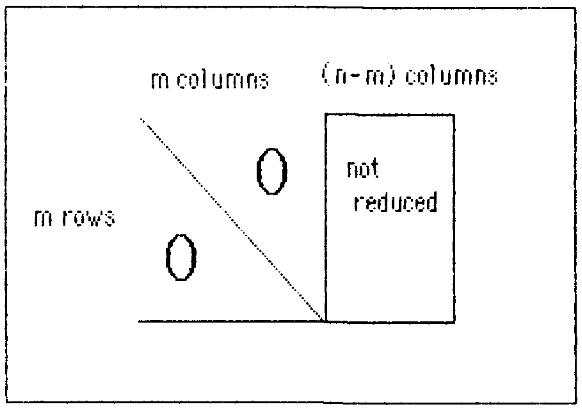


Figure 10. Underdetermined Case

Typically the Guass-Jordan algorithm is used to reduce a square matrix to row-echelon form. In this application, matrices are not square and row-echelon form is not always sufficient. Thus the Guass-Jordan method was modified to reduce a non square matrix to the form described in step 2 or likewise, derived in Eq (2.10). In some cases, the row-echelon form produced by a typical implementation of the Gauss-Jordan method would be sufficient for non square matrices, but given the case of a matrix with dimension mxn where n is greater than m, the desired form would not be reached. This underdetermined case is illustrated in figure 10. In figure 10 part of the original matrix was reduced to an mxm identity matrix, but this left the remaining (n-m) columns non-zero. For the ST method to work, the remaining columns in the underdetermined case must be reduced. This problem is eliminated by was using both row and column operations to reduce a given matrix. Row operations are used to first reduce the matrix to an upper triangular form with rank equal to that of the original matrix. Next applying column operations, the remaining elements above the diagonal are eliminated. This provides the desired form. While it would have been possible to flag the underdetermined cases for special treatment, this was not done. Instead, in the interest of generality and numerical accuracy considerations, which will be elaborated on later, a method using both row and column operations was adopted.

The addition of column operations to the existing Guass-Jordan algorithm required an accopanying change to the pivoting scheme normally employed with this algorithm. Typically, a satisfactory pivoting scheme for most cases would simply search each column below the diagonal for the best pivot element before attempting to eliminate the remaining elements (22:187-188; 24:38). This pivoting scheme works well when only row operations are used to reduce a matrix, but ignores the subsequent column operations which are required for the ST method. Since column operations possess the same potential problems found in row operations, namely subtracting numbers of similar size, any pivoting scheme used must also be able to look ahead for the subsequent column operations (27:593). One immediate technique would be adopting a complete pivoting scheme where the entire matrix is searched for the best pivot element. The obvious problem with this approach is the computational expense of searching the entire matrix. Since complete pivoting schemes normally only provide two to four times better results at a high cost (24:39) this method was not used. Instead a compromise was designed which uses at the most two column searches and a single row search. Thus, from a given diagonal element, a column search is conducted below the diagonal element followed by a row search to the right of the diagonal element. If the pivot element was chosen from the row search, a third search is conducted on the new column. This scheme is employed during the row reduction operations as the given matrix is reduced to an upper triangular form. The intent is to provide a good computing framework for the following column operations when the form of the matrix is less flexible. Here less flexible means it is difficult to pivot during column operations without loosing a zero element. The alternative of pivoting during column operations with loosing zeroes would begin to approach the computational burden of complete pivoting since the matrix would have to be rearranged back to the form of I_r at some point. Thus this scheme avoids the expense of complete pivoting but at the same time provides a good basis for the column operations.

Spreading the numerical operations between both the rows and columns has the added advantage of reducing the number of operations performed on the elements in each of the two augmented identity matrices. Instead of concentrating the changes on a single identity matrix like the unmodified Guass-Jordan method does, this reduction technique spreads the arithmetic operations across two identity matrices in an almost equal fashion. The result is less operations on individual elements of the submatrices S, T, M, and N. Since repeated simple arithmetic operations accumulate round-off error, and subsequent round-off errors are the result of the current error, the error in each submatrice should accumulate more slowly(27:593).

In summary, one change, column operations, was added to the Guass-Jordan algorithm to adapt it for the ST method. An accompanying pivoting scheme was developed to account for the column operations. The combined result is an effective routine which is, at the least, as numerically accurate as the well published version of the Guass-Jordan method.

Modified Gram-Schmidt Orthogonalization Algorithm. Steps 4A and 4B require, depending on the rank of the matrix of interest, specific rows of the T and M matrices and columns of the S and N matrices to be orthogonal. To orthogonalize the required rows or columns, the modified Gram-Schimdt orthogonalization process was used. This is a modified version of the original Gram-Schmidt algorithm and differs only in the order calculations are performed (24:151).

The use of the modified Gram-Schmidt, in the ST method, differs a little from its typical description since the orthogonalization process depends on the rank of the matrix of interest. Thus algorithm is first described below in its most general representation. Its application to the ST method is then explained and demonstrated with an example.

The modified Gram-Schmidt orthogonalization algorithm is described with the following short segment of a pseudocode. In this desciption, the purpose is to orthogonalize each vector denoted with the small letter a and subscripts. The algorithm is as follows:

```
Given the vectors $a_i$ for i=1,2,3,...,M

For k=1 to M do
   For j=k+1 to M do
      $a_j = a_j - [(a_j \bullet a_k)/(a_k \bullet a_k)]a_k$
   End of j loop
End of k loop
```

The algorithm description also serves to point out two useful features of this algorithm; its use of storage and its simple and direct nature. In respect to storage, the modified vectors replace themselves so no temporary storage is needed for intermediate results which reduces its storage over the original Gram-Schmidt process. Second, the process is completely described in a few short lines which attests to its simple and direct nature.

Since the ST method requires that each individual row of the T submatrix be orthogonal to the M submatrix, the change to the basic algorithm described above involves only the loop indices. For the ST method and the T and M submatrices, the outer loop is limited by the rank of the original matrix. The inner loop is controlled by the number of rows in the augment matrix A. In terms of the algorithm, this restricts the actual number of arithmetic operations performed over the case where a set of vectors or rows for matrices are fully orthogonalized. For the case of the T and M submatrices, each row in T is fully orthogonalized to the rows of M, but the rows of T are not orthogonalized to each other. The same applies to the orthogonalization of each column of S to the entire N submatrix. The following example demonstrates the orthogonalization process used for the ST method.

<u>Example 2</u>. Given the following reduced matrix with dimension 6x4 and a rank of 2:

$$\left[\begin{array}{c|c|c} I_r & 0 & T \\ \hline 0 & 0 & M \\ \hline S & N & 0 \end{array}\right] = \left[\begin{array}{cc|cc|cccccc} 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.33 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 3.00 & 0.00 & -2.00 & 0.00 & 0.00 & 0.00 \\ \hline 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & -1.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & -1.00 & 1.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\ \hline 0.00 & 0.00 & 1.00 & 0.00 & & & & & & \\ 0.00 & 0.00 & 0.00 & 1.00 & & & & & & \\ 0.00 & 1.00 & 3.00 & -2.00 & & & & & & \\ 1.00 & -.33 & -1.00 & 1.00 & & & & & & \end{array}\right]$$

Orthogonalize each individual row in T to all the rows of M. Here the orthogonalization process begins with the bottom row in M so the identity matrix, I_r will be preserved. If the process were started with the first row in T and worked down, the I_r form obtained in the previous step would be lost. This is because operations are not performed on only the rows of T or M, but on the entire matrix. Therefore care must be taken to preserve the work done to this point. Thus with the outer loop index set to 6 (the fourth row in M), and the inner loop is initialized to 5, the algorithm iterates down to 1. This results in the following changes to the T and M submatrices

$$\left[\frac{T}{M}\right] = \left[\begin{array}{rrrrrr} 0.00 & 0.00 & 0.33 & 0.00 & 0.00 & 0.00 \\ 1.50 & 0.00 & -2.00 & 0.00 & 0.00 & -1.50 \\ \hline 0.50 & 0.00 & -1.00 & 0.00 & 1.00 & -0.50 \\ 0.00 & 0.00 & 1.00 & 1.00 & 0.00 & 0.00 \\ -0.50 & 1.00 & 1.00 & 0.00 & 0.00 & 0.50 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{array}\right]$$

After the first iteration of the outer loop, the last row in M is orthogonal to each row above it including the two rows in T. The next pass of the outer loop will begin at the third row of M and orthogonalizes it to each row above it. The resulting matrix is:

$$\left[\frac{T}{M}\right] = \left[\begin{array}{rrrrrr} 0.06 & -0.13 & 0.20 & 0.00 & 0.00 & 0.06 \\ 0.80 & 1.40 & -0.60 & 0.00 & 0.00 & -0.80 \\ \hline 0.20 & 0.60 & -0.40 & 0.00 & 1.00 & -0.20 \\ 0.20 & -0.40 & 0.60 & 1.00 & 0.00 & -0.20 \\ -0.50 & 1.00 & 1.00 & 0.00 & 0.00 & 0.50 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{array}\right]$$

The process is repeated again for the second row of M orthogonalizing it to each row above it as the inner loop iterates from 3 to 1. The resulting matrix after this iteration is

$$\left[\frac{T}{M}\right] = \left[\begin{array}{cccccc} 0.04 & -0.08 & 0.12 & -0.12 & 0.00 & 0.04 \\ 0.87 & 1.25 & -0.37 & 0.37 & 0.00 & -0.87 \\ \hline 0.25 & 0.50 & -0.25 & 0.25 & 1.00 & -0.25 \\ 0.20 & -0.40 & 0.60 & 1.00 & 0.00 & -0.20 \\ -0.50 & 1.00 & 1.00 & 0.00 & 0.00 & 0.50 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{array}\right]$$

Finally, the first row in M is orthogonalized to each row above it, namely the two rows in T. The orthogonalization process is stopped at this point by the rank of the original matrix. The final T and M submatrices are

$$\left[\frac{T}{M}\right] = \left[\begin{array}{cccccc} 0.05 & -0.05 & 0.11 & -0.11 & 0.05 & 0.05 \\ 0.67 & 0.83 & -0.16 & 0.16 & 0.83 & -0.67 \\ \hline 0.25 & 0.50 & -0.25 & 0.25 & 1.00 & -0.25 \\ 0.20 & -0.40 & 0.60 & 1.00 & 0.00 & -0.20 \\ -0.50 & 1.00 & 1.00 & 0.00 & 0.00 & 0.50 \\ 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{array}\right]$$

The same process is performed for the submatrices S and N but on a column basis. This time the algorithm only completes two outer iterations to meet the orthogonalization requirement. Again the orthogonalization process is stopped by the rank of the original matrix. The final matrix with the results of both orthogonalization processes and the unchanged identity matrix is shown below. Step 5 follows directly from this final form as the product of the orthogonalized S and T submatrices

$$\left[\begin{array}{c|c|c} I_r & 0 & T \\ \hline 0 & 0 & M \\ \hline S & N & 0 \end{array}\right] = \left[\begin{array}{cc|cc|cccccc} 1.00 & 0.00 & 0.00 & 0.00 & 0.05 & -0.05 & 0.11 & -0.11 & 0.05 & 0.05 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.67 & 0.83 & -0.16 & 0.16 & 0.83 & -0.67 \\ \hline 0.00 & 0.00 & 0.00 & 0.00 & 0.25 & 0.50 & -0.25 & 0.25 & 1.00 & -0.25 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.20 & -0.40 & 0.60 & 1.00 & 0.00 & -0.20 \\ 0.00 & 0.00 & 0.00 & 0.00 & -0.50 & 1.00 & 1.00 & 0.00 & 0.00 & 0.50 \\ 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\ \hline -0.05 & -0.21 & 1.00 & 0.00 & & & & & & \\ -0.23 & 0.13 & 1.16 & 1.00 & & & & & & \\ 0.29 & 0.07 & 0.67 & -2.00 & & & & & & \\ 0.83 & 0.01 & 0.17 & 1.00 & & & & & & \end{array}\right]$$

One remaining aspect of this technique not yet discussed is it's numerical accuracy. As described above, the modified Gram-Schmidt process is a numerically stable technique (24:152). This is based on completely orthogonalizing every vector or row in this case to every other vector or row. In the ST method, the orthogonalization process is not complete, but instead limited by rank. In terms of the arithmetic operations performed, this means the modified Gram-Schmidt, as adapted for the ST methods, performs less arithmetic operations than the general case which is described above as numerically stable. Thus the accuracy of the Gram-Schmidt orthogonalization possess in this application should be, at the least, as good as the above reference suggests.

Time and Space Complexities. Two measures of any algorithm are its speed and its storage requirements, and these measures are often referred to as an algorithm's time and space complexities respectively (25:24-39). For the ST method, the speed or time complexity analysis will be limited to the two dominating processes, the modified Guass-Jordan algorithm and the modified Gram-Schmidt orthogonalization algorithm. For storage requirements or the space complexity, only the augmented matrix will be considered. These limitations are imposed to make the analysis more general and consequently more accurately compared to other existing algorithms.

The time complexity of the ST method is based on counting numerical operations performed in the two dominating processes. This is even further limited by using only floating-point operations as is commonly done (17:78).

Thus the basis of the modified Guass-Jordan reduction becomes two, three level loops. Summarized by the row operations performed on a matrix with dimensions mxn, the loops consist of

```
For k=1 to M
  For j=k+1 to M
    For i=k+1 to (M+N)
```

For column operations the loops are identical with the exception of the outer loop limits. The limit for the outer two loops is N. Using a common approximation for the above, the total row and column loops are equated to $m^3/3$ or $n^3/3$ floating-point operations respectively (17:80). The total operation count for the matrix reduction is then simply the sum of each which is $(m^3+n^3)/3$.

For the modified Gram-Schimdt orthogonalization process, the time analysis is the same. Each orthogonalization process contains a three-deep nested loop as illustrated above for the row reduction operations. In the worst case, with a matrix rank of one, the operation count for the row orthogonalization is $n^3/3$, and for the column orthogonalization the count is $m^3/3$. Again the combined count is simply the sum of each which is $(m^3+n^3)/3$.

Combining the operation count for the Guass-Jordan and Gram-Schmidt provides an overall operation count of $(2/3)(m^3+n^3)$. This applies to the computation of an $A_{1,2,3,4}$ generalized inverse. Computing a lower inverse would require less operations down to a minimum of $(m^3+n^3)/3$ for an $A_{1,2}$ generalized inverse. No other operations performed by the ST method would appreciably change the cubic derived above. While including these other operations might change the coefficients or add a lower term to the overall operation count, the cubic nature of the two dominating processes will still prevail as the methods chief computational burden and also serve as a good comparative reference.

The space complexity is based on the number of memory locations occupied by the augment matrix. While the actual implementation used considerably more (extra storage was used for validation and demonstration purposes), the heart of the algorithm requires only enough storage for the original matrix, two identity matrices, and the resulting generalized inverse. The analysis flows very directly from this perspective. For the original matrix, a block of memory determined by the dimensions of the matrix is needed. If given a matrix with dimensions m by n, then m times n memory locations are needed. Since the original matrix is augmented by two square identity matrices, an additional m^2+n^2 locations are needed. The resulting generalized inverse needs n times m locations since it has dimensions n by m based on the original matrix. Combining the space requirement for the original matrix, the two identity matrices, and the generalized inverses results in $(m+n)^2$ locations.

Other Algorithm and Computation Considerations. Lumped under this section are the subtleties and hidden insights enclosed in the computer implementation of the ST method which might not be obvious from the desciption above or its documentation. Each is addressed individually and with no specific order in the paragraphs which follow. These should be helpful to any future implementations of this method for computing the generalized inverse of a matrix.

The simplest data structure for implementing the augmented matrix is a square two dimension array. For a matrix with dimensions mxn, such an array would be dimensioned (m+n) by (m+n). Although this representation is the simplest, it also poses one problem regarding storage. In the space analysis the complexity was determined to be $(m+n)^2$. Using this scheme the original matrix and the two identities matrices will require $(m+n)^2$ locations alone. To keep the storage to the minimum found in the space analysis, the computed generalized inverse could be stored in the unused lower right corner of the (m+n) by (m+n) array. This was eluded to earlier (on page 44) when discussing the composition of the augmented matrix after it was reduced. While this may require a little extra effort during the final phase of the program to place the product of S and T back into the lower right corner of the large array, the trouble may be offset by the ease of manipulating only one data structure and still using minimum storage. A second alternative would be to store the computed generalized inverse on top of the identity matrix which was denoted as I_r. There are also problems with this approach, namely taking the transpose of the inverse while storing and retrieving it from this area, but again the convenience may out weight the trouble of manipulating only one data structure.

The use of a conditional statement with a floating-point representation introduces a problem when the condition is testing for equality. In the ST method, this problem arose during the matrix reduction process of step 2 when testing for zeroes. Since a given matrix element may never exactly equal a floating-point zero, due to small numerical errors, the question must be asked how should the a condition be setup to test for a floating-point zero? One method represents a zero relative to the scale of the machine used, and then bases the condition on the absolute value of this relative zero (24:43). Taking the machine used for the original development which had 16 decimal digits of accuracy, a zero was represented as $5x10^{-16}$. Thus the absolute value of any matrix element less than $5x10^{-16}$ was considered zero. Use of a different machine would require an adjustment to this representation of zero. Additionally, different size matrices would require adjustments to the amount of error allowed in a zero.

There are certainly few if any applications where all four generalized inverses, discussed in this work, are needed together. Thus only certain parts of the ST method, as implemented in appendix B, are needed for a given application. This reduces the complexity of a potential program significantly from what is included in appendix B. As mentioned above under the time complexity analysis, this results in a range of operation counts from $(1/3)(m^3+n^3)$ to $(2/3)m^3+n^3)$ floating point operations. The same applies to the other routines included in appendix B supporting the computation of a given generalized inverse. These include the multiplication, printing, and input routines. In any given application, these routines could be reduced significantly to support the given application's requirements lessening the computational burden of the ST method.

Another aspect of applying the ST method concerns the environment where the application would actually reside. Consider an environment with a large mathematical software package. Such a package could be used to piece together an effective ST implementation with minimal modifications. For example, by calling a Gauss-Jordan algorithm followed by any necessary column reductions and a call to a matrix multiplication routine, an $A_{1,2}$ inverse could be computed with little effort. There are certainly many other possibilities which are made possible due to the modular nature of the ST method.

One final topic concerns algorithm validation. Two methods were used to validate this effort. The most useful was testing the computed inverse against the one to four conditions it is suppose to satisfy. These conditions, Eqs (2-1) through (2-4), provide a reliable means of testing the algorithm. The second method was simply to compare the computed results to a known correct result. Appendix A provides a detailed example showing the use of both of these methods. The conditions are tested as each inverse is computed, and the final unique $A_{1,2,3,4}$ generalized inverse matches the known solution found in the referenced text. Of the two methods used, the most useful and informative was using the four conditions. This method also provides insight into possible numerical inaccuracies since the results can be compared to the correct solution. This is the same as taking the classical inverse and then multiplying it and the original inverse together. The resulting identity gives an indication of how accurate the computed inverse is just as using the four Penrose conditions does for a generalized inverse.

Algorithm Comparison. When comparing two or more algorithms which perform the same basic function, but do so by different methods, it can be difficult to find a common basis for comparison. This especially true in this case where there are many methods for computing a generalized inverse. Here, the comparison process is particularly complicated by the wide variety of notation used to describe such computational techniques. Additionally each description is normally only developed to provide an overview of the method versus the detail needed for an indepth comparison. To overcome these difficulties, six general categories were chosen to compare algorithms. Each category is described below.

The first category deals with the initial representation of the matrix. This category was chosen since the process of augmenting the initial matrix is central to the ST method. Thus any comparison involving the ST method should include this particular aspect. From a survey of several different methods, the second category was chosen as factorization methods. This category was chosen since, in one way or another, all algorithms examined performed some factorization of

Method	1	2	3
Characteristic	ST	$(A^t A)^{-1} A$ (21:142)	$Q_2 \Sigma^+ Q_1^t$ (27:135)
Intial Matrix Representation	$\left[\begin{array}{c\|c} A & I \\ \hline I & 0 \end{array}\right]$	$[A^t A \mid I]$	$[A]$
Factorization	ROW AND COLUMN OPS.	ROW OPERATIONS	SINGULAR VALUE DECOMPOSITION
Logical Flow	DIRECT	DIRECT	MULTIPLE DECISIONS
Time Complexity	$2/3(m^3+n^3)$	$4/3(n)^3$	$> 2/3(m^3+n^3)$
Space Complexity	$(m+n)^2$	$2n^2$	$(m+n)^2$
Flexibility	Computes 4 inverse	Restricted by rank	Computes 1 inverse

Figure 11. Algorithm Comparison

the initial matrix in the process of computing a generalized inverse. The third category deals with an algorithm's general structure. Here general structure means the logical flow of the algorithm. In other words, does the algorithm require extensive logic to piece together the individual parts of the computation or does it flow directly from start to finish. The fourth and fifth categories are respectively the time and space complexities of an algorithm. The final category deals with the flexibility of the algorithm. In this regard flexibilities concern what range of problems can the method handle or what constraints are levied on the problem's input.

Using these six categories, the ST method is compared to two representative methods for computing a generalized inverse. The comparison is summarized in figure 11 (Each method is defined and referenced to where it was obtained in the figure). The figure shows a number of general results. First, the ST method has the most complex initial representation, but this representation appears to lead to an advantage in the other characteristics especially in flexibility. The ST method is the only one of the three which can compute four generalized inverses without any restrictions. Method 2 seems to be a best in terms of storage, computations, and factorization, but is restricted by the rank and dimensions of a given matrix. Method 3 begins with the simplest initial representation but quickly becomes very complex. This algorithm will be examined in greater detail in the discussion of the adaptation for the ST method.

In summary, it hard to say one method is absolutely better than the other, but certainly the ST method is by far the most flexible. It can be used in more situations than the other two and does not suffer significantly in terms of the other categories of comparison due to this advantage in flexibility. Ignoring all other possibilities and using the six categories above, the ST method would probably offer the best choice for computing a given generalized inverse.

The ST Method Adapted to Singular Value Decomposition.

The ST method will be adapted to the results of a singular value decomposition algorithm to obtain an $A_{1,2,3,4}$ generalized inverse. This will begin by first reviewing an algorithm for singular value decomposition to provide the proper perspective for the ST adaption. Next the representation developed for the ST method will be derived from the results of the singular value decomposition algorithm to compute the $A_{1,2,3,4}$ generalized inverse. Finally this approach will be compared to another method which uses the results of a singular value decomposition algorithm to directly compute an $A_{1,2,3,4}$ generalized inverse.

The Singular Value Decomposition Algorithm. Given a matrix A^{mxn}, singular value decomposition factors the matrix into the product of two orthonormal matrices and a diagonal matrix. This product is defined as

$$A = Q_1 \Sigma Q_2^t$$

To understand how the ST method is adapted to the results of the singular value decomposition, the Q_1, Q_2, and Σ matrices, as shown above, will be developed with a simple example. Then the S and T submatrices will be derived from these three matrices. The process begins by first choosing the matrix A as

$$A = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Next, find the Q_2 matrix. This is begun by first forming A^tA and then computing $|A^tA-\lambda I|$ with the goal of finding the characteristic root of A^tA as follows

$$A^tA = [3 \; 4]\begin{bmatrix} 3 \\ 4 \end{bmatrix} = 25$$

$$|A^tA-\lambda I| = |25-\lambda| = 0$$

$$\lambda = 25$$

With a characteristic root of $\lambda = 25$, the process continues by finding the associated characteristic vector

$$A^tAx = 25x$$

$$x = [1]$$

Using the characteristic vector, the Q_2 matrix is defined as follows

$$Q_2 = [1]$$

The next step is to form Q_1. This begins by forming AA^t and then computing $| AA^t - \lambda I |$ to finds its characteristic roots similar to what was done to find Q_2. In this case the process produces the following

$$AA^t = \begin{bmatrix} 3 \\ 4 \end{bmatrix} [3\ 4] = \begin{bmatrix} 9 & 12 \\ 12 & 16 \end{bmatrix}$$

$$|AA^t - \lambda I| = \begin{vmatrix} 9-\lambda & 12 \\ 12 & 16-\lambda \end{vmatrix} = 0$$

$$\begin{vmatrix} 9-\lambda & 12 \\ 12 & 16-\lambda \end{vmatrix} = (9-\lambda)(16-\lambda)-144$$

$$(9-\lambda)(16-\lambda) = 144-25\lambda-\lambda^2-144 = 0$$

$$\lambda^2-25\lambda = 0$$

$$\lambda = 0 \ ; \ \lambda = 25$$

Notice at this point that $\lambda = 25$ is a common solution for both $| AA^t - \lambda I | = 0$ and $| A^tA - \lambda I | = 0$. The next step uses this common solution to obtain the characteristic vector for AA^t as shown below

$$[AA^t]\, x = \begin{bmatrix} 9 & 12 \\ 12 & 16 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 25 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

which is solved for x as the following pair of equations

$$9x_1 + 12x_2 = 25x_1$$

$$12x_1 + 16x_2 = 25x_2$$

which becomes

$$-16x_1 + 12x_2 = 0$$
$$12x_1 - 9x_2 = 0$$

Reduced, these equations simply become $x_1 = (3/4)x_2$. Then by choosing x_2 as 4, x_1 is equal to 3 and the characteristic vector becomes

$$x = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Normalizing x provides the following

$$x = \begin{bmatrix} \frac{3}{5} \\ \frac{4}{5} \end{bmatrix}$$

Now Ql is constructed using the normalized x. The first column of Ql is x leaving one additional column to be filled to make the product Ql and Σ conform. This second column is filled with a vector making Ql an orthogonal matrix. In this case, Ql is defined below showing this choice

$$Q_1 \triangleq \begin{bmatrix} \frac{3}{5} & \frac{4}{5} \\ \frac{4}{5} & -\frac{3}{5} \end{bmatrix}$$

The last matrix to be found by singular value decomposition is Σ. This matrix is the product of Ql, A and Q2 as shown below

$$Q_1 A Q_2 = \begin{bmatrix} \frac{3}{5} & \frac{4}{5} \\ \frac{4}{5} & -\frac{3}{5} \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix} [1] = \begin{bmatrix} 5 \\ 0 \end{bmatrix} [1] = \Sigma$$

The diagonal elements of Σ display the singular values for the original matrix A. In this case, the decomposition process provides only one singular value of A which is simply 5. This will become important as the S and T matrices are obtained.

<u>Deriving the S and T Submatrices</u>. With the Ql, Σ, and Q2 matrices, the S and T matrices of the ST method can now be obtained in a very direct manner. First a matrix is formed using the Ql, Σ and Q2 matrices and partitioned as shown below.

$$\left[\begin{array}{c|c} \Sigma & Q_1 \\ \hline Q_2 & 0 \end{array}\right] = \left[\begin{array}{c|cc} 5 & \frac{3}{5} & \frac{4}{5} \\ 0 & \frac{4}{5} & -\frac{3}{5} \\ \hline 1 & & \end{array}\right]$$

This representation should look very similar to the representation developed in the ST method. In fact, by only dividing each diagonal element in the Σ matrix using row operations, the ST method is obtained. This is equivalent to saying divide each row of Q1 by the corresponding singular values. For this example, there is only one diagonal element or singular value in Σ so the resulting matrix in the ST representation becomes

$$\left[\begin{array}{c|c|c} I_r & 0 & T \\ \hline 0 & 0 & M \\ \hline S & N & 0 \end{array}\right] = \left[\begin{array}{c|cc} 1 & \frac{3}{25} & \frac{4}{25} \\ \hline 0 & \frac{4}{5} & -\frac{3}{5} \\ \hline 1 & & \end{array}\right]$$

As described many times before, the $A_{1,2,3,4}$ generalized inverse simply becomes the product of S and T. For this example, $A_{1,2,3,4}$ is

$$A_{1,2,3,4} = [\,1\,]\begin{bmatrix} \frac{3}{25} & \frac{4}{25} \end{bmatrix} = \begin{bmatrix} \frac{3}{25} & \frac{4}{25} \end{bmatrix}$$

This adaptation of the ST method is certainly simpler than the more publisized technique defined by

$$A_{1,2,3,4} = Q_2 \Sigma^{+} Q_1^{t}$$

where

IV Application to the Lyapunov Matrix Equation

To avoid the pitfalls associated with the computation of a generalized inverse, potential applications were modified to circumvent matrices not suitable to the classical method of matrix inversion. Further, some potential applications probably never even considered the use of a generalized inverse, especially the less known generalized inverses like the $A_{1,2,3}$. One such application, which probably falls somewhere between these two possibilities, will be adapted to use an $A_{1,2}$ generalized inverse. This application, the Lyapunov matrix equation, will be solved using an $A_{1,2}$ generalized inverse. The solution will be developed in detail in order to show how the $A_{1,2}$ generalized inverse fits this application.

The Lyapunov Matrix Equation

The necessary and sufficient conditions needed to show the Lyapunov equation, Eq (4.1), has a solution will be established. This

$$AX-XB+C = 0 \tag{4.1}$$

where

$A \in \mathbb{C}^{mxm}$

$B \epsilon \c{C}^{nxn}$

$C \epsilon \c{C}^{mxn}$

solution will be defined in terms of an $A_{1,2}$ generalized inverse. Two steps will be required to accomplish this goal. First, the solution of the Lyapunov matrix equation must be defined in terms of two pairs of equations. Each pair of equations will trap the solution of the Lyapunov matrix equation, if a solution exists. Establishing each pair of equations will require the use of several identities which will be developed in the bulk of this section. Second, once the two pair of equations are established, their solution, and consequently the solution of the Lyapunov equation, will be expressed using an $A_{1,2}$ generalized inverse. Thus a general common solution of the pairs of equations will be defined to express the solution of the Lyapunov matrix equation.

Theorem 4-1. Let Eq (4-1) have a solution X, then the following matrices are simialr

$$\begin{bmatrix} A & C \\ 0 & B \end{bmatrix} \text{ and } \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} \qquad (4.2)$$

Proof. Let X be a solution of Eq (4.1), then the following holds

$$\begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} \begin{bmatrix} A & C \\ 0 & B \end{bmatrix} \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} = \begin{bmatrix} A & C-XB \\ 0 & B \end{bmatrix} \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} \qquad (4.3)$$

$$\begin{bmatrix} A & C-XB \\ 0 & B \end{bmatrix} \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} = \begin{bmatrix} A & AX+C-XB \\ 0 & B \end{bmatrix} \tag{4.4}$$

$$\begin{bmatrix} A & AX+C-XB \\ 0 & B \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} \tag{4.5}$$

■

Using the given, namely that Eq (4.1) has a solution, the two matrices of Eq (4.2) are similar. Note this holds for any X since the matrices which appear in the left hand side of Eq (4.3) as shown below in Eq (4.6) are inverses

$$\begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} \text{ and } \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} \tag{4.6}$$

With theorem 4-1, it has been shown that there exists an X such that the matrices of Eq (4.2) are similar and consequently Eq (4.1) has a solution. The next step is to extend this result. To do this, define the matrix R as follows

$$R \triangleq \begin{bmatrix} A & C \\ 0 & B \end{bmatrix} \tag{4.7}$$

then

$$[R-\lambda I] = \begin{bmatrix} A-\lambda I & C \\ 0 & B-\lambda I \end{bmatrix}$$

Next find the determinant of $[R-\lambda I]$ as follows

$$|R-\lambda I| = \begin{vmatrix} A-\lambda I & C \\ 0 & B-\lambda I \end{vmatrix} \tag{4.8}$$

$$\begin{vmatrix} A-\lambda I & C \\ 0 & B-\lambda I \end{vmatrix} = |A-\lambda I|\,|B-\lambda I| \tag{4.9}$$

$$|A-\lambda I|\,|B-\lambda I| = f_A(\lambda) \circ g_B(\lambda) = f_R(\lambda) \tag{4.10}$$

This shows that $|R-\lambda I|$ factors into a pair of polynomials, namely the characteristic polynomials of the matrices A and B. The degree of $f_A(\lambda)$ is less than or equal to m and the degree of $g_B(\lambda)$ is less than or equal to n. Also, $f_R(R)$ is equal to zero since every matrix satisfies its own characteristic equation.

Now using the results of theorem 4-1, namely

$$\begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} \begin{bmatrix} A & C \\ 0 & B \end{bmatrix} \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} \tag{4.11}$$

let $f_A(\lambda)$ be defined as follows for λ replaced by R

$$f_A(R) = \begin{bmatrix} U & M \\ V & N \end{bmatrix} \tag{4.12}$$

where U, V, M, and N are polynomials in the matrices A, B, and C. Any polynomial, $f_A(R)$, commutes with R where the coefficients are complex or real, this results in the following

$$Rf_A(R) = \begin{bmatrix} A & C \\ 0 & B \end{bmatrix} \begin{bmatrix} U & M \\ V & N \end{bmatrix} = \begin{bmatrix} AU+CV & AM+CN \\ BV & BN \end{bmatrix} \tag{4.13}$$

$$\begin{bmatrix} AU+CV & AM+CN \\ BV & BN \end{bmatrix} = \begin{bmatrix} U & M \\ V & N \end{bmatrix} \begin{bmatrix} A & C \\ 0 & B \end{bmatrix} \tag{4.14}$$

$$\begin{bmatrix} U & M \\ V & N \end{bmatrix} \begin{bmatrix} A & C \\ 0 & B \end{bmatrix} = \begin{bmatrix} UA & UC+MB \\ VA & VC+NB \end{bmatrix} = f_A(R)R \tag{4.15}$$

This implies the following identities

$$AU+CV = UA \tag{4.16}$$

$$BV = VA \tag{4.17}$$

$$AM+CN = UC+MB \tag{4.18}$$

$$BN = VC+NB \tag{4.19}$$

Again recalling the results of theorem 4-1 in the form of Eq (4.11)

$$\begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} \begin{bmatrix} A & C \\ 0 & B \end{bmatrix} \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} \tag{4.11}$$

This equation holds for any solution X of Eq (4.1)

$$AX-XB+C = 0 \tag{4.1}$$

Thus for any polynomial $f_A(\lambda)$ given by Eq (4.12)

$$f_A(R) = \begin{bmatrix} U & M \\ V & N \end{bmatrix} \tag{4.12}$$

and under the similarity transformation as defined and proved earlier for Eq (4.2), it follows

$$\begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} f_A(R) \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} = \begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} \begin{bmatrix} U & M \\ V & N \end{bmatrix} \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} \tag{4.20}$$

$$= \begin{bmatrix} U\text{-}XV & M\text{-}XN \\ V & N \end{bmatrix} \begin{bmatrix} I & X \\ 0 & I \end{bmatrix}$$

$$= \begin{bmatrix} U\text{-}XV & (U\text{-}XV)X+M\text{-}XN \\ V & VX+N \end{bmatrix}$$

$$\begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} f_A(R) \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} = \begin{bmatrix} f_A(A) & 0 \\ * & f_A(B) \end{bmatrix}$$

where $f_A(B)$ is not equal to zero and $*$ is not of concern. Next by matching the terms of Eq (4.20), namely the respective terms of the equality

$$\begin{bmatrix} U\text{-}XV & (U\text{-}XV)X+M\text{-}XN \\ V & VX+N \end{bmatrix} = \begin{bmatrix} f_A(A) & 0 \\ * & f_A(B) \end{bmatrix} \tag{4.21}$$

The follwing pair of linear equations are obtained

$$U\text{-}XV = 0 \tag{4.22}$$

$$(U-XV)X+M-XN = M-XN = 0 \tag{4.23}$$

As it will be proved below, any solution of Eq (4.1) must satisfy the above pair of equations (4.22) and (4.23). In other words, the pair of all solutions for equations (4.22) and (4.23) trap the solutions of the Lyapunov matrix equation, Eq (4.1).

Theorem 4-2. If N^{-1} exist, and X is a common solution of equations (4.22) and (4.23), then X is also a solution to Eq (4.1).

Proof. Let X be a common solution to Eqs (4.22) and (4.23), then using the identities defined in Eqs (4.16) through (4.19), derive the Lyapunov matrix equation as follows

$$\begin{aligned}
0 &= (XN-M)B \\
&= XNB-MB \\
&= XMB+UC-AM-CN \\
&= X(BN-VC)-AM-CN+UC \\
&= X(BN-VC)-AXN-CN+UC \\
&= XBN-AXN-CN+(UC-XVC) \\
&= (XBN-AXN-CN)+(U-XV)C \\
&= (XB-AX-C)N
\end{aligned}$$

■

Theorem 4-3. Let V-1 exist and let X be a common solution of Eqs (4.22) and (4.23), then X is also a solution of Eq (4.1).

Proof. Let X be a common solution to Eqs (4.22) and (4.23), then using the identities defined in Eqs (4.16) through (4.19), derive the Lyapunov matrix equation as follows

$$\begin{aligned}
0 &= (U-XV)A \\
&= UA-XVA \\
&= XVA-(UA) \\
&= XVA-(AU+CV) \\
&= X(VA)-AU-CV \\
&= XBV-AU-(CV) \\
&= XBV-AU-(UA-AU) \\
&= XBV-UA \\
&= XBV-AU-CV \\
&= -AU+XBV-CV \\
&= -AXV+XBV-CV \\
&= (-AX+BX-C)V
\end{aligned}$$

■

In addition to the two Eqs (4.22) and (4.23), a second pair of equations can be used to trap the solution to the Lyapunov matrix equation. The common solution of at least one pair will define the solution to the Lyapunov matrix equation. The second pair of equations are

$$VX+N = 0 \tag{4.24}$$

$$UX+M = 0 \tag{4.25}$$

These equations are derived in a manner similar to Eqs (4.23) and (4.24). First, the equations arise from the results of theorem 4-1, but with two terms rearranged. Now the similarity transformation defined for the matrices of Eq (4.2)

$$\begin{bmatrix} A & C \\ 0 & B \end{bmatrix} \text{ and } \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} \qquad (4.2)$$

becomes equivalently

$$\begin{bmatrix} I & X \\ 0 & I \end{bmatrix} \begin{bmatrix} A & C \\ 0 & B \end{bmatrix} \begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} \qquad (4.26)$$

Now this similarity transformation is used in place of Eq (4.20) with $f_B(R)$ substituted for $f_A(R)$. Now similar to Eq (4.20) it follows

$$\begin{bmatrix} I & X \\ 0 & I \end{bmatrix} f_B(R) \begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} = \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} \begin{bmatrix} U & M \\ V & N \end{bmatrix} \begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} \qquad (4.27)$$

$$= \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} \begin{bmatrix} U & M-UX \\ V & N-VX \end{bmatrix}$$

$$= \begin{bmatrix} U+XV & (M-UX)+X(N-VX) \\ V & N-VX \end{bmatrix}$$

$$\begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} f_B(R) \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} = \begin{bmatrix} f_B(A) & 0 \\ * & f_B(B) \end{bmatrix}$$

Here $f_B(B)$ is equal to zero which provides the final key needed to obtain the Eqs (4.24) and (4.25). For a more complete derivation of

Eqs (4.24) and (4.25) see (6:13-14). Like theorems 4-2 and 4-3, Eqs (4.24) and (4.25) tie once again back to the Lyapunov matrix equation. Theorems 4-4 and 4-5 show this with the given conditions.

Theorem 4-4. If V^{-1} exist, and X is a common solution of equations (4.24) and (4.25), then X is also a solution to Eq (4.1).

Proof. Let X be a common solution to Eqs (4.24) and (4.25), then using the identities defined in Eqs (4.16) through (4.19), derive the Lyapunov matrix equation as follows

$$\begin{aligned} 0 &= (VX+N)B \\ &= VXB+(NB) \\ &= VXB+BN-VC \\ &= VXB-BV-VC \\ &= VXB-VC-(BV)X \\ &= VXB-VC-(VA)X \\ &= V(XB-C-AX) \end{aligned}$$

■

Theorem 4-5. If U^{-1} exist, and X is a common solution of equations (4.24) and (4.25), then X is also a solution to Eq (4.1).

Proof. Let X be a common solution to Eqs (4.24) and (4.25), then using the identities defined in Eqs (4.16) through (4.19), derive

the Lyapunov matrix equation as follows

$$0 = A(UX+N)$$

$$
\begin{aligned}
&= AUX+AM \\
&= AUX+(AM) \\
&= (AU)X+(UC+MB-CN) \\
&= (UA-CV)X+UC+MB-CN \\
&= UAX+UC-(CV)X+MB-CN \\
&= UAX+UC-CN+MB-CN \\
&= UAX+UC+(M)B \\
&= UAX+UC+(-UX)B \\
&= U(AX-XB+C)
\end{aligned}
$$

∎

As stated earlier, the first step in developing a solution for the Lyapunov matrix equation was deriving an intermediate expression for its solution. The two pairs of equations, (4.22), (4.23) and (4.24), (4.25), comprises this intermediate expression. Now, a common solution for each pair must be obtained.

Common Solution. Now that the solution of the Lyapunov matrix equation has been expressed by an alternate representation, the final step is to find the solution in terms of this representation. For Eqs (4.22) through (4.25), this means finding the common solution for these pairs of equations. In Odell (31), a common solution for n equations is derived using an $A_{1,2,3,4}$ generalized inverse. Here n will be fixed at two and an $A_{1,2}$ will be substituted in place of the $A_{1,2,3,4}$ generalized inverse used by Odell.

To understand why an $A_{1,2}$ generalized inverse can be used in place the $A_{1,2,3,4}$ generalized inverse used by Odell, examine the necessary and sufficient condition needed for a common solution to exist. Given the pair of equations

$$A_1X = B_1 \tag{4.28}$$

$$A_2X = B_2 \tag{4.29}$$

By Odell [31:272], the following must hold for Eqs (4.28) and (4.29) to have a common solution

$$A_1A_{1(1,2,3,4)}B_1 = B_1 \tag{4.30}$$

$$(A_2-A_2A_1A_1)(A_2-A_2A_1A_1)_{(1,2,3,4)}(B_2-A_1A_1B_1) = (B_2-A_1A_1B_1) \tag{4.31}$$

Examining the conditions expressed for these two equations reveals that an $A_{1,2}$ inverse will satisfy both conditions. This is a result of the partial definition of an $A_{1,2}$ generalized inverse which was expressed earlier in the form

$$AXA = A \tag{2.1}$$

remembering X here is an $A_{1,2}$ generalized inverse. For Eq (2.1) to hold, AX must equal an identity matrix of appropriate dimensions. Applying this result to the condition stated in Eqs (4.30) and (4.31) shows that the $A_{1,2}$ satisfies these conditions since the following equations also hold

$$A_1A_{1(1,2)}B_1 = B_1 \qquad (4.32)$$

$$(A_2-A_2A_1A_{1(1,2)})(A_2-A_2A_1A_{1(1,2)})_{(1,2)} \qquad (4.31)$$

$$(B_2-A_1A_1B_1) = (B_2-A_1A_1B_1)$$

since

$$A_1A_{1(1,2)} = I$$

$$(A_2-A_2A_1A_{1(1,2)})(A_2-A_2A_1A_{1(1,2)})_{(1,2)} = I$$

by the definition of an $A_{1,2}$ generalized inverse.

Thus using an $A_{1,2}$ generalized inverse, Odell's result can now be stated for a pair of equations and in terms of Eqs (4.24) and (4.25). Thus given

$$VX+N = 0 \qquad (4.24)$$

$$UX+M = 0 \qquad (4.25)$$

The common solutions for these equation, by Odell (31,272), is found from

$$X = V_{1,2}N+(I-V_{12}V) \qquad (4.34)$$

$$X = V_{1,2}N+(I-V_{12}V)[(U-UV_{12}V)_{12}(M-UV_{12}N)] + \qquad (4.35)$$

$$(I-V_{12}V)[I-(U-UV_{12}V)_{12}\ (U-UV_{12}V)]$$

if and only if the following holds

$$VV_{1,2}N = N \tag{4.36}$$

$$(U-V_{1,2}V)(U-V_{1,2}V)_{1,2}(M-UVN) = (M-UVN) \tag{4.37}$$

This result can also be extended for Eqs (4.23) and (4.24) where the unknown is on the left hand side. In this case, the necessary and suficient conditions, which determine whether or not Eqs (4.22) and (4.23) have a common solution, are rearranged to account for the form of Eqs (4.22) and (4.23). Specifically, the second characteristic of an $A_{1,2}$ generalized inverse is utilized to obtain the new expressions for the necessary and sufficient conditions and also the form of the common solutions for Eqs (4.22) and (4.23). This characteristic of the $A_{1,2}$ generalized inverse was stated earilier as

$$XAX = X \tag{2.2}$$

Now, applying Eq (2.2) to the conditions needed for a common solution to exist for Eqs (4.23) and (4.24) provides a similar result to that developed for Eqs (4.30) and (4.31). This result can be stated as

$$V_{1,2}VU = U \tag{4.38}$$

$$(N-NVV_{1,2})(N-NVV_{1,2})_{1,2}(M-NUV_{1,2}) = (M-NUV_{1,2}) \tag{4.39}$$

Now if Eqs (4.38) and (4.39) hold, then the common solution for Eqs

(4.22) and (4.23) can be obtained from the following

$$X = UV_{1,2}+(I-VV_{1,2}) \tag{4.40}$$

$$X = UV_{1,2}+(I-UV_{1,2})(M-NUV_{1,2})(N-NVV_{1,2})_{1,2}+ \\ (I-VV_{1,2})(I-(N-NVV_{1,2})(N-NVV_{1,2})_{1,2}) \tag{4.41}$$

Probably the most interesting result from the above application of a generalized inverse to the Lyapunov matrix equation is the potential savings obtained by substituting the $A_{1,2}$ generalized inverse over the $A_{1,2,3,4}$ generalized inverse. It should be apparent at this point, that an $A_{1,2}$ is easier to compute than an $A_{1,2,3,4}$ (see appendix C for a quantitative comparison of an $A_{1,2}$ versus an $A_{1,2,3,4}$). Thus if an $A_{1,2}$ satifies the needs of an application, it would certainly be more desirable than an $A_{1,2,3,4}$ for computational reasons. This is certainly the case for the application discussed above which leads back the comments made in the introductory remarks to this chapter regarding the use of a generalized inverses. Namely, the use of a lesser generalized inverse was probably not considered for solving for the common solution of multiple equations, as needed above, primarily due to the lack of stable and direct methods for computing such an inverse. The ST method provides the necessary stable and direct computational technique needed to make the $A_{1,2}$ of use in this application.

V. Conclusions and Recommendations

The ST method provides a simple, direct, and numerically stable technique for computing four generalized inverses of a matrix. It also provides a unified approach for computing these inverses over other techniques which normally only apply for computing a single generalized inverse. The ST method also highlights the computational differences between each gereralized inverse. This combined with the properties of a respective inverse provides insight into the use and cost of computing a generalized inverse for a given application. Thus, by taking into account the cost of computing an inverse and its properties, the ST method may help reveal where a lesser generalized inverse will satisfy the requirements over a more computational expensive inverse. This was certainly the case for finding the solution of the Lyapunov matrix equation. In summary, the ST method is a simple, direct, and unified technique for computing a generalized inverse.

There are two areas where the ST method, in particular its computer implementation, can be extended. First, further study is warranted in extending the ST method to unbound precision. The next logical extention would be to address matrices with single or multiple parameters. Both are recommended because the ST method is a good computational framework for large problems, and these potential applications deal with matrices of this type or require these capabilities.

Appendix A: Example 3

The purpose of example 1 was to introduce the ST method in a simple and direct manner. With the details of the ST method now well established, example 3 extends example 1 by taking a more complete look at the steps comprising the ST method. This includes the intermediate results of the row and column operations and the use of Penrose's four equations to test the computed generalized inverses. The matrix A in this example was taken from Noble (21:145,146). The example begins with the given matrix A of dimensions 6 x 4:

$$[A] = \begin{bmatrix} -1.000 & .000 & 1.000 & 2.000 \\ -1.000 & 1.000 & .000 & -1.000 \\ .000 & -1.000 & 1.000 & 3.000 \\ .000 & 1.000 & -1.000 & -3.000 \\ 1.000 & -1.000 & .000 & 1.000 \\ 1.000 & .000 & -1.000 & -2.000 \end{bmatrix}$$

First augment A with an identity matrix of dimensions 6 x 6 beside A and an identity matrix below A with dimensions 4 x 4 providing

$$\left[\begin{array}{c|c} A & I \\ \hline I & 0 \end{array}\right] = \left[\begin{array}{cccc|cccccc} -1.000 & .000 & 1.000 & 2.000 & 1.000 & .000 & .000 & .000 & .000 & .000 \\ -1.000 & 1.000 & .000 & -1.000 & .000 & 1.000 & .000 & .000 & .000 & .000 \\ .000 & -1.000 & 1.000 & 3.000 & .000 & .000 & 1.000 & .000 & .000 & .000 \\ .000 & 1.000 & -1.000 & -3.000 & .000 & .000 & .000 & 1.000 & .000 & .000 \\ 1.000 & -1.000 & .000 & 1.000 & .000 & .000 & .000 & .000 & 1.000 & .000 \\ 1.000 & .000 & -1.000 & -2.000 & .000 & .000 & .000 & .000 & .000 & 1.000 \\ \hline 1.000 & .000 & .000 & .000 & & & & & & \\ .000 & 1.000 & .000 & .000 & & & & & & \\ .000 & .000 & 1.000 & .000 & & & & & & \\ .000 & .000 & .000 & 1.000 & & & & & & \end{array}\right]$$

Using this augmented representation of A, the next step is to reduce the original matrix to a square identity matrix with dimensions equal to the rank of A. The reduction process uses a modified Guass-Jordan algorithm with a pivoting scheme which will perform, at the most, three pivots to locate the best diagonal element for both row and column operations. The following matrix shows the result of the first pivot call where three pivots were performed to locate the best diagonal element for reducing both the elements below the diagonal and the elements across from it:

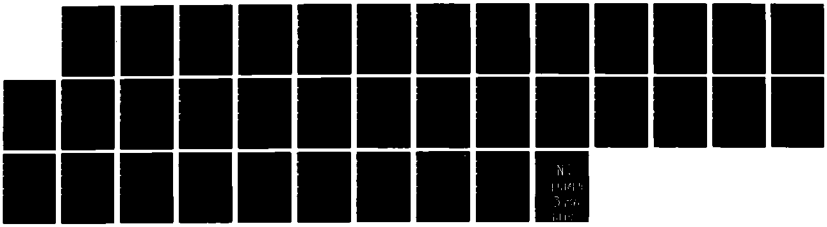
$$\left[\begin{array}{rrrr|rrrrrr}
3.000 & -1.000 & 1.000 & .000 & .000 & .000 & 1.000 & .000 & .000 & .000 \\
-1.000 & 1.000 & .000 & -1.000 & .000 & 1.000 & .000 & .000 & .000 & .000 \\
2.000 & .000 & 1.000 & -1.000 & 1.000 & .000 & .000 & .000 & .000 & .000 \\
-3.000 & 1.000 & -1.000 & .000 & .000 & .000 & .000 & 1.000 & .000 & .000 \\
1.000 & -1.000 & .000 & 1.000 & .000 & .000 & .000 & .000 & 1.000 & .000 \\
-2.000 & .000 & -1.000 & 1.000 & .000 & .000 & .000 & .000 & .000 & 1.000 \\
\hline
.000 & .000 & .000 & 1.000 & & & & & & \\
.000 & 1.000 & .000 & .000 & & & & & & \\
.000 & .000 & 1.000 & .000 & & & & & & \\
1.000 & .000 & .000 & .000 & & & & & &
\end{array}\right]$$

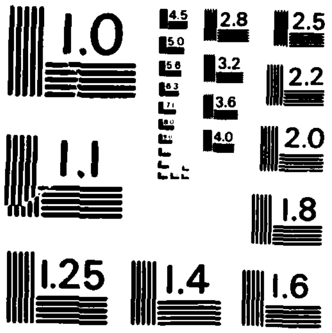
With the best diagonal element established, the first column, of the now shift matrix A, is eliminated with row operations. The result is shown in the following matrix:

$$\left[\begin{array}{rrrr|rrrrrr}
3.000 & -1.000 & 1.000 & .000 & .000 & .000 & 1.000 & .000 & .000 & .000 \\
.000 & .667 & .333 & -1.000 & .000 & 1.000 & .333 & .000 & .000 & .000 \\
.000 & .667 & .333 & -1.000 & 1.000 & .000 & -.667 & .000 & .000 & .000 \\
.000 & .000 & .000 & .000 & .000 & .000 & 1.000 & 1.000 & .000 & .000 \\
.000 & -.667 & -.333 & 1.000 & .000 & .000 & -.333 & .000 & 1.000 & .000 \\
.000 & -.667 & -.333 & 1.000 & .000 & .000 & .667 & .000 & .000 & 1.000 \\
\hline
.000 & .000 & .000 & 1.000 & & & & & & \\
.000 & 1.000 & .000 & .000 & & & & & & \\
.000 & .000 & 1.000 & .000 & & & & & & \\
1.000 & .000 & .000 & .000 & & & & & &
\end{array}\right]$$

The pivot process continues, but now the pivot is performed around

AD-A163 840 A NEW METHOD FOR COMPUTING THE GENERALIZED INVERSES OF 2/2
A MATRIX AND ITS A (U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI C F MURRAY
UNCLASSIFIED DEC 85 AFIT/GOR/MA/85D-2 F/G 12/1 NL




1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

the second diagonal element. This pivot call resulted in one pivot which is shown below:

$$\left[\begin{array}{rrrr|rrrrrr}
3.000 & .000 & 1.000 & -1.000 & .000 & .000 & 1.000 & .000 & .000 & .000 \\
.000 & -1.000 & .333 & .667 & .000 & 1.000 & .333 & .000 & .000 & .000 \\
.000 & -1.000 & .333 & .667 & 1.000 & .000 & -.667 & .000 & .000 & .000 \\
.000 & .000 & .000 & .000 & .000 & .000 & 1.000 & 1.000 & .000 & .000 \\
.000 & 1.000 & -.333 & -.667 & .000 & .000 & -.333 & .000 & 1.000 & .000 \\
.000 & 1.000 & -.333 & -.667 & .000 & .000 & .667 & .000 & .000 & 1.000 \\
\hline
.000 & 1.000 & .000 & .000 & & & & & & \\
.000 & .000 & .000 & 1.000 & & & & & & \\
.000 & .000 & 1.000 & .000 & & & & & & \\
1.000 & .000 & .000 & .000 & & & & & &
\end{array}\right]$$

Again, the next step is to reduce the elements of only A below the diagonal element with row operations which produces

$$\left[\begin{array}{rrrr|rrrrrr}
3.000 & .000 & 1.000 & -1.000 & .000 & .000 & 1.000 & .000 & .000 & .000 \\
.000 & -1.000 & .333 & .667 & .000 & 1.000 & .333 & .000 & .000 & .000 \\
.000 & .000 & .000 & .000 & 1.000 & -1.000 & -1.000 & .000 & .000 & .000 \\
.000 & .000 & .000 & .000 & .000 & .000 & 1.000 & 1.000 & .000 & .000 \\
.000 & .000 & .000 & .000 & .000 & 1.000 & .000 & .000 & 1.000 & .000 \\
.000 & .000 & .000 & .000 & .000 & 1.000 & 1.000 & .000 & .000 & 1.000 \\
\hline
.000 & 1.000 & .000 & .000 & & & & & & \\
.000 & .000 & .000 & 1.000 & & & & & & \\
.000 & .000 & 1.000 & .000 & & & & & & \\
1.000 & .000 & .000 & .000 & & & & & &
\end{array}\right]$$

The next potential diagonal element is zero, and all possible pivot candidates are also zeroes. This means all rows below the last diagonal element have been reduced and no further row operations are needed. This also denotes that the T submatrix is almost defined since it is a function of the row operations. The only remaining operations which may impact T is dividing by the diagonal element after the column operations have been completed. This will be more apparent in a later matrix.

Since the pivots performed prior to each set of row operations were designed to look ahead to provide a good computational basis for the column operations, no futher pivoting is performed during the column operations. Instead, the column operations are performed directly on the remaining non zero elements. The result of the first set of column operations on row one of A is shown below:

$$\left[\begin{array}{rrrr|rrrrrr}
3.000 & .000 & .000 & .000 & .000 & .000 & 1.000 & .000 & .000 & .000 \\
.000 & -1.000 & .333 & .667 & .000 & 1.000 & .333 & .000 & .000 & .000 \\
.000 & .000 & .000 & .000 & 1.000 & -1.000 & -1.000 & .000 & .000 & .000 \\
.000 & .000 & .000 & .000 & .000 & .000 & 1.000 & 1.000 & .000 & .000 \\
.000 & .000 & .000 & .000 & .000 & 1.000 & .000 & .000 & 1.000 & .000 \\
.000 & .000 & .000 & .000 & .000 & 1.000 & 1.000 & .000 & .000 & 1.000 \\
\hline
.000 & 1.000 & .000 & .000 & & & & & & \\
.000 & .000 & .000 & 1.000 & & & & & & \\
.000 & .000 & 1.000 & .000 & & & & & & \\
1.000 & .000 & -.333 & .333 & & & & & &
\end{array}\right]$$

The next set of column operations reduce the remaining non zero elements of row two across from the second diagonal element. This is shown in the following matrix

$$\left[\begin{array}{rrrr|rrrrrr}
3.000 & .000 & .000 & .000 & .000 & .000 & 1.000 & .000 & .000 & .000 \\
.000 & -1.000 & .000 & .000 & .000 & 1.000 & .333 & .000 & .000 & .000 \\
.000 & .000 & .000 & .000 & 1.000 & -1.000 & -1.000 & .000 & .000 & .000 \\
.000 & .000 & .000 & .000 & .000 & .000 & 1.000 & 1.000 & .000 & .000 \\
.000 & .000 & .000 & .000 & .000 & 1.000 & .000 & .000 & 1.000 & .000 \\
.000 & .000 & .000 & .000 & .000 & 1.000 & 1.000 & .000 & .000 & 1.000 \\
\hline
.000 & 1.000 & .333 & .667 & & & & & & \\
.000 & .000 & .000 & 1.000 & & & & & & \\
.000 & .000 & 1.000 & .000 & & & & & & \\
1.000 & .000 & -.333 & .333 & & & & & &
\end{array}\right]$$

With only diagonal elements remaing in A, the final step needed to reduce A to the desired identity matrix is to divide through by the diagonal elements. With all row and column operations complete, the submatrices are now defined, and the representation, derived in theorem 2-1 as equivalent to the augment matrix A, is obtained. This representation is shown in the next equation along with the sub-matrices ɔ, T, M, and N highlighted with the partitioning lines:

$$\left[\begin{array}{c|c|c} I_r & 0 & T \\ \hline 0 & 0 & M \\ \hline S & N & 0 \end{array}\right] = \left[\begin{array}{cc|cc|cccccc} 1.000 & .000 & .000 & .000 & .000 & .000 & .333 & .000 & .000 & .000 \\ .000 & 1.000 & .000 & .000 & .000 & -1.000 & -.333 & .000 & .000 & .000 \\ \hline .000 & .000 & .000 & .000 & 1.000 & -1.000 & -1.000 & .000 & .000 & .000 \\ .000 & .000 & .000 & .000 & .000 & .000 & 1.000 & 1.000 & .000 & .000 \\ .000 & .000 & .000 & .000 & .000 & 1.000 & .000 & .000 & 1.000 & .000 \\ .000 & .000 & .000 & .000 & .000 & 1.000 & 1.000 & .000 & .000 & 1.000 \\ \hline .000 & 1.000 & .333 & .667 & & & & & & \\ .000 & .000 & .000 & 1.000 & & & & & & \\ .000 & .000 & 1.000 & .000 & & & & & & \\ 1.000 & .000 & -.333 & .333 & & & & & & \end{array}\right]$$

Using the above matrix, an $A_{1,2}$ generalized inverse is computed directly from the product of the S and T submatrices. This product is shown in the following matrix

$$\begin{bmatrix} .000 & -1.000 & -.333 & .000 & .000 & .000 \\ .000 & .000 & .000 & .000 & .000 & .000 \\ .000 & .000 & .000 & .000 & .000 & .000 \\ .000 & .000 & .333 & .000 & .000 & .000 \end{bmatrix}$$

Applying the two equations an $A_{1,2}$ *generalized inverse satisfies* provides the original matrix and the computed inverse as follows for each respective equation.

$AA_{1,2}A = A$

$$\begin{bmatrix} -1.000 & .000 & 1.000 & 2.000 \\ -1.000 & 1.000 & .000 & -1.000 \\ .000 & -1.000 & 1.000 & 3.000 \\ .000 & 1.000 & -1.000 & -3.000 \\ 1.000 & -1.000 & .000 & 1.000 \\ 1.000 & .000 & -1.000 & -2.000 \end{bmatrix}$$

$A_{1,2}A\ A_{1,2} = A_{1,2}$

$$\begin{bmatrix} .000 & -1.000 & -.333 & .000 & .000 & .000 \\ .000 & .000 & .000 & .000 & .000 & .000 \\ .000 & .000 & .000 & .000 & .000 & .000 \\ .000 & .000 & .333 & .000 & .000 & .000 \end{bmatrix}$$

The next inverse of concern, an $A_{1,2,3}$ generalized inverse, is again computed from the product of the S and T submatrices, but after each row in T has been orthogonalized to every row of M. Example 2, in chapter 3, provides more details regarding the orthogonalization technique used. Thus an $A_{1,2,3}$ generalized inverse is computed from the following framework which shows the orthogonalized T and M submatrices and the previous S submatrix:

$$\left[\begin{array}{c|c|c} I_r & 0 & T \\ \hline 0 & 0 & M \\ \hline S & N & 0 \end{array}\right] = \left[\begin{array}{rr|rr|rrrrrr} 1.000 & .000 & .000 & .000 & .056 & -.056 & .111 & -.111 & .056 & -.056 \\ .000 & 1.000 & .000 & .000 & -.222 & -.278 & .056 & -.056 & .278 & .222 \\ \hline .000 & .000 & .000 & .000 & 1.000 & -.250 & -.250 & .250 & .250 & .500 \\ .000 & .000 & .000 & .000 & .000 & -.200 & .600 & 1.000 & .200 & -.400 \\ .000 & .000 & .000 & .000 & .000 & .667 & -.333 & .000 & 1.000 & -.333 \\ .000 & .000 & .000 & .000 & .000 & 1.000 & 1.000 & .000 & .000 & 1.000 \\ \hline .000 & 1.000 & .333 & .667 & & & & & & \\ .000 & .000 & .000 & 1.000 & & & & & & \\ .000 & .000 & 1.000 & .000 & & & & & & \\ 1.000 & .000 & -.333 & .333 & & & & & & \end{array}\right]$$

The product of S and T in this case provides the following matrix

$$\begin{bmatrix} -.222 & -.278 & .056 & -.056 & .278 & .222 \\ .000 & .000 & .000 & .000 & .000 & .000 \\ .000 & .000 & .000 & .000 & .000 & .000 \\ .056 & -.056 & .111 & -.111 & .056 & -.056 \end{bmatrix}$$

This $A_{1,2,3}$ satisfies the previous two equations and in addition the equation

$$(AA_{1,2,3})^* = (AA_{1,2,3})$$

where the product of the $A_{1,2,3}$ and the original matrix A is the symmetric matrix shown below

$$\begin{bmatrix} .333 & .167 & .167 & -.167 & -.167 & -.333 \\ .167 & .333 & -.167 & .167 & -.333 & -.167 \\ .167 & -.167 & .333 & -.333 & .167 & -.167 \\ -.167 & .167 & -.333 & .333 & -.167 & .167 \\ -.167 & -.333 & .167 & -.167 & .333 & .167 \\ -.333 & -.167 & -.167 & .167 & .167 & .333 \end{bmatrix}$$

To obtain an $A_{1,2,4}$ generalized inverse, the S and N submatrices are orthogonalized like the T an M submatrices were orthogonalized for the $A_{1,2,3}$ generalized inverse above. The product of the orthogonalized S submatrix and the non orthogonalized T submatrix form the $A_{1,2,4}$ generalized inverse. The following matrix shows the computation framework needed to compute this inverse:

$$\left[\begin{array}{c|c|c} I_r & 0 & T \\ \hline 0 & 0 & M \\ \hline S & N & 0 \end{array}\right] = \left[\begin{array}{cc|cc|cccccc} 1.000 & .000 & .000 & .000 & .000 & .000 & .333 & .000 & .000 & .000 \\ .000 & 1.000 & .000 & .000 & .000 & -1.000 & -.333 & .000 & .000 & .000 \\ \hline .000 & .000 & .000 & .000 & 1.000 & -1.000 & -1.000 & .000 & .000 & .000 \\ .000 & .000 & .000 & .000 & .000 & .000 & 1.000 & 1.000 & .000 & .000 \\ .000 & .000 & .000 & .000 & .000 & 1.000 & .000 & .000 & 1.000 & .000 \\ .000 & .000 & .000 & .000 & .000 & 1.000 & 1.000 & .000 & .000 & 1.000 \\ \hline -.059 & .647 & .286 & .667 & & & & & & \\ -.235 & -.412 & -.071 & 1.000 & & & & & & \\ .294 & -.235 & 1.000 & .000 & & & & & & \\ .824 & -.059 & -.357 & .333 & & & & & & \end{array}\right]$$

The product of S and T in this case produce

$$\begin{bmatrix} .000 & -.647 & -.235 & .000 & .000 & .000 \\ .000 & .412 & .059 & .000 & .000 & .000 \\ .000 & .235 & .176 & .000 & .000 & .000 \\ .000 & .059 & .294 & .000 & .000 & .000 \end{bmatrix}$$

which satifies the first two equations and the equation

$$(A_{1,2,4}\ A)^* = A_{1,2,4}\ A$$

The product of $A_{1,2,4}$ and A again produce a symmetric matrix:

$$\begin{bmatrix} .647 & -.412 & -.235 & -.059 \\ -.412 & .353 & .059 & -.235 \\ -.235 & .059 & .176 & .294 \\ -.059 & -.235 & .294 & .824 \end{bmatrix}$$

The final generalized inverse an $A_{1,2,3,4}$, is computed from the orthogonalized version of both S and T. For this example, the computational framework needed to compute this inverse is

$$\left[\begin{array}{c|c|c} I_r & 0 & T \\ \hline 0 & 0 & M \\ \hline S & N & 0 \end{array}\right] = \left[\begin{array}{cc|cc|cccccc} 1.000 & .000 & .000 & .000 & .056 & -.056 & .111 & -.111 & .056 & -.056 \\ .000 & 1.000 & .000 & .000 & -.222 & -.278 & .056 & -.056 & .278 & .222 \\ \hline .000 & .000 & .000 & .000 & 1.000 & -.250 & -.250 & .250 & .250 & .500 \\ .000 & .000 & .000 & .000 & .000 & -.200 & .600 & 1.000 & .200 & -.400 \\ .000 & .000 & .000 & .000 & .000 & .667 & -.333 & .000 & 1.000 & -.333 \\ .000 & .000 & .000 & .000 & .000 & 1.000 & 1.000 & .000 & .000 & 1.000 \\ \hline -.059 & .647 & .286 & .667 & & & & & & \\ -.235 & -.412 & -.071 & 1.000 & & & & & & \\ .294 & -.235 & 1.000 & .000 & & & & & & \\ .824 & -.059 & -.357 & .333 & & & & & & \end{array}\right]$$

The A inverse is

$$\begin{bmatrix} -.147 & -.176 & .029 & -.029 & .176 & .147 \\ .078 & .127 & -.049 & .049 & -.127 & -.078 \\ .069 & .049 & .020 & -.020 & -.049 & -.069 \\ .059 & -.029 & .088 & -.088 & .029 & -.059 \end{bmatrix}$$

where all four previous equations are satisfied.

Appendix B: Generalized Inverse Program

```
C*****************************************************************
C                                                                *
C   DATE: 11/11/85                                               *
C   VERSION: 1.0                                                 *
C   TITLE: GENERALIZED INVERSE COMPUTATION AND VALIDATION        *
C   LANGUAGE: FORTRAN 77                                         *
C                                                                *
C   USE: COMPUTE AND CKECKS THE COMPUTATION OF FOUR GENERALIZED  *
C        INVERSES.                                               *
C                                                                *
C   CONTENTS:                                                    *
C          MAIN PROGRAM  - INPUTS MATRIX                         *
C                        - REDUCES MATRIX WITH MODIFIED          *
C                          GUASS-JORDAN                          *
C                        - CONTROLS TO COMPUTE GENERALIZED       *
C                          INVERSES                              *
C          IPIVOT        - PERFORMS A THREE STEP PIVOT           *
C          MULT          - MULTIPLIES TWO MATRICES PLACING THE   *
C                          RESULT INTO AN OPTIONAL THIRD MATRX   *
C          GROW          - GRAM-SCHMIDT COMPUTATION FOR ROWS     *
C          GCOLUMN       - GRAM-SCHMIDT FOR COLUMNS              *
C          PRINTA        - GENERAL MATRIX PRINT ROUTINE          *
C                                                                *
C   FUNCTION:                                                    *
C        COMPUTES THE A1,2, A1,2,3, A1,2,4, A1,2,3,4 GENERALIZED *
C        INVERSES AND VALIDATES EACH THE FOUR CONDITIONS DEFINED *
C        BY PENROSE.                                             *
C                                                                *
C*****************************************************************
```

```
C***********************************************************
C                                                          *
C   DATE: 11/11/85                                         *
C   VERSION: 1.0                                           *
C                                                          *
C   NAME: GINVERSE (MAIN PROGRAM)                          *
C   DESCRIPTION: CONTROLS THE REDUCTION OF THE INPUT MATRIX TO *
C        AN IDENTITY MATRIX AND ALSO CALLS ALL NECESSARY   *
C        SUBROUTINES TO COMPUTE A GIVEN INVERSE            *
C                                                          *
C   PASSED VARIABLES: A,ST,HOLD,R,C,ASIZE,RANK             *
C   RETURNS: NONE                                          *
C   FILES READ: MATRIX INPUT                               *
C   SUBROUTINES CALLED: IPIVOT,MULT,GROW,GCOLUMN,PRINTA    *
C   CALLING SUBROUTINE: NONE                               *
C                                                          *
C***********************************************************

      PROGRAM GINVERSE

C*************************************
C ARRAY TYPE AND DIMENSIONS/ DECLARATIONS
C*************************************

      DOUBLE PRECISION    A
      DIMENSION           A(1:10,1:10)

      DOUBLE PRECISION    ST
      DIMENSION           ST(1:10,1:10)

      DOUBLE PRECISION    ORG
      DIMENSION           ORG(1:10,1:10)

      DOUBLE PRECISION    HOLD
      DIMENSION           HOLD(1:10,1:10)

      DOUBLE PRECISION    SR
      DIMENSION           SR(1:10,1:10)

C************************
C VARIABLE TYPE DECLARATIONS
C************************
```

```
         INTEGER          R
         INTEGER          C
         INTEGER          ASIZE
         INTEGER          MINRC
         INTEGER          RANK
         INTEGER          I
         INTEGER          J
         INTEGER          K
         INTEGER          TEST
         INTEGER          OPTION
         INTEGER          FORMATT
         INTEGER          HEADER
         INTEGER          SELECT
         INTEGER          TEST
         INTEGER          PIVOTRLT

         CHARACTER        FILENAM*10
         DOUBLE PRECISION MULTIPLIER

         PARAMETER(ZERO=5.E-15)

C******************
C DATA INITIALIZATION
C******************

1000     FORMAT(1X,' SELECT OUTPUT OPTION')
1001     FORMAT(1X,'   OUTPUT TO SCREEN   - SELECT 1')
1002     FORMAT(1X,'   OUTPUT TO PRINTER  - SELECT 2')
1003     FORMAT(1X)
1004     FORMAT(1X,' SELECTION: ')
1005     FORMAT(I2)
1006     FORMAT(1X,' SELECT FORMAT OPTION')
1007     FORMAT(1X,'   XX.XX     - SELECT  1')
1008     FORMAT(1X,'   XXX.XX    - SELECT  2')
1009     FORMAT(1X,'   XX.XXX    - SELECT  3')
1010     FORMAT(1X,'   XX.XXXX   - SELECT  4')
1011     FORMAT(1X,' SELECT INPUT FILE')
1012     FORMAT(1X,'  CHOOSE FROM LIST OR SELECT A NEW FILE:')
1013     FORMAT(1X,'    1)  X.CASE 1')
1014     FORMAT(1X,'    2)  X.CASE 2')
1015     FORMAT(1X,'    3)  X.CASE 3')
1016     FORMAT(1X,'    4)  X.CASE 4')
```

```
1017  FORMAT(1X,'     5)  X.NOBLES MATRIX')
1018  FORMAT(1X,'     6)  NEW FILE')
1019  FORMAT(1X,'  INPUT FILE NAME IN APPROPRIATE FORMAT: ')
1020  FORMAT(A10)
1021  FORMAT('1***** GENERALIZED INVERSES FOR FILE:',A10,' ******')
1022  FORMAT(1X,'***** ST METHOD        LANGUAGE: FORTRAN 77')
1023  FORMAT(I2)

      WRITE(9,1000)
      WRITE(9,1001)
      WRITE(9,1002)
      WRITE(9,1003)
      WRITE(9,1004)
      READ (9,1005) OPTION

      WRITE(9,1003)
      WRITE(9,1006)
      WRITE(9,1007)
      WRITE(9,1008)
      WRITE(9,1009)
      WRITE(9,1010)
      WRITE(9,1003)
      WRITE(9,1004)
      READ (9,1005) FORMATT

      WRITE(9,1003)
      WRITE(9,1011)
      WRITE(9,1012)
      WRITE(9,1013)
      WRITE(9,1014)
      WRITE(9,1015)
      WRITE(9,1016)
      WRITE(9,1017)
      WRITE(9,1018)
      WRITE(9,1003)
      WRITE(9,1004)
      READ (9,1005) SELECT

      IF (SELECT.EQ.1) THEN
          OPEN(1,FILE="X.CASE 1",STATUS="OLD")
          FILENAM="X.CASE 1"
      END IF
      IF (SELECT.EQ.2) THEN
```

```
              OPEN(1,FILE="X.CASE 2",STATUS="OLD")
              FILENAM="X.CASE 2"
          END IF
          IF (SELECT.EQ.3) THEN
              OPEN(1,FILE="X.CASE 3",STATUS="OLD")
              FILENAM="X.CASE 3"
          END IF
          IF (SELECT.EQ.4) THEN
              OPEN(1,FILE="X.CASE 5",STATUS="OLD")
              FILENAM="X.CASE 5"
          END IF
          IF (SELECT.EQ.5) THEN
              OPEN(1,FILE="X.NOBLES MATRIX",STATUS="OLD")
              FILENAM="X.NOBLES MATRIX"
          END IF
          IF (SELECT.EQ.6) THEN
              WRITE(9,1003)
              WRITE(9,1019)
              READ (9,1020) FILENAM
              OPEN(1,FILE=FILENAM,STATUS="OLD")
          END IF
          WRITE(9,1003)
          WRITE(9,1003)
          IF (OPTION.EQ.1) THEN
              WRITE(9,1021) FILENAM
              WRITE(9,1022)
           ELSE
              WRITE(9,1021) FILENAM
              WRITE(9,1022)
          END IF

C READ SELECTED FILE

          READ (1,1023) R
          READ (1,1023) C
          ASIZE = R+C
          DO 2001 I=1,ASIZE
              DO 2000 J=1,ASIZE
                 READ (1,*) A(I,J)
                ORG(I,J) = A(I,J)
 2000        CONTINUE
 2001     CONTINUE
```

```
C AUGMENT A WITH THE APPROPRIATE IDENTITY MATRICES

        DO 2011 I=1,R
            DO 2010 J=C+1,ASIZE
                TEST = J-C
                IF (TEST.EQ.I) THEN
                    A(I,J) = 1.0
                  ELSE
                    A(I,J) = 0.0
                END IF
 2010       CONTINUE
 2011   CONTINUE

        DO 2020 I=R+1,ASIZE
            DO 2021 J=1,C
                TEST = I-R
                IF (TEST.EQ.J) THEN
                   A(I,J) = 1.0
                 ELSE
                   A(I,J) = 0.0
                END IF
2020        CONTINUE
2021    CONTINUE


C DETERMINE THE MINIMUM RANK

        IF (R.LT.C) THEN
            MINRC = R
         ELSE
            MINRC = C
        END IF
        CALL PRINTA(1,1,ASIZE,ASIZE,1,OPTION,FORMATT,R,C,A)
        IF (OPTION.EQ.1) READ (9,*) XX


C*************************************************************
C*************************************************************
```

```
C***************
C MATRIX REDUCTION
C***************

      DO 3020 I=1,R-1
         PIVOTRLT=IPIVOT(R,C,MINRC,ASIZE,I,A)
         IF (PIVOTRLT.EQ.1) THEN
            DO 3010 K=I+1,R
               MULTIPLIER = A(K,I)/A(I,I)
                A(K,I) = 0.0
                DO 3000 J=I+1,ASIZE
                      A(K,J) = A(K,J)-(A(I,J)*MULTIPLIER)
3000           CONTINUE
3010        CONTINUE
          ELSE
               GO TO 3021
         END IF
3020  CONTINUE

3021  CONTINUE

      I = 1

3030  IF ((ABS(A(I,I)).GT.ZERO).AND.(I.LT.C)) THEN

          DO 3050 J=I+1,C
           MULTIPLIER = A(I,J)/A(I,I)
             A(I,J) = 0.0
             DO 3040 K=R+1,ASIZE
                A(K,J) = A(K,J)-(A(K,I)*MULTIPLIER)
3040         CONTINUE
3050     CONTINUE
          I =I+1
         GO TO 3030
      END IF

      RANK = 0
      I =1
```

```
3060 IF ((ABS(A(I,I)).GT.ZERO).AND.(I.LE.MINRC)) THEN
         RANK = RANK+1
         MULTIPLIER = A(I,I)
         DO 3070 J=1,ASIZE
            A(I,J) = A(I,J)/MULTIPLIER
3070     CONTINUE

         I = I+1
         GO TO 3060
      END IF

      CALL PRINTA(1,1,ASIZE,ASIZE,2,OPTION,FORMATT,R,C,A)
      IF (OPTION.EQ.1) READ (9,*) XX


C*************************************************************
C*************************************************************




C***********************************
C COMPUTE THE FOUR GENERALIZED INVERSES
C
C     A1,2
C     A1,2,3
C     A1,2,4
C     A1,2,3,4
C
C***********************************



C***** COMPUTE A1,2 GENERALIZED INVERSE

         CALL MULT(R+1,1,C,1,C+1,R,RANK,A,A,ST)
         CALL PRINTA(1,1,C,R,3,OPTION,FORMATT,C,R,ST)
         IF (OPTION.EQ.1) READ (9,*) XX
```

```
C        CHECK A1,2 USING

C          1) (A A1,2 A) = A

           CALL MULT(1,1,R,1,1,R,C,ORG,ST,HOLD)
           CALL MULT(1,1,R,1,1,C,R,HOLD,ORG,HOLD)
           CALL PRINTA(1,1,R,C,4,OPTION,FORMATT,R,C,HOLD)
           IF (OPTION.EQ.1) READ (9,*) XX

C          2) (A1,2 A A1,2) = A1,2 )

           CALL MULT(1,1,C,1,1,C,R,ST,ORG,HOLD)
           CALL MULT(1,1,C,1,1,R,C,HOLD,ST,HOLD)
           CALL PRINTA(1,1,C,R,5,OPTION,FORMATT,C,R,HOLD)
           IF (OPTION.EQ.1) READ (9,*) XX


C*****COMPUTE A1,2,3 GENERALIZED INVERSE

           DO 4010 I=1,R
              DO 4000 J=C+1,ASIZE
               SR(I,J-C) = A(I,J)
4000          CONTINUE
4010       CONTINUE

           CALL GSROW(RANK,R,C,ASIZE,A)
           CALL MULT(R+1,1,C,1,C+1,R,RANK,A,A,ST)
           CALL PRINTA(1,1,C,R,6,OPTION,FORMATT,C,R,ST)
           IF (OPTION.EQ.1) READ (9,*) XX

C          CHECK A1,2,3 USING (A A1,2,3)* = (A A1,2,3)

           CALL MULT(1,1,R,1,1,R,C,ORG,ST,HOLD)
           CALL PRINTA(1,1,R,R,7,OPTION,FORMATT,R,R,HOLD)
           IF (OPTION.EQ.1) READ (9,*) XX
```

```
C*****COMPUTE A_{1,2,4} GENERALIZED INVERSE

            CALL GSCOL(RANK,R,C,ASIZE,A)
            CALL MULT(R+1,1,C,1,1,R,RANK,A,SR,ST)
            CALL PRINTA(1,1,C,R,8,OPTION,FORMATT,C,R,ST)
            IF (OPTION.EQ.1) READ (9,*) XX

C        CHECK A_{1,2,4} USING (A_{1,2,4} A)* = A_{1,2,4} A)

            CALL MULT(1,1,C,1,1,C,R,ST,ORG,HOLD)
            CALL PRINTA(1,1,C,C,9,OPTION,FORMATT,C,C,HOLD)
            IF (OPTION.EQ.1) READ (9,*) XX


C*****COMPUTE A_{1,2,3,4} GENERALIZED INVERSE

            CALL MULT(R+1,1,C,1,C+1,R,RANK,A,A,ST)
            CALL PRINTA(1,1,C,R,10,OPTION,FORMATT,C,R,ST)
            IF (OPTION.EQ.1) READ (9,*) XX

C        CHECK A_{1,2,3,4} USING CONDITIONS APPLIED TO
                         A_{1,2}  A_{1,2,3} and A_{1,2,4}

C              CHECK A_{1,2,3,4} USING (A A_{1,2,3,4})* = (A A_{1,2,3,4})
                  CALL MULT(1,1,R,1,1,R,C,ORG,ST,HOLD)
                  CALL PRINTA(1,1,R,R,11,OPTION,FORMATT,R,R,HOLD)
                  IF (OPTION.EQ.1) READ (9,*) XX

C              CHECK A_{1,2,3,4} USING (A_{1,2,3,4} A)* = A_{1,2,3,4} A)
                  CALL MULT(1,1,C,1,1,C,R,ST,ORG,HOLD)
                  CALL PRINTA(1,1,C,C,12,OPTION,FORMATT,C,C,HOLD)
                  IF (OPTION.EQ.1) READ (9,*) XX


         STOP
         END
```

```
C**********************************************************
C                                                         *
C     DATE: 11/11/85                                      *
C     VERSION: 1.0                                        *
C
C     NAME: IPIVOT                                        *
C     DESCRIPTION: FINDS PIVOT ELEMENTS AND SWITCHES ROWS AND/ *
C                  OR COLUMNS.                            *
C     PASSED VARIABLES: R,C,MINRC,ASIZE,RANK,A            *
C     RETURNS: FUNCTION VALUE                             *
C     FILES READ: NONE                                    *
C     SUBROUTINES CALLED: NONE                            *
C     CALLING SUBROUTINE: GIINVERSE                       *
C                                                         *
C**********************************************************

      FUNCTION IPIVOT(R,C,MINRC,ASIZE,CD,A)

      DOUBLE PRECISION  A
      DIMENSION         A(1:10,1:10)
      INTEGER           R
      INTEGER           C
      INTEGER           ASIZE
      INTEGER           MINRC
      DOUBLE PRECISION  LARGEST
      DOUBLE PRECISION  EXCHANGE
      INTEGER           SWITCH
      INTEGER           SD
      INTEGER           SR
      INTEGER           SC
      INTEGER           I
      INTEGER           CD
      INTEGER           K
      INTEGER           COUNT
      INTEGER           ROW
      INTEGER           LASTSD

      PARAMETER(ZERO=5.E-15)

      SD = CD
      SR = CD
      SC = CD
      SWITCH = 0
```

```
          ROW = CD
          LASTSD = 0
          COUNT = 0

    1000  IF ((SD.LE.MINRC).AND.(SWITCH.EQ.0.0)) THEN

             IF (LASTSD.NE.SD) ROW=SD
             IF (LASTSD.EQ.SD).AND.(SC.EQ.CD) GO TO 1021
             LASTSD = SD
             IF (SC.EQ.CD) LARGEST=ABS(A(SD,SD))

             DO 1020 K=SD+1,R
               IF (ABS(A(K,ROW)).LE.LARGEST) GO TO 1010
                 LARGEST = ABS(A(K,ROW))
                 SR = K
                 SC = ROW
    1010       CONTINUE
    1020     CONTINUE

    1021     COUNT = COUNT+1
             IF ((LARGEST.GT.0.0).AND.(COUNT.GT.1)) SWITCH=1
             IF ((SWITCH.EQ.1).AND.(COUNT.GT.1)) GO TO 1050

             DO 1040 I=SD+1,C
                 IF (ABS(A(SD,I)).LE.LARGEST) GO TO 1030
                   LARGEST = ABS(A(SD,I))
                   SC = I
                   SR = SD
                   ROW = I
    1030         CONTINUE
    1040     CONTINUE

    1050     CONTINUE
             IF ((SWITCH.EQ.0).AND.(SC.EQ.CD).AND.(SR.EQ.CD)) SD=SD+1
             GO TO 1000
          END IF


          IF ((SC.EQ.CD).AND.(SR.EQ.CD).AND.(LARGEST.GT.0.0)) THEN
             SC = SD
             SR = SD
          END IF
```

```
      IF (SC.NE.CD) THEN
        DO 1060 I=1,ASIZE
          EXCHANGE = A(I,SC)
          A(I,SC) = A(I,CD)
          A(I,CD) = EXCHANGE
1060    CONTINUE
      END IF


      IF (SR.NE.CD) THEN
        DO 1070 I=1,ASIZE
          EXCHANGE = A(SR,I)
          A(SR,I) = A(CD,I)
          A(CD,I) = EXCHANGE
1070    CONTINUE
      END IF


      IF (ABS(A(CD,CD)).GT.ZERO) THEN
          IPIVOT=1
       ELSE
          IPIVOT=0
      END IF


      RETURN
      END

C*************************************************************
C*************************************************************
```

```
C*****************************************************************
C                                                                *
C      DATE: 11/11/85                                            *
C      VERSION: 1.0                                              *
C                                                                *
C      NAME: PRINTS                                              *
C      DESCRIPTION: PRINTS HEADER VARIABLE HEADER, FORMAT AND    *
C                   MATRIX DATA BASED ON VARIABLES PASSED        *
C      PASSED VARIABLES: STR, STCER,EC,HEADER,FORMATT,R,C,A      *
C      RETURNS: NONE                                             *
C      FILES READ: NONE                                          *
C      SUBROUTINES CALLED: NONE                                  *
C      CALLING SUBROUTINE: GINVERSE                              *
C                                                                *
C*****************************************************************

      SUBROUTINE PRINTA(STR,STC,ER,EC,HEADER,OPTION,FORMATT,R,C,A)

      DOUBLE PRECISION  A
      DIMENSION         A(1:10,1:10)
      INTEGER           R
      INTEGER           C
      INTEGER           STR
      INTEGER           STC
      INTEGER           ER
      INTEGER           EC
      INTEGER           HEADER
      INTEGER           OPTION
      INTEGER           FORMATT
      INTEGER           I
      INTEGER           J

      PARAMETER(ZERO=5.E-15)

 1000 FORMAT(1X,'Initial matrix with augmented identity matrices')
 1001 FORMAT(1X,'Reduced matrix with S, T, M, and N submatrices')
 1002 FORMAT(1X,'A1,2 generalized inverse')
 1003 FORMAT(1X,'Check A1,2 against the following conditions:')
 1004 FORMAT(1X,'          (A A1,2 A) = A')
 1005 FORMAT(1X,'       (A1,2 A A1,2) = A1,2')
 1006 FORMAT(1X,'A1,2,3 generalized inverse')
 1007 FORMAT(1X,'Check A1,2,3 against  (A A1,2,3)* = (A A1,2,3)')
```

```
1008 FORMAT(1X,'A1,2,4 generalized inverse')
1009 FORMAT(1X,'Check A1,2,4 against  (A1,2,4 A)* = (A1,2,4 A)')
1010 FORMAT(1X,'A1,2,3,4 generalized inverse')
1011 FORMAT(1X,'Check A1,2,3,4 with (A A1,2,3,4)* = (A A1,2,3,4)')
1012 FORMAT(1X,'Check A1,2,3,4 with (A1,2,3,4 A)* = (A1,2,3,4 A)')
1013 FORMAT(1X)
1014 FORMAT(12(F5.2,1X))
1015 FORMAT(12(F6.2,1X))
1016 FORMAT(12(F6.3,1X))
1017 FORMAT(12(F7.4,1X))

     IF (OPTION.NE.1) GO TO 2000

          IF (HEADER.EQ.1)  WRITE(9,1000)
          IF (HEADER.EQ.2)  WRITE(9,1001)
          IF (HEADER.EQ.3)  WRITE(9,1002)
          IF (HEADER.EQ.4)  WRITE(9,1003)
          IF (HEADER.EQ.4)  WRITE(9,1013)
          IF (HEADER.EQ.4)  WRITE(9,1004)
          IF (HEADER.EQ.5)  WRITE(9,1005)
          IF (HEADER.EQ.6)  WRITE(9,1006)
          IF (HEADER.EQ.7)  WRITE(9,1007)
          IF (HEADER.EQ.8)  WRITE(9,1008)
          IF (HEADER.EQ.9)  WRITE(9,1009)
          IF (HEADER.EQ.10) WRITE(9,1010)
          IF (HEADER.EQ.11) WRITE(9,1011)
          IF (HEADER.EQ.12) WRITE(9,1012)

       GO TO 2010

2000      IF (HEADER.EQ.1)  WRITE(10,1000)
          IF (HEADER.EQ.2)  WRITE(10,1001)
          IF (HEADER.EQ.3)  WRITE(10,1002)
          IF (HEADER.EQ.4)  WRITE(10,1003)
          IF (HEADER.EQ.4)  WRITE(10,1013)
          IF (HEADER.EQ.4)  WRITE(10,1004)
          IF (HEADER.EQ.5)  WRITE(10,1005)
          IF (HEADER.EQ.6)  WRITE(10,1006)
          IF (HEADER.EQ.7)  WRITE(10,1007)
          IF (HEADER.EQ.8)  WRITE(10,1008)
          IF (HEADER.EQ.9)  WRITE(10,1009)
          IF (HEADER.EQ.10) WRITE(10,1010)
```

```
                  IF (HEADER.EQ.11) WRITE(10,1011)
                  IF (HEADER.EQ.12) WRITE(10,1012)

       2010  CONTINUE

             IF (OPTION.EQ.1) THEN
                  WRITE(9,1013)
                ELSE
                  WRITE(10,1013)
             END IF

             DO 2060 I=STR,ER
                  DO 2020 J=STC,EC
                       IF (ABS(A(I,J)).LT.ZERO) A(I,J) = 0.0
       2020       CONTINUE

                  IF (OPTION.NE.1) GO TO 2030
                  IF (FORMATT.EQ.1) WRITE (9,1014) (A(I,J),J=STC,EC)
                  IF (FORMATT.EQ.2) WRITE (9,1015) (A(I,J),J=STC,EC)
                  IF (FORMATT.EQ.3) WRITE (9,1016) (A(I,J),J=STC,EC)
                  IF (FORMATT.EQ.4) WRITE (9,1017) (A(I,J),J=STC,EC)

                GO TO 2040

       2030       CONTINUE
                  IF (FORMATT.EQ.1) WRITE(10,1014) (A(I,J),J=STC,EC)
                  IF (FORMATT.EQ.2) WRITE(10,1015) (A(I,J),J=STC,EC)
                  IF (FORMATT.EQ.3) WRITE(10,1016) (A(I,J),J=STC,EC)
                  IF (FORMATT.EQ.4) WRITE(10,1017) (A(I,J),J=STC,EC)

       2040     CONTINUE

       2060  CONTINUE


             RETURN
             END

       C************************************************************
       C************************************************************
```

```
C*****************************************************************
C                                                                *
C       DATE: 11/11/85                                           *
C       VERSION: 1.0                                             *
C                                                                *
C       NAME: MULT                                               *
C       DESCRIPTION: MULTIPLIES ANY TWO MATRICES OR SUBMATRICES  *
C                    AND PLACES THE RESULT IN AN OPTIONAL THIRD  *
C                    MATRIX OR SUBMATRIX.                        *
C       PASSED VARIABLES: XR, XC, D1, YR, YC, D2, CONFORMITY, M1 *
C                         M2, AND M3                             *
C       RETURNS: NONE                                            *
C       FILES READ: NONE                                         *
C       SUBROUTINES CALLED: NONE                                 *
C       CALLING SUBROUTINE: GINVERSE                             *
C                                                                *
C*****************************************************************

        SUBROUTINE MULT(XR,XC,D1,YR,YC,D2,CONFORMITY,M1,M2,M3)

        DOUBLE PRECISION  M1,M2,M3
        DIMENSION         M1(1:10,1:10)
        DIMENSION         M2(1:10,1:10)
        DIMENSION         M3(1:10,1:10)
        INTEGER           XR,XC,D1,YR,YC,D2,CONFORMITY

        DOUBLE PRECISION  ROW
        DIMENSION         ROW(1:10)
        INTEGER           DXR
        INTEGER           DXC
        INTEGER           DYR
        INTEGER           DYC
        INTEGER           IXR
        INTEGER           JYC
        INTEGER           KXC
        INTEGER           KYR
        INTEGER           I
        INTEGER           J
        INTEGER           K

        DXR = XR-1
        DXC = XC-1
        DYR = YR-1
```

```
      DYC = YC-1

      DO 1040 I=1,D1
         IXR = I+DXR
            DO 1020 J=1,D2
               JYC = J+DYC
               ROW(J)=0.0
               DO 1010 K=1,CONFORMITY
                  KXC = DXC+K
                  KYR = DYR+K
                  ROW(J) = ROW(J)+(M1(IXR,KXC)*M2(KYR JYC))
1010           CONTINUE
1020        CONTINUE
            DO 1030 J=1,D2
               M3(I,J) = ROW(J)
1030        CONTINUE
1040  CONTINUE

      RETURN
      END


C*************************************************************
C*************************************************************
```

```
C************************************************************
C                                                           *
C       DATE: 11/11/85                                      *
C       VERSION: 1.0                                        *
C                                                           *
C       NAME: GSROW                                         *
C       DESCRIPTION: ORTHOGONALIZES THE ROWS OF THE SUBMATRICES *
C                    T AND M                                *
C       PASSED VARIABLES: RANK, R, C, ASIZE, A              *
C       RETURNS: NONE                                       *
C       FILES READ: NONE                                    *
C       SUBROUTINES CALLED: NONE                            *
C       CALLING SUBROUTINE: GINVERSE                        *
C                                                           *
C************************************************************

        SUBROUTINE GSROW(RANK,R,C,ASIZE,A)

        DOUBLE PRECISION  A
        DIMENSION         A(1:10,1:10)
        INTEGER           RANK,R,C,ASIZE
        DOUBLE PRECISION  DOT1
        DOUBLE PRECISION  DOT2
        INTEGER           I
        INTEGER           J
        INTEGER           K


        PARAMETER(ZERO=5.E-15)


        DO 1040 I=R,RANK+1,-1
           DOT2=0.0
           DO 1000 K=C+1,ASIZE
              DOT2 = DOT2+(A(I,K)**2)
1000       CONTINUE
           DO 1030 J=I-1,1,-1
              DOT1=0.0
              DO 1010 K=C+1,ASIZE
                DOT1 = DOT1+(A(I,K)*A(J,K))
1010          CONTINUE
```

```
                IF ((ABS(DOT1).GT.ZERO).AND.(ABS(DOT2).GT.ZERO)) THEN
                  DOT1 = DOT1/DOT2
                  DO 1020 K=C+1,ASIZE
                    A(J,K) = A(J,K)-(DOT1*A(I,K))
1020              CONTINUE
                ENDIF
1030        CONTINUE
1040  CONTINUE


      RETURN
      END

C*************************************************************
C*************************************************************
```

```
C***********************************************************
C                                                          *
C     DATE: 11/11/85                                       *
C     VERSION: 1.0                                         *
C                                                          *
C     NAME: GSCOLUMN                                       *
C     DESCRIPTION: ORTHOGONALIZE THE COLUMNS OF THE SUBMATRICES *
C                  S AND N                                 *
C     PASSED VARIABLES: RANK, R, C, ASIZE, A               *
C     RETURNS: NONE                                        *
C     FILES READ: NONE                                     *
C     SUBROUTINES CALLED: NONE                             *
C     CALLING SUBROUTINE: NONE                             *
C                                                          *
C***********************************************************

      SUBROUTINE GSCOL(RANK,R,C,ASIZE,A)

      DOUBLE PRECISION  A
      DIMENSION         A(1:10,1:10)
      INTEGER           RANK,R,C,ASIZE

      DOUBLE PRECISION  DOT1
      DOUBLE PRECISION  DOT2
      INTEGER           I
      INTEGER           J
      INTEGER           K


      PARAMETER(ZERO=5.E-15)


      DO 1040 I=C,RANK+1,-1
         DOT2=0.0
         DO 1000 K=R+1,ASIZE
            DOT2 = DOT2+(A(K,I)**2)
1000     CONTINUE
         DO 1030 J=I-1,1,-1
            DOT1=0.0
            DO 1010 K=R+1,ASIZE
              DOT1 = DOT1+(A(K,I)*A(K,J))
1010        CONTINUE
```

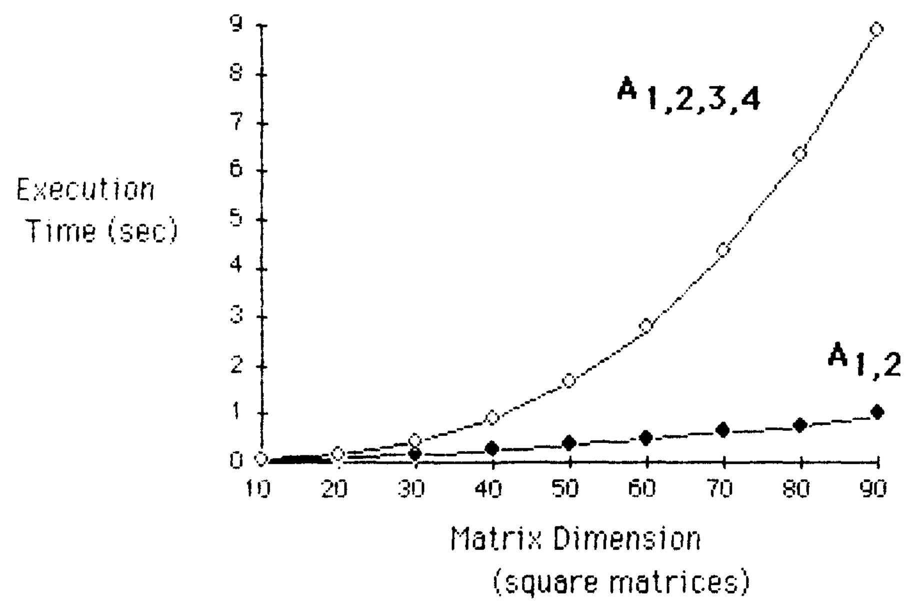
```
                IF ((ABS(DOT1).GT.ZERO).AND.(ABS(DOT2).GT.ZERO)) THEN
                   DOT1 = DOT1/DOT2
                   DO 1020 K=R+1,ASIZE
                      A(K,J) = A(K,J)-(DOT1*A(K,I))
1020               CONTINUE
                ENDIF
1030         CONTINUE
1040    CONTINUE


        RETURN
        END


C***********************************************************
C***********************************************************
```

Appendix C: Graph of an $A_{1,2}$ Versus an $A_{1,2,3,4}$

Starting with the $A_{1,2}$ and continuing through the $A_{1,2,3,4}$, each successive inverse satisfies either more or a different combination of Penrose's equations. This relationship between each generalized inverse is also reflected in the computer time needed to compute an inverse. To meet the additional constraints, successive inverses require the same or more computer time. This relationship was originally mentioned in chapter 3 and its practical aspects mentioned again in chapter 4 where an $A_{1,2}$ was used in place of an $A_{1,2,3,4}$ generalized inverse. The following graph emphasizes this realtionship by highlighting the differences in computing time between an $A_{1,2}$ and an $A_{1,2,3,4}$ over a range of square matrices.



Bibliography

1. Althoen, Steven C. and Robert J. Bumcrot. Matrix Methods in Finite Mathematics. New York: W.W.Norton, 1976.

2. Amsbury, Wayne. Data Structure: From Arrays to Priority Queues. Belmont California: Wadswoth Publishing Company, 1985.

3. Ben-Israel, Adi and Thomas N. E. Greville. Generalized Inverses Theory and Application. New York: Wiley-Interscience, 1974.

4. Boullion, Thomas L and Patrick L. Odell. Generalized Inverse Matrices. New York: Wiley-Interscience, 1971.

5. Cambell S. L. and C .D. Meyers. Generalized Inveres of Linear Transformations. London: Pitman, 1979.

6. Colletti, Bruce. Solutions of Nonlinear Matrix Equations. MS Thesis, GOS 85D-1. School of Engineering, Air Force Institute of Technology. December 1985.

7. Deif, Assem S. Advanced Marix Theory for Scientist and Engineers. London: Abacus Press, 1982.

8. Doma, A. M. E. Generalized Inverses of Matrices and its Applications. MS Thesis, GCS 83D-1. School of Engineering, Air Force Institute of Technology. December 1983 (AD-A138 274).

9. Goodman S.E. and S.T. Hedetiemi. Introduction to the Design and Analysis of Algorithms. New York: McGraw-Hill, 1977.

10. Jones, John, Jr. Lecture Notes from MA 7.99. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, May 85.

11. Jones, John, Jr. "Matrix Equations Over Rings," Mathematics and Computers in Simulation, XXV: 489-492 (December 1983).

12. Jones, John, Jr. "Solutions of Matrix Equations Arising in Discrete Simulation of Systems Containing Parameters," Advances in Computer Methods for Partial Differential Equations, edited by R Vichnevetsky and R. S. Stepleman. New Brunswick, New Jersey: IMACS, 1983.

13. Keller-McNulty, S. and W. J. Kennedy. "Error-Free Computation of a Reflective Generalized Inverse," Linear Algebra and Its Application, 67: 157-167 (1985).

14. Krishnamurthy, E. V. Fast Parallel Exact Computation of the Generalized Inverse and Rank of a Matrix. Contract AFOSR-77-3271. Maryland University, College Park MD, July 81 (AD-A105 566).

15. Lovass-Nagy, Victor, Richard J. Miller, and David L. Powers. "An Introduction to the Application of the Simplest Matrix-Generalized Inverse in Systems Science," IEEE Transactions on Circuits and Systems, 25: 766-771 (September 1978).

16. Meyers, Glenford J. The Art of Software Testing. New York: John Wiley and Sons, 1979.

17. Miller, Webb. The Engineering of Numerical Software. Englewood Cliffs, New Jersey: Prentice-Hall, 1984.

18. Morris, Gerald L. and Patrick L Odell. "Common Solutions for n Matrix Equations With Applications," Journal of the Association for Computing Machinery, 15: 272-274 (April 1986).

19. Nashed, M. Generalized Inverses and Applications. New York: Academic, 1979.

20. Noble, Ben. Applied Linear Algebra. Englewood Cliffs, New Jersey: Prentice-Hall, 1969.

22. Noble, Ben and James W. Daniel (Second Edition). Applied Linear Algebra. Englewood Cliffs, New Jersey: Prentice-Hall, 1977.

23. Plemmons, R. J. and M. Neumann. Generalized Inverse-Positivity and Splitting of M-Matrices. Contract DAAG29-77-G-0166. Tennessee University, Knoxville TN, March 24, 1977 (AD-A071 849).

23. Rao C. Radhakrishna and Sujit Kumar Mitra. Generalized Inverse of Matrices and its Application. New York: John Wiley and Sons, 1971.

24. Rice, John R. Matrix Computations and Mathematical Software. New York: McGraw-Hill Book Company, 1981.

25. Sahni, Sartaj, and Ellis Horowitz. Fundamentals of Computer Algorithms. Rockville,Maryland: Computer Science Press, 1984.

26. Strang, Gilbert. Linear Algebra and its Applications. London: Academic Press, 1976.

27. Ward, Cheney and David King. Numerical Mathematics and Computing. Belmont, California: Wadsworth, 1985.

Vita

Captain Craig F. Murray was born on 3 January 1958 in Scranton Pennsylvania. He graduated from high school in 1976 and attended the United States Air Force Academy earning a Bachelor of Science degree in Computer Science in May 1980. Upon graduation he was assigned to the 7th Missile Warning Squadron, Beale AFB, California, as a computer systems analyst. He received a Masters of Science in Systems Management from the University of Southern California in June 1983. In May 1984, Captain Murray entered the School of Engineering, Air Force Institute of Technology.

Permanent Address: Box 49
Waverly, Pa 18471

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

AD-A163 840

REPORT DOCUMENTATION PAGE

1a. [illegible]PORT SECURITY CLASSIFICATION [illegible]classified	1b. RESTRICTIVE MARKINGS
2a. [illegible]URITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distibution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GOR/MA/85D-2	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering	6b. OFFICE SYMBOL *(If applicable)* AFIT/ENG
7a. NAME OF MONITORING ORGANIZATION	
6c. ADDRESS *(City, State and ZIP Code)* Air Force Institute of Technology Wright-Patterson AFB OH 45433	7b. ADDRESS *(City, State and ZIP Code)*
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL *(If applicable)*
9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS *(City, State and ZIP Code)*	10. SOURCE OF FUNDING NOS.
11. TITLE *(Include Security Classification)* See Box 19 (unclassified)	

10. SOURCE OF FUNDING NOS.			
PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

12. [illegible]SONAL AUTHOR(S)
[illegible]aig F. Murray, Captain, USAF

13a. [illegible]PE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Yr., Mo., Day)*	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	1985 December	131

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB. GR.	Linear Algebra, Generalized Inverses, Lyapunov Equation
12	01		

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

A New Method for Computing the Generalized Inverses of a Matrix and its Application to the Lyapunov Matrix Equation

Thesis Advisor: Dr. J. Jones Jr.

Approved for public release: IAW AFR 190-1/.
[illegible]
LYNN E. WOLAVER 16 JAN 86
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

20. [illegible]STRIBUTION/AVAILABILITY OF ABSTRACT [illegible]ASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER *(Include Area Code)*	22c. OFFICE SYMBOL
Dr. John Jones Jr.	513-255-3098	AFIT/ENC

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

Abstract

This work examines a new method for computing four generalized inverses of a matrix. This method, the ST method, is based on the careful selection of a sequence of matrix multiplications and partitionings which provide a new foundation for computing four generalized inverses. Central to this approach is the partitioning of the two submatrices, R and C, where the product of their submatrices will give the generalized inverses of interest. Thus using this new representation, the generalized inverses of a matrix can be computed in a simple and direct manner.

In this work, four generalized inverses are derived and computed in a systematic manner from this representation. These results are strongly tied to the solution of the matrix equation $Ax=b$ where the general solution is given in terms of this new representation and computational technique. The computational technique is presented with an example and also as an algorithm. Included with the algorithm is an analysis of its computer implementation. The use of one generalized inverse is used to find the solution of a Lyapunov matrix equation.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE