GIVENS TRANSFORMATIONS FOR LEAST SQUARES

BY

HSIAO-LAN WANG LOH

Bachelor of Arts

Fu-Jen Catholic University

Taiwan, Republic of China

1977

Submitted to the faculty of the
Graduate College of the
Oklahoma State University
in partial fullfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1983

GIVENS TRANSFORMATIONS FOR

LEAST SQUARES

Thesis Approved:

J P Chandler
Thesis Adviser

W. E. Hedrick

A. A. Thoreson

Norman N Durham
Dean of the Graduate College

PREFACE

This study implements the orthogonal decomposition method based on Givens transformations to solve linear least squares problems. A comparison has been made with the methods based on Householder transformations and the modified Gram-Schmidt algorithm with respect to storage requirements, time requirements, and accuracy.

I would like to thank Dr. G. E. Hedrick and Dr. S. A. Thoreson for their suggestions, and Dr. D. W. Grace for substituting during my oral examination. A special thanks goes to my major advisor, Dr. John P. Chandler, whose assistance and guidance were invaluable for this thesis and for my studies at Oklahoma State University.

The deepest appreciation is extended to my parents and my father-in-law for their love and confidence. My final thanks goes to my husband, Hsiaoli, whose encouragement and considerateness played an important role in completion of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

This thesis will implement the orthogonal decomposition method based on Givens transformations (Givens rotations) in a portable FORTRAN subroutine, named GIVEN, to solve linear least squares problems. Then it will compare this method with the methods based on the modified Gram-Schmidt algorithm (modified Gram-Schmidt projections) and Householder transformations (Householder reflections) with respect to speed, accuracy, and storage requirements.

Forming and solving the normal equations numerically (see Chapter II) is a common and the cheapest way to solve linear least squares problems, but the result is often quite unsatisfactory. The main reason is that serious loss of accuracy can occur when the crossproduct matrix A^TA is formed [22, 26]. Orthogonal decomposition (QR decomposition) methods based on the modified Gram-Schmidt algorithm [1, 2, 3, 4, 10, 22, 33] or Householder transformations [5, 7, 14, 16, 19, 22, 33] are generally the most accurate approaches to solve linear least squares problems. However, they require storage for the whole design matrix A in main memory; thus the size of problems for which they can be used are restricted. Moreover, they are not suited to updating the solution by adding a new row to the design matrix or to delete a row from the design matrix when the original design matrix has already been triangularized.

An orthogonal decomposition method based on Givens transformations is nearly as accurate as any other orthogonal decomposition method, but has two major advantages [12, 13]. The first is that the design matrix can be processed one row at a time. Secondly, zeros already present in the design matrix are readily exploited to reduce arithmetic.

Givens transformations have been used in the least squares problems by Fowlkes [11], Chambers [8], and Gentleman [12, 13]. However, Gentleman inserts a diagonal scaling matrix D between the factors of the Cholesky decomposition (matrix square root). This new version of Givens transformations eliminates all square roots and halves the number of multiplications required. Furthermore, it can be used to solve weighted least squares problems, and to remove a row from the design matrix by adding it again with the negative of its previous weight [15]. However, any method of removing rows is potentially unstable. Meanwhile, weighted problems are not necessary for accuracy tests. Therefore, deletion or weighted problems will not be considered in this study.

Chapter II will discuss the theoretical background of solving linear least squares problems including normal equations, modified Gram-Schmidt algorithm, Householder transformations, and Givens transformations.

Chapter III will present a description of the test problems. There are three sets of test problems including integer matrices, polynomials, and ill-conditioned problems. Integer matrices are chosen for ensuring that all error is generated during computation since integer matrices can be expressed in the computer exactly. Furthermore, ill-conditioned problems are chosen in order to prove that orthogonal decomposition methods are stable.

In Chapter IV, a description will be made for the programs to be tested, which are GIVEN, ORTHL [1], BLSQS [4], and LLSQF [20]. GIVEN is converted from the ALGOL procedures in Gentleman [13]. Although weighted problems will not be included in this study, GIVEN still preserves the feature that it can be used on weighted problems. For unweighted problems, the user simply sets the variable WEIGHT the value 1 for each row (each row has its own weight) of the design matrix. Detailed program functions and users instructions of GIVEN will also be shown respectively in this chapter. ORTHL and BLSQS are the implementations of medified Gram-Schmidt, and LLSQF is the implementation of Householder transformations. The main purpose of the above mentioned programs is to compare accuracy among orthogonal decomposition methods.

Chapter V will present a discussion of these three orthogonal decomposition methods with respect to storage requirements, time requirements, and error bounds.

Test results will be listed in Chapter VI, and will be followed by a discussion of these results. An average number of significant digits lost will be computed for each program on each problem. Chapter VII will give conclusions of this thesis, and will make suggestions for further research. Finally, program listings will be collected in Appendices.

CHAPTER II

THEORETICAL BACKGROUND

The linear least squares problem arises in a variety of areas and in a variety of contexts. In particular, it is intimately connected with the approximations of data and with the parts of statistics which are concerned with the normal distribution. Before discussing the theoretical background of methods used to solve linear least squares problems, it is necessary to specify what a linear least squares problem is. The model linear least squares problems is to compute a vector of regression coefficients $\vec{x}$ so as to minimize the sum of the squares of the components of the residual vector $\vec{r}$ which is defined by

$$\vec{r}_{m\times 1} = \vec{b}_{m\times 1} - A_{m\times n} \cdot \vec{x}_{n\times 1} . \qquad (2\text{-}1)$$

A is a given rectangular matrix with rank r ($r \leq n$); b is a given vector of observations; and m is greater than n ($m >> n$ usually). This problem is usually denoted by

$$|\vec{r}|_2 = |\vec{b} - A\vec{x}|_2 = \min. \qquad (2\text{-}2)$$

where $|\ldots|_2$ indicates the euclidean norm. The problem is said to be linear because $\vec{r}$ depends on $\vec{x}$ linearly. If $r<n$ then there is no unique solution [5]. Under these conditions, it is required simultaneously that $|\vec{x}|_2$ to be a minimum related to the Moore-Penrose generalized inverse matrix. This circumstance is a very natural one for many

statistical and numerical problems; however, the problems which will be tested in this study are full ranked (i.e. r=n) since the program ORTHL (will be discussed in Chapter IV) requires that the matrix A has independent columns.

There are three general approaches for computing $\vec{x}$ [8]:

a. Solve the normal equations

$$A^T A\vec{x} = A^T\vec{b} \ , \tag{2-3}$$

by forming the Cholesky decomposition of A^TA.

b. Form an orthogonal decomposition of A.

c. Form a singular value decomposition of A.

Since one of the main purposes of this study is to compare the orthogonal decomposition methods, singular value decomposition is not covered in this study.

Normal Equations

Let $\vec{x}$ be a solution of least squares problem of minimizing (2-1). Since

$$\vec{r} = \vec{b} - A\vec{x} = \vec{b} - \vec{b}_1 = \vec{b}_2 \ , \tag{2-4}$$

but $\vec{b}_2$ belongs to the orthogonal complement of R(A). Hence

$$\vec{0} = A^T\vec{b}_2 = A^T\vec{r} = A^T(\vec{b} - A\vec{x}) \ . \tag{2-5}$$

Therefore the solution of (2-3) minimizes the least squares problem (2-1). Unfortunately, the matrix A^TA is frequently ill-conditioned [25] and influenced greatly by roundoff errors. The following example of

Golub [14] illustrates this well. Suppose that

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ a & 0 & 0 & 0 & 0 \\ 0 & a & 0 & 0 & 0 \\ 0 & 0 & a & 0 & 0 \\ 0 & 0 & 0 & a & 0 \\ 0 & 0 & 0 & 0 & a \end{bmatrix}, \tag{2-6}$$

then

$$A^TA = \begin{bmatrix} 1+a^2 & 1 & 1 & 1 & 1 \\ 1 & 1+a^2 & 1 & 1 & 1 \\ 1 & 1 & 1+a^2 & 1 & 1 \\ 1 & 1 & 1 & 1+a^2 & 1 \\ 1 & 1 & 1 & 1 & 1+a^2 \end{bmatrix}. \tag{2-7}$$

Clearly for $a \neq 0$, the rank of A^TA is five, and the eigenvalue of A^TA are $5+a^2$, a^2, a^2, a^2, a^2.

Assume that the elements of A^TA are computed using double precision arithmetic and then rounded to single precision accuracy. Now let ε be the largest number on the computer such that $fl(1.0+\varepsilon)=1.0$ where $fl(...)$ indicates floating point computation. Now if $a<\sqrt{\varepsilon}/2$, then

$$fl(A^TA) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{2-8}$$

The rank of the computed representation of (2-8) will be one. Consequently, no matter how accurate the linear equation solver, it will be impossible to solve the normal equations (2-3). On the other hand, forming the normal equations can square the condition number of the problem [1]. If the condition number is denoted by cond(A), then

$$\text{cond}(A^TA) \leq \text{cond}^2(A). \tag{2-9}$$

This fact shows that in general using t-digit binary arithmetic, it may be impossible to obtain even an approximate solution to (2-3) unless $\text{cond}(A) < 2^{-t/2}$. Longley [23] has given examples in which the solution of the normal equations obtains almost no digits of accuracy in least squares problems.

Orthogonal decomposition method is a better way to solve linear least squares problems. This approach is also called QR decomposition since it finds Q and R such that

$$A = QR, \tag{2-10}$$

where Q is an m×n orthogonal matrix and R is an n×n upper triangular matrix. Indeed, use of an orthogonalization process on A for obtaining a least squares solution is known in the literature [18]. The linear least squares problem becomes $QR\vec{x}=\vec{b}$. If this equation is premultipied by Q^T, then

$$Q^TQR\vec{x} = Q^T\vec{b}. \tag{2-11}$$

Since $Q^TQ=I$,

$$R\vec{x} = Q^T\vec{b} = \vec{\theta}. \tag{2-12}$$

Thus, (2-12) can be solved easily by using successive back substitutions. Notably, $R=Q^TA$ and the right hand side $Q^T\vec{b}$ are obtained by applying the same operation Q^T to A and $\vec{b}$ respectively. Further,

$$\text{cond}(Q^TA) = \text{cond}\ (A). \tag{2-13}$$

The orthogonal decomposition may be carried out via Householder transformations, the modified Gram-Schmidt algorithm, or Givens transformations. These will be discussed in the following sections.

Householder Transformations

Householder transformations are also known as elementary reflectors and as elementary Hermitian matrices [27]. Since Householder was the first to use elementary reflectors in a systematic way to introduce zeros into a matrix [19], the first name is more common than the last two names. Golub [14] was the first to work out the details and in conjunction with Businger [7] publish an algorithm.

Let $A=A^{(1)}$, and let $A^{(2)}$, $A^{(3)}$, ... , $A^{(n+1)}$ be defined as follows:

$$A^{(k+1)} = P^{(k)}A^{(k)} \quad (k=1,\ 2,\ \ldots\ ,\ n), \tag{2-14}$$

where $P^{(k)}$ is a symmetric, orthogonal matrix of the form

$$P^{(k)} = I - \beta_k\vec{u}^{(k)}\vec{u}^{(k)T}, \tag{2-15}$$

The elements of $P^{(k)}$ are derived so that $a_{i,k}^{(k+1)} = 0$ for $i=k+1, \ldots, m$. In other words, Householder transformations are used to zero out the subdiagonal part of each column. Moreover, $P^{(k)}$ is generated as

follows:

$$\sigma_k = \left(\sum_{i=1}^{m} (a_{i,k}^{(k)})^2\right)^{1/2}, \tag{2-16}$$

$$\beta_k = \left[\sigma_k(\sigma_k + |a_{k,k}^{(k)}|)\right]^{-1}, \tag{2-17}$$

$$\vec{u}_i^{(k)} = 0, \quad \text{for } i<k, \tag{2-18}$$

$$\vec{u}_i^{(k)} = \text{sgn}(a_{i,k}^{(k)})\ (\sigma_k + |a_{i,k}^{(k)}|), \quad \text{for } i=k, \tag{2-19}$$

$$\vec{u}_i^{(k)} = a_{i,k}^{(k)}, \quad \text{for } i>k. \tag{2-20}$$

Since $P^{(k)}$ is not computed explicitly, it is clear that

$$P^{(k)}A^{(k)} = (I - \beta_k \vec{u}^{(k)}\vec{u}^{(k)T})\ A^{(k)} \tag{2-21}$$

$$= A^{(k)} - \vec{u}^{(k)}(\beta_k \vec{u}^{(k)T} A^{(k)}). \tag{2-22}$$

Therefore $A^{(k+1)}$ and $\vec{b}^{(k+1)}$ are obtained by

$$A^{(k+1)} = A^{(k)} - \vec{u}^{(k)}(\beta_k \vec{u}^{(k)T} A^{(k)}) \tag{2-23}$$

and

$$\vec{b}^{(k+1)} = \vec{b}^{(k)} - \vec{u}^{(k)}(\beta_k \vec{u}^{(k)T} \vec{b}^{(k)}), \tag{2-24}$$

respectively. In computing (2-23) and (2-24), one can take the advantage that the first (k-1) components of $\vec{u}^{(k)}$ are zeroes. After k^{th} transformation, $A^{(k+1)}$ becomes as follows:

$$A^{(k+1)} = \begin{bmatrix} \underset{\sim}{R}^{(k+1)} & \\ \hline 0 & \end{bmatrix} \qquad (2\text{-}25)$$

where $\underset{\sim}{R}^{(k+1)}$ is a k×k upper triangular matrix which is unchanged by subsequent transformations.

Modified Gram-Schmidt Algorithm

Gram-Schmidt orthogonalization is another method for decomposing a matrix into the product of a matrix with orthogonal columns and a triangular matrix as (2-10). The classical formulas expressing $\vec{q}_j$ in terms of $\vec{a}_j$ and the previously determined vectors $\vec{q}_1, \ldots, \vec{q}_{j-1}$ appear as follows:

$$\vec{q}_1 = \vec{a}_1 , \qquad (2\text{-}26)$$

$$\vec{q}_j = \vec{a}_j - \sum_{i=1}^{j-1} r_{ij}\vec{q}_i \qquad (j = 2, \ldots, n), \qquad (2\text{-}27)$$

where

$$r_{ij} = (\vec{a}_j^T \vec{q}_i) \; / \; (\vec{q}_i^T \vec{q}_i^T) . \qquad (2\text{-}28)$$

To convert to matrix notation, define A to be the matrix with columns of $\vec{a}_j$, Q to be the matrix with columns of $\vec{q}_j$, and R to be the upper triangular matrix with unit diagonal elements with the strictly upper triangular elements given by (2-28). Then, (2-26) and (2-27) can be written as A=QR. The experimental evidence in Rice [26] indicated that equations (2-26) to (2-28) have significantly less numerical stability

than the modified Gram-Schmidt method given below. Rice was the first person to point out and explain the superior numerical properties of the modified Gram-Schmidt algorithm. Then Bjorck [2] gave detailed error analysis. This modified Gram-Schmidt algorithm was established by Rice [26] and it is described as follows:

$$\vec{a}_j^{(1)} = \vec{a}_j \qquad j = 1, \ldots, n \tag{2-29}$$

For i=1 to n

$$\vec{q}_i = \vec{a}_i^{(i)} \tag{2-30}$$

$$d_i^2 = \vec{q}_i^T \vec{q}_i \tag{2-31}$$

For j=i+1 to n

$$r_{ij} = \vec{a}_j^{(i)T} \vec{q}_i \; / \; d_i^2 \tag{2-32}$$

$$\vec{a}_j^{(i+1)} = \vec{a}_j^{(i)} - r_{ij}\vec{q}_i \; . \tag{2-33}$$

To use modified Gram-Schmidt in the solution of linear least squares problems, one can form the augmented matrix

$$\tilde{A} = [A \mid \vec{b}], \tag{2-34}$$

and apply modified Gram-Schmidt algorithm to the m×(n+1) matrix $\tilde{A}$ to obtain

$$\tilde{A} = \tilde{Q}\tilde{R}, \tag{2-35}$$

where the matrix $\tilde{R}$ is also upper triangular with unit diagonal elements. The strictly upper triangular elements of R are given by (2-32). The vectors $\vec{q}_i$ given by (2-30) constitute the column vectors of the m×(n+1) matrix $\tilde{Q}$. The (n+1)×(n+1) diagonal matrix $\tilde{D}$ with diagonal elements $\vec{\tilde{d}}_i$,

$i = 1, \ldots, n+1$, is obtained by (2-31). Futher, the amount of computations and storage required are not increased by this modification. Wampler [29] has found that the modified Gram-Schmidt and Householder programs have essentially equivalent accuracy. However, Jordan [21] obtained experimental results that the modified Gram-Schmidt algorithm performs a little more accurately than Householder transformations do.

Givens Transformations

One way to view the method based on Givens transformations is as a numerically stable way to update the Cholesky decomposition of the crossproduct matrix to add one more row [12]. A Givens transformation rotates two row vectors

$$0 \ldots 0 \quad r_i \quad r_{i+1} \quad \ldots\ldots \quad r_k \ldots$$

$$0 \ldots 0 \quad x_i \quad x_{i+1} \quad \ldots\ldots \quad x_k \ldots ,$$

and replaces them with two new vectors

$$0 \ldots 0 \quad r'_i \quad r'_{i+1} \quad \ldots\ldots \quad r'_k \ldots$$

$$0 \ldots 0 \quad 0 \quad x'_{i+1} \quad \ldots\ldots \quad x'_k \ldots ,$$

where

$$r'_k = cr_k + sx_k, \tag{2-36}$$

$$x'_k = -sr_k + cx_k, \tag{2-37}$$

$$c^2 + s^2 = 1. \tag{2-38}$$

The requirement that x_i is transformed to zero indicates that

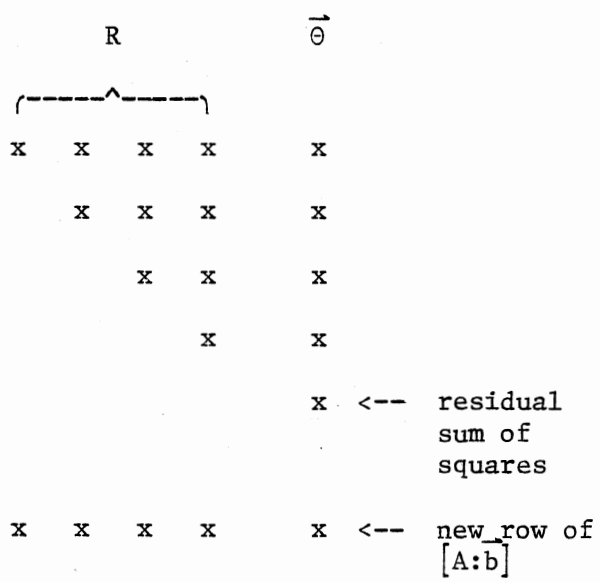
$$r_i' = (r_i^2+x_i^2)^{1/2} , \tag{2-39}$$

$$c = r_i / (r_i^2+x_i^2)^{1/2} = r_i / r_i' , \tag{2-40}$$

$$s = x_i / (r_i^2+x_i^2)^{1/2} = x_i / r_i' . \tag{2-41}$$

The transformation obviously leaves unchanged zeros appearing in corresponding elements of both vectors.

When a new row has been added in R, as shown in the following diagram,



R can be retriangularized by rotating the new row successively with the first, second, third, etc. rows of R until the entire new row of A has been transformed to zero. This process needs m×n square roots totally for solving a least squares problem. However, square roots are avoided in Gentleman [12, 13]. The trick is to find not R itself, but rather a diagonal matrix D and a unit upper triangular matrix $\bar{R}$ such that

$$R = \sqrt{D}\,\bar{R} \ . \tag{2-42}$$

Rotation is made on a row of the product $\sqrt{D}\,\bar{R}$ with a scaled row of A as follows:

$$0 \ \ldots \ 0 \ \sqrt{d} \ \ \ldots \ \sqrt{d}\,\bar{r}_k \ \ldots$$

$$0 \ \ldots \ 0 \ \sqrt{\delta}\,x_i \ \ldots \ \sqrt{\delta}\,x_k \ \ldots \ .$$

From (2-36) to (2-41), the transformed rows can be written as follows:

$$0 \ \ldots \ 0 \ \sqrt{d'} \ \ \ldots \ \sqrt{d'}\,\bar{r}'_k \ \ldots$$

$$0 \ \ldots \ 0 \ \ 0 \ \ \ldots \ \sqrt{\delta'}\,x'_k \ \ldots \ ,$$

where

$$d' = d + \delta x_i^2 \ , \tag{2-43}$$

$$\delta' = d\delta / \ (d+\delta x_i^2) = d\delta \ / \ d' \ , \tag{2-44}$$

$$\bar{c} \ = d \ / \ (d+\delta x_i^2) = d \ / \ d' \ , \tag{2-45}$$

$$\bar{s} \ = \delta x_i \ / \ (d+\delta x_i^2) = \delta x_i \ / \ d' \ , \tag{2-46}$$

$$x'_k = x_k - x_i\,\bar{r}_k \ , \tag{2-47}$$

$$\bar{r}'_k = \bar{c}\,\bar{r}_k + \bar{s}\,x_k \ . \tag{2-48}$$

In other words, the transformed rows can be expressed as a row of a new $\sqrt{D}\,\bar{R}$ and a new scaled row of A. Formulas (2-43) to (2-48) not only can avoid the square roots of (2-36) to (2-41), but also reduce the number of multiplications required [12]; the retriangularization, thus, can be

done faster.

Furthermore, Gentleman points out that the formula (2-48) can be written in a different way to save another multiplication. It is given by

$$\bar{r}_k' = \bar{r}_k + \bar{s}\, x_k' \ . \tag{2-49}$$

It is easy to verified as the following equations that (2-48) and (2-49) obtain the same value for $\bar{r}_k'$.

$$\bar{r}_k' = \bar{r}_k + \bar{s}\, x_k' \tag{2-50}$$

$$= \bar{r}_k + (\delta x_i/d')\ (x_k - x_i \bar{r}_k) \tag{2-51}$$

$$= \bar{r}_k - (\delta x_i/d')\ x_i r_k + (\delta x_i/d')\ x_k \tag{2-52}$$

$$= \bar{r}_k\ (1 - \delta x_i^2/d') + \bar{s}\, x_k \tag{2-53}$$

$$= \bar{r}_k\ ((d' - \delta x_i^2)/d') + \bar{s}\, x_k \tag{2-54}$$

$$= \bar{r}_k\ (d/d') + \bar{s}\, x_k \tag{2-55}$$

$$= \bar{r}_k\, \bar{c} + \bar{s}\, x_k \ . \tag{2-56}$$

Thus, only half as many multiplications are needed as usual with Givens transformations. In practice, (2-49) may be numerically unstable, and this will be shown in Chapter VI, although the instability can be detected and avoided.

If $\vec{b}$ is treated as just another column of A, then $\vec{\theta}$ is obtained, where

$$\vec{\theta} = \sqrt{D}\ \vec{\bar{\theta}} \ , \tag{2-57}$$

and an extra element of D obtained which is, in fact, just the residual

sum of squares. From (2-12), (2-42), and (2-57),

$$\overline{R}\,\vec{x} = \vec{\vec{\Theta}} \; . \tag{2-58}$$

This equation is at least as easy to solve as (2-12) since $\overline{R}$ is unit triangular.

CHAPTER III

DESCRIPTION OF TEST PROBLEMS

There are three sets of test problems, including integer matrices, polynomials, and ill-conditioned problems. They will be described in this chapter. These problems are selected because they have been used very often for testing the accuracy of methods which are used to solve linear least squares problems.

Integer Matrices

The first set of problems, (1-A) to (1-E), are taken from Jordan [21]. They have the same design matrix A but different right hand sides. Specifically, A is taken as the first five columns of the inverse of the 6×6 segment of the Hilbert matrix as follows:

$$A = \begin{bmatrix} 36 & -630 & 3360 & -7560 & 7560 \\ -630 & 14700 & -88200 & 211680 & -220500 \\ 3360 & -88200 & 564480 & -1411200 & 1512000 \\ -7560 & 211680 & -1411200 & 3628800 & -3969000 \\ 7560 & -220500 & 1512000 & -3969000 & 4410000 \\ -2772 & 83160 & -582120 & 1552320 & -1746360 \end{bmatrix}$$

The right hand side, $\vec{b}_{(A)}$, of the first problem (1-A) is taken so that the solution vector $\vec{x} = (1, 1/2, 1/3, 1/4, 1/5)^T$. Other right hand sides are formed as the following formulas:

$$\vec{b}_{(B)} = \vec{b}_{(A)} + \vec{v} \ , \tag{3-1}$$

$$\vec{b}_{(C)} = \vec{b}_{(A)} + 3\ \vec{v} \ , \tag{3-2}$$

$$\vec{b}_{(D)} = \vec{b}_{(A)} + 12\ \vec{v} \ , \tag{3-3}$$

$$\vec{b}_{(E)} = \vec{b}_{(A)} + 120\ \vec{v} \ , \tag{3-4}$$

where $\vec{v} = (4620, 3960, 3465, 3080, 2772, 2520)^T$. Therefore, the right hand sides become as follows:

$\vec{b}_{(A)}$	$\vec{b}_{(B)}$	$\vec{b}_{(C)}$	$\vec{b}_{(D)}$	$\vec{b}_{(E)}$
463	5083	14323	55903	554863
-13860	-9900	-1980	33660	461340
97020	100485	107415	138600	512820
-258720	-255640	-249480	-221760	110880
291060	293832	299376	324324	623700
-116424	-113904	-108864	-86184	185976

Since $\vec{v}$ is orthogonal to the columns of A (i.e. $\vec{v}^T\vec{a}_i = 0$, i=1, 2, ..., n), the solutions should be precisely the same for these five problems. All elements in A and $\vec{b}$ are integers; therefore they can be exactly presented in the IBM 3081 (all programs will be tested on an IBM 3081). This fact ensures that all significant digits lost are generated during computation. On the other hand, they are chosen not only because they are integer matrices but also because they are very ill-conditioned. The condition number can be roughly estimated by

$$\text{cond (A)} \simeq \max |a_{ij}| \cdot \max |a_{ij}^{-1}| \ , \tag{3-5}$$

where a_{ij}^{-1} denotes the elements in A^{-1}. Since the largest magnitude element in the Hilbert matrix is 1, the condition number of problems (1-A) to (1-E) is 441000×1 (i.e. 4.41×10^5) roughly. However, in the program LLSQF [19], which will be discussed in the next chapter, the condition number is defined by

$$\text{cond}(A) = |R|_1 \cdot |R^{-1}|_1 , \qquad (3\text{-}6)$$

where $|\ldots|_1$ denotes 1-norm and R is the decomposed triangular matrix of A as in the equation (2-12). The condition number of these problems that are computed by LLSQF is 5.18×10^6. Both condition numbers obtained by using either the formula (3-5) or (3-6) are very large.

A FORTRAN subroutine from Herndon [17], named INVHIL, which compute the inverse of Hilbert matrix is listed in Appendix E.

The inverse of a Hilbert segment is often used as a linear least squares test problem. In Businger and Gulub [7, 33], Golub [14], and Golub and Wilkinson [16], the same problem as (1-A) and some other right hand sides with the same property as the right hand sides of (1-A) to (1-E). Bjorck and Golub [5] chose the first six columns of the inverse of 8×8 Hilbert segment for the design matrix A. $\vec{b}$ was taken so that $\vec{x} = (1/3, 1/4, 1/5, 1/6, 1/7, 1/8)^T$ with various error components. Therefore, the first set of test problems are very important.

Polynomials

The second set of test problems contains two problems, (2-A) and (2-B), and are also selected from Jordan [21]. They are least squares problems for polynomials of degree n-1 with 2^m+1 equidistant data points

(i.e. $\Delta x = 2^{-m}$) on the interval $[0,1]$. The values of m and n are constrained such that x_i^r can be exactly represented in the computer, where $0 \leq r \leq n-1$ and $0 \leq i \leq m$. The solution vector has all components equal to 1. Then, problem (2-A) has m=7 and n=7 as follows:

$$A = (a_{ij}) = \left[(i-1)\ 2^{-7}\right]^{j-1} ,$$

$$1 \leq i \leq 129 ,$$

$$1 \leq j \leq 7 ,$$

$$\vec{x} = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1)^T ,$$

$$\vec{b} = A\vec{x} .$$

Problem (2-B) has m=10 and n=5 as follows:

$$A = (a_{ij}) = \left[(i-1)\ 2^{-10}\right]^{j-1} ,$$

$$1 \leq i \leq 1025,$$

$$1 \leq j \leq 5 ,$$

$$\vec{x} = (1,\ 1,\ 1,\ 1,\ 1)^T ,$$

$$\vec{b} = A\vec{x} .$$

Wampler [30] also used polynomial problems for testing his least squares programs.

Ill-Conditioned Problems

There are two problems, (3-A) and (3-B), in the third set of test problems, which are chosed from Bauer [1, 33] and Lawson and Hanson [22], respectively. They are chosen because they are very ill-conditioned. Problem (3-A) contains all integer elements in A and $\vec{b}$ as follows:

$$A = \begin{bmatrix} -74 & 80 & 18 & -11 & -4 & -8 \\ 14 & -69 & 21 & 28 & 0 & 7 \\ 66 & -72 & -5 & 7 & 1 & 1 \\ -12 & 66 & -30 & -23 & 3 & -3 \\ 3 & 8 & -7 & -4 & 1 & 0 \\ 4 & -12 & 4 & 4 & 0 & 1 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 51 \\ -61 \\ -56 \\ 69 \\ 10 \\ -12 \end{bmatrix}.$$

The exact solution to this problem should be

$$\vec{x} = (1, 2, -1, 3, -4, 0)^T .$$

The design matrix A of problem (3-B) is as follows:

$$A = \begin{bmatrix} -.13405547 & -.20162827 & -.16930778 & -.18971990 & -.17387234 \\ -.10379475 & -.15766336 & -.13346256 & -.14848550 & -.13597690 \\ -.08779597 & -.12883867 & -.10623007 & -.12011796 & -.10932972 \\ .02058554 & .00335331 & -.01641270 & .00078606 & .00271659 \\ -.03248093 & -.01876799 & .00410639 & -.01415894 & -.01384391 \\ .05967662 & .06667714 & .04352153 & .05740438 & .05024962 \\ .06712457 & .07352437 & .04489770 & .06471862 & .05876455 \\ .08687186 & .09368296 & .05672327 & .08141043 & .07302330 \\ .02149662 & .06222662 & .07213486 & .06200069 & .05570931 \\ .06687407 & .10344506 & .09153849 & .09508223 & .08393667 \\ .15879069 & .18088339 & .11540692 & .16160702 & .14796479 \\ .17642887 & .20361830 & .13057860 & .18385729 & .17005549 \\ .11414080 & .17259611 & .14816471 & .16007466 & .14374096 \\ .07846038 & .14669563 & .14365800 & .14003842 & .12571177 \\ .10803175 & .16994623 & .14971519 & .15885312 & .14301547 \end{bmatrix}$$

The right hand side of problem (3-B) is that

$$\vec{b} = (-.4361,\ -.3437,\ -.2657,\ -.0392,\ .0193,$$
$$.0747,\ .0935,\ .1079,\ .1930,\ .2058,$$
$$.2606,\ .3142,\ .3539,\ .3615,\ .3647)^T .$$

The condition numbers of (3-A) and (3-B) are 3.66×10^6 [1, 33] and 1.39×10^7 [22] respectively.

Ill-conditioned problems also appears in Martin et al. [24, 33]. He chose a 7×7 Hilbert matrix. In order to avoid rounding errors, the matrix was scaled by the factor 360360 so that all coefficients were integer. Since orthogonal decomposition methods avoid magnifying condition number, ill-conditioned problems are important in the accuracy test for linear least squares problems.

CHAPTER IV

DESCRIPTION OF PROGRAMS

There are four programs to be tested in this study. All of them are coded in standard FORTRAN and named GIVEN, ORTHL, BLSQS, and LLSQF, respectively. Program listings are collected in Appendix A to Appendix D as well as their test programs in Appendix F to Appendix I. This chapter will have a detailed description for the program GIVEN with complete user instructions. After that, description of ORTHL, BLSQS, and LLSQF will be presented briefly.

GIVEN - Implementation of Givens Transformations

This program is converted from Gentleman [12, 13] in which ALGOL procedures are presented. It is an implementation of Givens transformations. However, an option indicator, ITYPE, has been used in GIVEN as an input parameter which does not appear in Gentleman. ITYPE will be explained in user instructions. Figure 1 shows the program structure of GIVEN. It is obvious that GIVEN controls the program flow and connects to the user supplied calling program. The functions of these subroutines and user instructions will be described in the following sections.

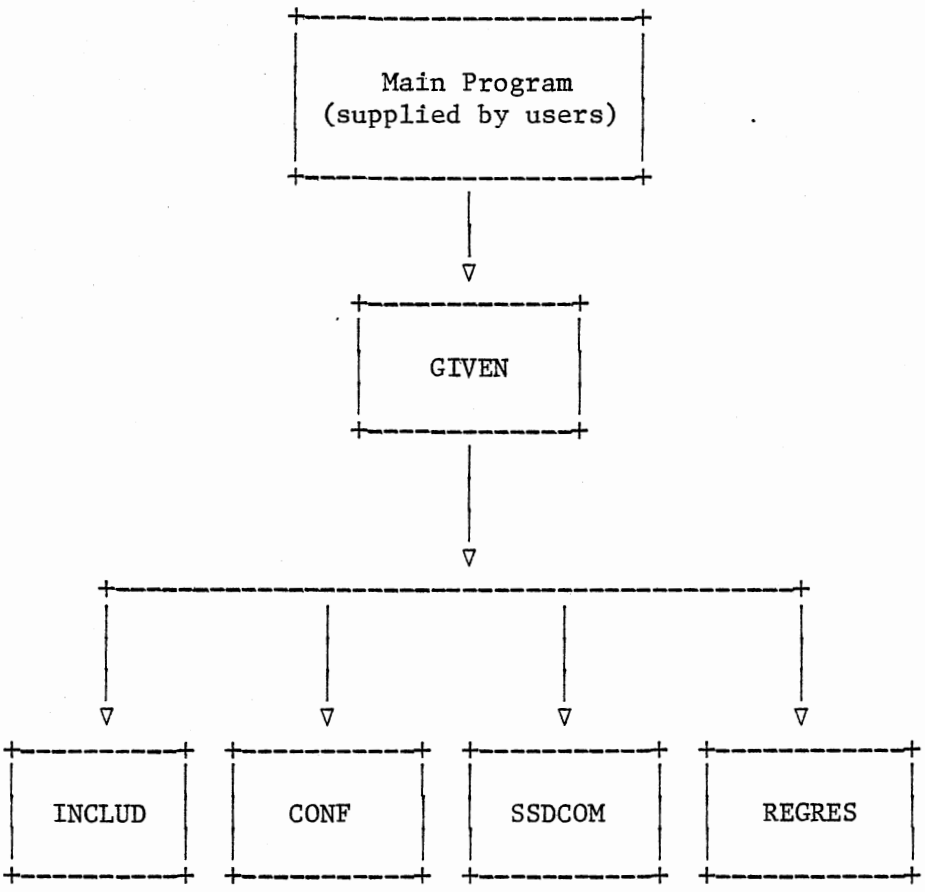


Figure 1. Program Sturcture of GIVEN.

Functions of Subroutines in GIVEN

1. Subroutine GIVEN. Subroutine GIVEN controls the data input and produces the results of the regression solution. Furthermore, it calls the other four subroutines INCLUD, CONF, SSDCOM, and REGRES to perform the least squares computations. Since the results will be printed out automatically by GIVEN itself, users need not to worry about the output.

2. Subroutine INCLUD. This subroutine updates D, $\overline{R}$, $\overrightarrow{\overrightarrow{\theta}}$, and SSERR to include the effect of a new row of A and $\vec{b}$. For an initial decomposition, D, $\overline{R}$, $\overrightarrow{\overrightarrow{\theta}}$, and SSERR should be set to zero before processing the first row.

3. Subroutine CONF. Given $\overline{R}$ and some integer J, CONF finds the contrast which could not be estimated if D were zero; that is, finds the linear combination of the first J columns of A which would vanish [11]. Most cases in which A is not of full rank (that is, where the independent variables are confounded) can readily be detected by some D becoming small or vanishing. The common method of resolving the resulting indeterminacy is to find the confounded contrast as produced by this subroutine, and then either to force one of the confounded variables (those with non-zero coefficients in the contrast) to have regression coefficient zero, or to orthogonalize the regression coefficients of a subset of confounded variables to the others [12]. The later is achieved by requiring the vanishing of a linear combination of regression coefficients equal to the confounded contrast for the components in the subset, and zero for other components. Constraints like either of the above, which merely resolve indeterminacy, can readily be imposed by including them as extra rows of A and $\vec{b}$.

4. Subroutine SSDCOM. Given D and $\vec{\vec{\Theta}}$, this subroutine computes the sum of squares decompositions. This, and not the regression coefficients, is what is needed for standard hypothesis testing.

5. Subroutine REGRES. This subroutine computes the regression coefficients $\vec{x}$ from the input quantities $\bar{R}$ and $\vec{\vec{\Theta}}$.

Instructions for Users of GIVEN

1. Important Symbols. Important symbols are shown in Table I, and Table II presents their attributes, dimensions, and other characteristics.

2. Calling Sequence. Formal parameters are described in Table I and Table II. The calling sequence is

```
CALL GIVEN (NCOL,NR,ITYPE,AROW,D,TBAR,RBAR).
```

3. Input Sequences. Data input via input device are WEIGHT, AROW, BROW, and/or NZERO. These variables are well described in Table I and Table II. The option indicator ITYPE for input data may have the value 1 or 2. ITYPE=1 indicates that the design matrix is a normal matrix (that is, A is not sparse). On the contrary, ITYPE=2 indicates that the design matrix A is a sparse matrix. Users must note that different input sequences of data cards are used for each option as follows:

ITYPE = 1:

record 1: WEIGHT of the first row of A

record 2: the first row of $[A|\vec{b}]$

record 3: WEIGHT of the second row of A

record 4: the second row of $[A|\vec{b}]$

TABLE I

SYMBOL LEGEND

Symbol	Description
NCOL	number of unknowns
NR	dimension of RBAR; NR=NCOL*(NCOL-1)/2
TOL1	tolerance for detecting rank deficiency
TOL2	tolerance for identifying the confounded variables
ITYPE	input sequence option indicator
AROW	one row of the design matrix A to be processed currently
BROW	the current element of right hand side $\vec{b}$
WEIGHT	weight of each row of A
NZERO	column index of the nonzero element in the current row
D	the diagonal scaling matrix
RBAR	the superdiagonal elements of $\bar{R}$, stored sequentially by rows
TBAR	$\vec{\vec{\theta}}$, where $\sqrt{D}\,\vec{\vec{\theta}}$ is the vector of orthogonal coefficients
SSERR	the sum of squares error
J	see description in subroutine CONF
CONTRA	the coefficients of the confounded trast among the independent variables if the system is rank deficient
SS	the sum of squares decomposition, i.e. the squares of the orthogonal coefficients
BETA	the regression coefficients

TABLE II

ATTRIBUTES AND CHARACTERISTICS OF
VARIABLES IN GIVEN

Symbol	Attr.	Dim.	GIVEN	INCLUD	CONF	SSDCOM	REGRES
NCOL	int		read	in	in	in	in
NR	int		in	in	in		in
TOL1	real		cons				
TOL2	real		cons				
ITYPE	int		in				
AROW	real	1:NCOL	read	in/out			
BROW	real		read	in/out			
WEIGHT	real		read	in			
NZERO	int		read				
D	real	1:NCOL		in/out		in	
RBAR	real	1:NR		in/out	in		in
TBAR	real	1:NCOL		in/out		in	in
SSERR	real			in/out			
J	int		val		in		
CONTRA	real	1:NCOL			out		
SS	real	1:NCOL				out	
BETA	real	1:NCOL					out

Abbreviations:

Attr. - Attribute
Dim. - Dimension
int - integer

in - input parameter
out - output parameter
cons - constant
val - value

.
.
.
(repeat record 1 and 2 for the next row of $[A|\vec{b}]$)
.
.
.
(until WEIGHT=0)

Note: a) WEIGHT=0 indicates end of input data.
b) Set all values of WEIGHT to 1 for unweighted problems.

ITYPE = 2:

record 1: WEIGHT of the first row of A

record 2: BROW of the first row

record 3: NZERO

record 4: AROW(NZERO) of the first row
.
. (repeat record 3 and record 4 for the next
. row until NZERO=0)
.
(repeat from record 1 for the next row of A)
.
.
.
(until WEIGHT=0)

Note: NZERO=0 indicates end of each row.

4. Input Data Format. The following data formats are built into subroutine GIVEN.

WEIGHT - E14.7

NZERO - I2

AROW - E14.7

BROW - E14.7

Users, perhaps, need to change them if the formats are not suitable to their problem.

5. Output. The output of this program contains the number of equations, RBAR, D, SSERR, and the solution vector AROW. They are output with clear expositions.

6. Input and Output Devices. Unit 5 is used as the input device, and unit 6 is used as the output device. Users may change them merely by changing the values of IN and LP if they desire.

7. Tolerances. The values of the tolerances, TOL1 and TOL2, which are used to detect rank definiencies and to identify the confounded variables, are respectively set to 10^{-16} and 10^{-8} for running on an IBM 3081 with 56-bit mantissa in double precision. For single precision computation, they are set to 10^{-8} and 10^{-8}, respectively. Too big a tolerance will make the result inaccurate. Usually, user may set TOL1 to 10^{-k}, where k is the approximate number of decimal digits that can be expressed in the machine, and TOL2 is a small number relative to the magnitude of elements in the solution vector. Users may change the values of these two tolerances in subroutine GIVEN if necessary.

ORTHL and BLSQS - Implementations of Modified Gram-Schmidt

Both ORTHL and BLSQS are implementations of the modified Gram-Schmidt algorithm with iterative refinement of the solutions [3, 4]. Iterative refinement is a scheme for improving an approximate solution to the linear least squares problems. This scheme was first proposed by Golub [14] and used also in Bauer [1, 33].

Given the approximate solution $\vec{x}$ of the least squares problem of minimizing $|\vec{b} - A\vec{x}|_2$, the method of iterative refinement can be defined briefly as the following statements:

1. Compute $\vec{r} = \vec{b} - A\vec{x}$ in double precision.

2. Solve $d = A^{+}\vec{r}$ in single precision.

3. $\vec{x} \leftarrow\!- \vec{x} + \vec{d}$.

Iteration should be terminated when $\vec{d}$ becomes negligible compared to $\vec{x}$. Note that $\vec{r}$ in statement 1 is required to be computed in double precision if refinement is to work correctly. The accuracy achieved by using iterative refinement will be approximately the same as that obtained by a double precision decomposition [3].

ORTHL was converted by Chandler [9] from Bauer's ALGOL procedure ORTHOLIN2, and BLSQS was developed by Bolliger [6] from the ALGOL algorithm by Bjorck [4]. Both programs use double precision for the computation of inner products in iterative refinement. In order to compare with GIVEN in pure single precision and/or pure double precision, these programs have been slightly modified to eliminate mixed precision arithmetic.

ALGOL procedures are available in Bauer [1, 33], Bjorck [4], Clayton [10], and Walsh [28].

LLSQF - Implementation of Householder Transformations

LLSQF is an implementation of Householder transformations for solving linear least squares problems, and is adapted from the IMSL Library [16]. This program also implements the iterative refinement scheme to reduce the error in the computed least squares solutions. The original LLSQF also uses double precision for iterative refinement; therefore, it was changed to pure single/double precision in order to

agree with the other programs.

LLSQF calls some other subroutines and/or functions which are also members of IMSL Library including UERTST, UGETIO, SASUM, SDOT, SNRM2, VHS12, DASUM, DDOT, and DNRM2. These subroutines are shown in Figure 2. DASUM, DDOT, and DNRM2 are only used in double precision arithmetic, and SASUM, SDOT, and SNRM2 are only used in single precision arithmetic. Since VHS12 is the subroutine to perform iterative refinement, it is used in both single and double precision arithmetic. However, this study does not compare mixed precisions; therefore, DVHS12 and SVHS12 are generated from VHS12 for double precision and single precision, respectively. Furthermore, UERTST and UGETIO are used to output some messages. They involve integer and character variables only; hence, they can be used in both precision computations.

ALGOL procedures that implement Householder transformations are available in Bjorck et al. [5], Businger et al. [7, 33], and FORTRAN subroutines in Lawson and Hanson [22].

All the programs mentioned above, including GIVEN, ORTHL, BLSQS, and LLSQF have been run on an IBM 3081 in both single and double precision for the test problems that have been mentioned in Chapter III. The test results will be shown in Chapter VI.

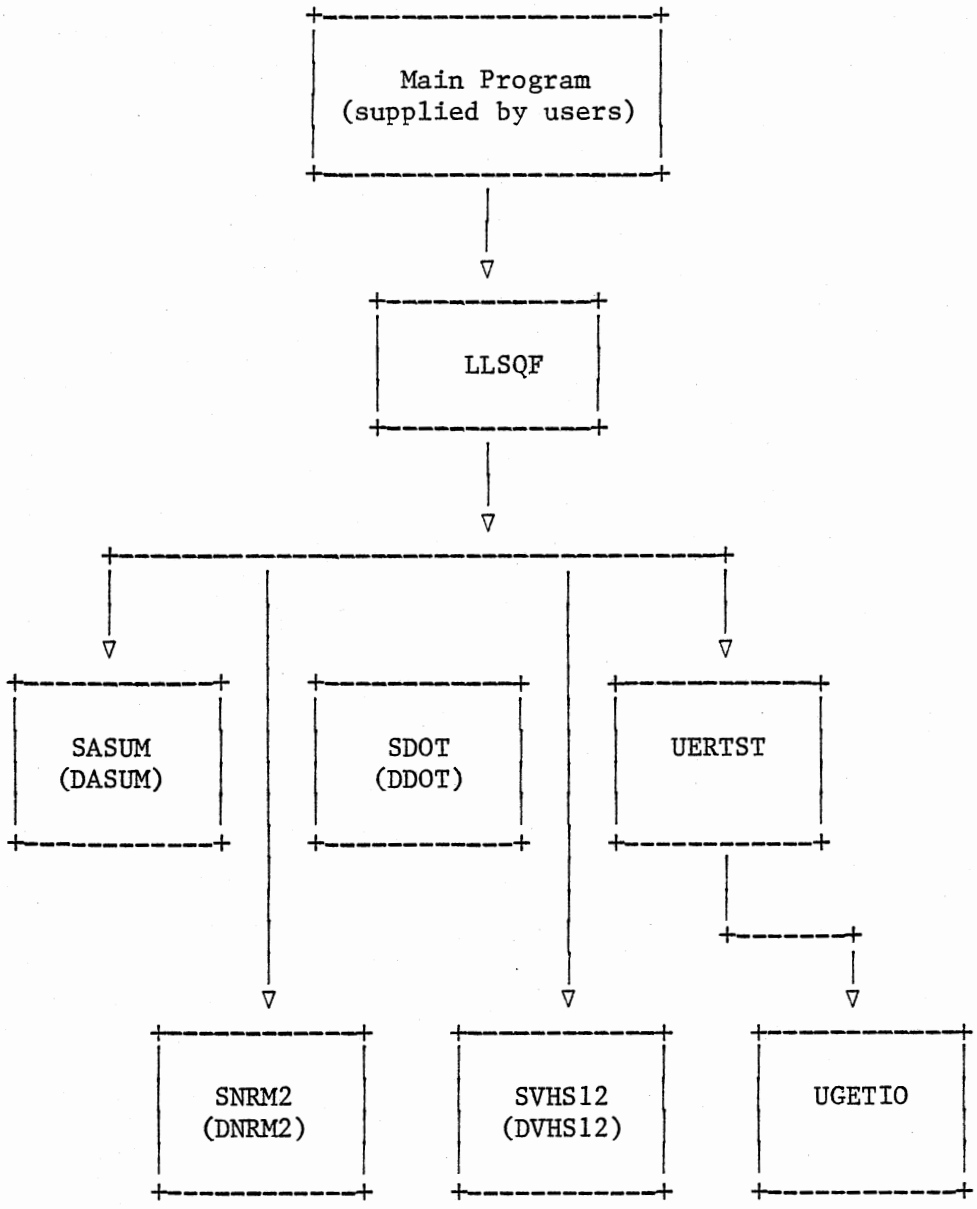


Figure 2. Program Structure of LLSQF.

CHAPTER V

COMPARISON WITH RESPECT TO STORAGE, TIME, AND ERROR BOUNDS

Storage Requirement

One of the advantages of Givens transformations is that the design matrix can be processed one row at a time. The importance of this is that it is very storage efficient. Design matrices are frequently too large to be stored in high speed memory, and hence they must be fetched as required; furthermore, it turns out that the natural and convenient way to fetch them is usually by rows [11].

Lawson and Hanson [22] established a modified approach of Householder transformations and modified Gram-Schmidt for transforming the matrix $[A:\vec{b}]$ to upper triangular form without requiring that the entire matrix $[A:\vec{b}]$ be in computer storage at one time. That is, Householder transformations and modified Gram-Schmidt method can be organized to accumulate blocks of rows sequentially to handle problems in which $m \times n$ is very large and $m >> n$.

The matrix A and the vector $\vec{b}$ are partitioned in the form:

$$A = \begin{bmatrix} A_1 \\ \cdot \\ \cdot \\ \cdot \\ A_q \end{bmatrix}, \qquad b = \begin{bmatrix} \vec{b}_1 \\ \cdot \\ \cdot \\ \cdot \\ \vec{b}_q \end{bmatrix}, \tag{5-1}$$

where each A_i is $m_i \times n$ and each b_i is a vector of length m_i. Of course,

$m = m_1 + m_2 + \ldots + m_q$. The smallest value of m_i may be 1. The algorithm will construct a sequence of triangular matrices $[R_i : \vec{d}_i]$, i=1, ... , q, with the property that the least squares problem

$$R_i \vec{x} = \vec{d}_i \tag{5-2}$$

has the same solution set and the same residual norm as the problem

$$\begin{bmatrix} A_1 \\ \cdot \\ \cdot \\ \cdot \\ A_q \end{bmatrix} \cdot \vec{x} = \begin{bmatrix} \vec{b}_1 \\ \cdot \\ \cdot \\ \cdot \\ \vec{b}_q \end{bmatrix} \tag{5-3}$$

In Lawson and Hanson [22], if

$$v = \begin{cases} 0 & (\text{if } j=0), \\ \sum_{i=1}^{j} m_i & (\text{if } j>0), \end{cases} \tag{5-4}$$

and

$$u = \max_{1 \le j \le q} \{ m_j + \min [n+1, v_{j-1}] \} , \tag{5-5}$$

then the algorithm can take place in a computer storage array W having at least u rows and n+1 columns. Further, by more complicated programming the storage required could be reduced by exploiting the fact that each matrix $[R_j : \vec{d}_j]$ is upper triangular.

Although this sequential accumulation approach can reduce storage requirements, it has two disadvantages. First, the operation count is increased as the block size is decreased. This fact will be discussed

in the next section. Secondly, the whole matrix A still needs to be stored in main memory when iterative refinement is required.

Table III shows the work arrays required for the programs GIVEN, ORTHL, BLSQS, and LLSQF. Other single spaced variables are ignored since they are small compared to arrays. Although the storage requirements are various for alternative coding skill, the method based on Givens transformations is the most attractive in storage requirement since $O(n^2) << O(mn)$ especially when $m >> n$. In the case of $m >> n$, even if pure double precision arithmetic is used in Givens transformations, the storage required is still much less than the storage needed by Householder transformations or modified Gram-Schmidt in which mixed precision arithmetic is used.

Time Requirement

In discussing the time required for orthogonal decomposition methods, only the number of operations is compared. The times needed for I/O, compilation, and loading, etc. are not considered although they might dominate the time required to solve a least squares problem.

The number of operations has been discussed in Lawson and Hanson [22]. Operations could be additions/subtractions, multiplications/divisions, or square roots. Their comparisons are listed in Table IV. Apparently Gentleman's modification of the Givens method is competitive with the standard Householder method for nonsequential processing [12, 22]. The sequential Householder accumulation increases the number of operations as the block size is decreased. In the worst case of k=1, the operations counts for additions and multiplications are approximately doubled relative to the number of operations for nonsequential

TABLE III

STORAGE REQUIREMENTS FOR PROGRAM IMPLEMENTATIONS

GIVEN		ORTHL		BLSQS		LLSQF	
Array Name	Array Size	Array Name	Array Size	Array Name	Array Size	Array Name	Array Size
AROW	n	A	(m,n)	A	(m,n+1)	A	(m,n)
D	n	B	m	B	m	B	m
RBAR	n(n+1)/2	X	m	X	n	X	n
TBAR	n	P	m	RES	m	H	n
		PP	n	QR	(m+n,n)		
		D	n	IPIV	n		
		R	(m,n)	XV	n+1		
		U	(m,n)	RESV	m		
				D	n		
				F	m+n		
				G	m+n		
				Y	n		
				XMY1	m		
				XMY2	n		
TOTAL							
$O(n^2)$		O(mn)		O(mn)		O(mn)	

TABLE IV

COMPARISON OF OPERATIONS REQUIRED

Method	add/subt	mult/div	squ root
Givens (original)	P**	2P	n
Givens (modified)	P	P	0
Householder* (original)	P	P	n
Householder* (sequential***)	P(k+1)k	P(k+1)k	n
Modified Gram-Schmidt*	$P+n^3/3$	$P+n^3/3$	0

* Operation count of this method does not include the operations for iterative refinement.

** P has the value of $mn^2-n^3/3$ where m is the number of rows of A; n is the number of columns of A.

*** Suppose that the entering blocks of data each contains k rows. That is, k=m/q where q is the number of blocks to be processed sequentially.

processing. The Householder transformations always require $n^3/3$ operations fewer than the modified Gram-Schmidt method since the matrix Q in Householder transformations is not explicitly computed.

The time required for Householder transformations and the modified Gram-Schmidt method are increased when iterative refinement has been implemented. Tradeoffs involve time, storage, and accuracy in the implementation of iterative refinement. Although actural comparative performance of computer programs based on any of these methods will also depend strongly on coding details, the modified Givens transformation is more economically attractive and convenient to be used than the other methods.

Error Bounds

Wilkinson [32] gives an error analysis of a single Givens transformation for formulas (2-36) to (2-41). The desired and computed values of r_k' and x_k' can be bounded by

$$\left| \begin{matrix} fl(r') - r_k' \\ fl(x') - x_k' \end{matrix} \right|_2 \leq 6\varepsilon \left| \begin{matrix} r_k \\ x_k \end{matrix} \right|_2 , \tag{5-6}$$

where ε is the largest number such that $fl(1+\varepsilon)=1$. A similar calculation for the Givens transformations without square roots, as formulas (2-43) to (2-48), shows that the difference between the desired and computed values of $\sqrt{d}'\ \bar{r}_k'$ and $\sqrt{\delta}'\ x_k'$ can be bounded by

$$\left| \begin{matrix} fl(\sqrt{d}')\ fl(\bar{r}_k') - \sqrt{d}'\ \bar{r}_k' \\ fl(\sqrt{\delta}')\ fl(x_k') - \sqrt{\delta}'\ x_k' \end{matrix} \right|_2 \leq 7.5\varepsilon \left| \begin{matrix} \sqrt{d}\ \bar{r}_k \\ \sqrt{\delta}\ x_k \end{matrix} \right|_2 , \tag{5-7}$$

where the factor 7.5 is very generous.

Gentleman [12] indicates that the cheaper formula (2-49) is numerically unstable if d is very small compared to δx_i^2. It produces terrible results for least squares problems with very well-conditioned design matrices. Thus, he established a more general form of bounds as follows:

$$\left| \begin{array}{c} fl(\sqrt{d'})\ fl(\bar{r}_k') - \sqrt{d'}\ \bar{r}_k' \\ fl(\sqrt{\delta'})\ fl(x_k') - \sqrt{\delta'}\ x_k' \end{array} \right|_2 \leq$$

$$\{(4.52)^2+[4.52+8.04(d'/d)^{\frac{1}{2}}]^2\}^{\frac{1}{2}}\ \varepsilon \left| \begin{array}{c} \sqrt{d}\ \bar{r}_k \\ \sqrt{\delta}\ x_k \end{array} \right|_2 . \qquad (5\text{-}8)$$

For (5-8) it is clear that the instability is exactly associated with d'/d. Gentleman suggests that the formula (2-49) should not be used unless $d'/d \leq 100$ and use formula (2-48) instead for unstable cases. When $d'/d = 100$, the bound is obtained by

$$\left| \begin{array}{c} fl(\sqrt{d'})\ fl(\bar{r}_k') - \sqrt{d'}\ \bar{r}_k' \\ fl(\sqrt{\delta'})\ fl(x_k') - \sqrt{\delta'}\ x_k' \end{array} \right|_2 \leq 85.04\varepsilon \left| \begin{array}{c} \sqrt{d}\ \bar{r}_k \\ \sqrt{\delta}\ x_k \end{array} \right|_2 . \qquad (5\text{-}9)$$

A backward error analysis for the solution of linear least squares problems by Givens transformations is presented in Gentleman [12]. The difference between the computed triangular matrix and some exactly orthogonal transformation of the original matrix is bounded by

$$| U - \hat{Q}^T A |_F \leq \eta n^{\frac{1}{2}} [m+(n-5)/4] (1+\eta)^{m+n-3} | A |_F , \qquad (5\text{-}10)$$

$$| \vec{u} - \hat{Q}^T \vec{b} |_2 \leq \eta n^{\frac{1}{2}} [m+(n-5)/4] (1+\eta)^{m+n-3} | \vec{b} |_2 , \qquad (5\text{-}11)$$

where U is an m×n upper triangular matrix either equal to the computed matrix R or the product of the computed matrix $\sqrt{D}$ with the computed matrix $\bar{R}$ as in the formula (2-42); $\vec{u}$ is an m-vector whose leading n elements are either $\bar{\theta}$ or $\sqrt{D}\,\bar{\bar{\theta}}$ as in the formula (2-57). $\hat{Q}^T$ is the orthogonal m×m matrix that is the product of exact plane rotations (they are not the same plane rotations had been used throughout); η is either 6ε, 7.5ε, or 85ε as appropriate; and $|...|_F$ denoted the Frobenius norm. The error in backsubstituting a triangular system is negligible, therefore it is not discussed in this study.

For the method based on the modified Gram-Schmidt algorithm, Bjorck [2, 3] derives bounds for errors related to the factorization of A and $\vec{b}$ as follows:

$$| R - \hat{Q}^T A |_F \leq 1.9(n-1)^{\frac{1}{2}} n\varepsilon | A |_F , \qquad (5\text{-}12)$$

$$| \vec{y} - \hat{Q}^T \vec{b} |_2 \leq 1.9\, n^{\frac{1}{2}} (n+1)\varepsilon | \vec{b} |_2 . \qquad (5\text{-}13)$$

These bounds are valid if inner-products are accumulated in double precision. The bounds must be increased by a factor of 2m/3+1 for single precision arithmetic.

Lawson and Hanson [22] analyze the error bounds for Householder transformations clearly. The error associated with the application of k succussive Householder transformations is bounded by

$$| A_{k+1} - Q_k \cdots Q_1 A |_F \leq (6m-3k+40)k\varepsilon | A |_F . \qquad (5\text{-}14)$$

CHAPTER VI

TEST RESULTS

Test results are listed from Table V to Table XII. The solutions listed in these tables for BLSQS, LLSQF, and ORTHL are obtained with iterative refinements. The results obtained without iterative refinements will be discussed later. Table XIII shows the average number of significant digits lost for each test except for problem (3-B). The average number of significant digits lost, S, is obtained by

$$S = \left[\sum_{i=1}^{n} (d - c_i) \right] / n, \tag{6-1}$$

where c_i is the number of significant digits gained correctly for each element in the solution vector $\vec{x}$. Here d is the approximate number of decimal digits which can be expressed in the computer. The value of d can be computed by the formula

$$16^{-h+1} = 10^{-d+1}. \tag{6-2}$$

Then

$$d = 1 + (h-1) \log 16 / \log 10 , \tag{6-3}$$

where h is the number of hexidecimal digits in the mantissa. For the IBM 3081, h is 14 for double precision and 6 for single precision.

TABLE V

TEST RESULTS OF PROBLEMS (1-A) AND (1-B) IN
DOUBLE PRECISION ARITHMETIC

		GIVEN	BLSQS*	LLSQF	ORTHL
(1-A)	x_1	0.9999999999999716	0.9999999999999932	1.0000000000078881	1.0000000000001756
	x_2	0.5000000000002174	0.5000000000001261	0.5000000000246713	0.5000000000004070
	x_3	0.3333333333334898	0.3333333333334255	0.3333333333435627	0.3333333333334549
	x_4	0.2500000000000891	0.2500000000000526	0.2500000000043859	0.2500000000000364
	x_5	0.2000000000000374	0.2000000000000221	0.2000000000015385	0.2000000000000085
(1-B)	x_1	1.0000000010618187	1.0000000001375334	0.9999999268401792	1.0000000005879908
	x_2	0.5000000035142349	0.5000000004449910	0.4999999755325871	0.5000000019750643
	x_3	0.3333333348321534	0.3333333335207196	0.3333333228376746	0.3333333341825702
	x_4	0.2500000006536209	0.2500000000810422	0.2499999954059259	0.2500000003722848
	x_5	0.2000000002318849	0.2000000000285829	0.1999999983659048	0.2000000001325607

* The same results have been obtained by BLSQS without iterative refinement.

TABLE VI

TEST RESULTS OF PROBLEMS (1-C) AND (1-D) IN
DOUBLE PRECISION ARITHMETIC

		GIVEN	BLSQS*	LLSQF	ORTHL
(1-C)	x_1	1.000000031850952	0.999999746512339	0.9999997803422034	1.000000017650046
	x_2	0.5000000105405908	0.4999999915582043	0.4999999265421750	0.5000000059285523
	x_3	0.3333333378286429	0.3333333297206321	0.3333333018233540	0.3333333358824851
	x_4	0.2500000019602838	0.2499999934209580	0.2499999862079275	0.2500000011174761
	x_5	0.2000000006954288	0.1999999994389965	0.1999999950942623	0.2000000003979006
(1-D)	x_1	1.000000127407247	0.9999997549168600	0.9999991211205443	1.000000070596765
	x_2	0.5000000421628930	0.4999999178431638	0.4999997060914313	0.5000000237131642
	x_3	0.3333333513146082	0.3333333980469485	0.3333332072614688	0.3333333435295277
	x_4	0.2500000078410853	0.2499999845425777	0.2499999448180365	0.2500000044697354
	x_5	0.2000000027816782	0.1999999944988563	0.1999999803722587	0.2000000015915454

* The same results have beenoobtained by BLSQS without iterative refinement.

TABLE VII

TEST RESULT OF PROBLEM (1-E) IN
DOUBLE PRECISION ARITHMETIC

		GIVEN	BLSQS*	LLSQF	ORTHL
	x_1	1.000001274078713	0.9999989325845028	0.9999912105831275	1.0000007059858107
	x_2	0.5000004216303036	0.4999996425678251	0.4999970607166682	0.5000002371404218
(1-E)	x_3	0.3333335731463022	0.3333331799077687	0.3333320725320155	0.3333334352997964
	x_4	0.2500000784107963	0.2499999328168350	0.2499994481447138	0.2500000446995640
	x_5	0.2000000278167136	0.1999999760966501	0.1999998037100309	0.2000000159163011

* The same result has been obtained by BLSQS without iterative refinement.

TABLE VIII

TEST RESULTS OF PROBLEMS (1-A) AND (1-B) IN SINGLE PRECISION ARITHMETIC

		GIVEN	BLSQS*	LLSQF	ORTHL
(1-A)	x_1	1.011581	0.000014	0.850402	5.201400
	x_2	0.503354	0.708983	0.455584	2.017883
	x_3	0.334663	0.308322	0.315644	1.012878
	x_4	0.250557	0.163342	0.242648	0.555840
	x_5	0.200193	0.135602	0.197483	0.310946
(1-B)	x_1	11.71515	90.65291	26.66490	-13.8247
	x_2	4.10072	30.32495	9.09479	-4.3502
	x_3	1.88199	13.09037	4.02362	-1.7189
	x_4	0.92904	5.82361	1.86638	-0.6397
	x_5	0.44183	2.17994	0.77524	-0.1143

* The same results have been obtained by BLSQS without iterative refinement.

TABLE IX

TEST RESULTS OF PROBLEMS (2-A) AND (2-B) IN
DOUBLE PRECESION ARITHMETIC

		GIVEN	BLSQS*	LLSQF	ORTHL
(2-A)	x_1	0.9999999999999859	1.000000000000000	1.000000000000012	1.000000000000000
	x_2	0.9999999999999628	1.000000000000004	0.9999999999999818	0.9999999999999995
	x_3	1.000000000000311	0.9999999999999639	1.000000000001359	1.000000000000006
	x_4	0.9999999999989189	1.000000000000121	0.9999999999955208	0.9999999999999682
	x_5	1.000000000001830	0.9999999999997997	1.000000000007294	1.000000000000073
	x_6	0.9999999999985991	1.000000000000162	0.9999999999942267	0.9999999999999191
	x_7	1.000000000000391	0.9999999999999489	1.000000000001773	1.000000000000033
(2-B)	x_1	0.9999999999998568	1.000000000000000	0.9999999999999902	1.000000000000000
	x_2	1.000000000000019	0.9999999999999998	1.000000000000063	0.9999999999999996
	x_3	1.000000000000110	1.000000000000000	0.9999999999998528	1.000000000000002
	x_4	1.000000000000111	1.000000000000000	1.000000000000169	0.9999999999999974
	x_5	0.9999999999999005	0.9999999999999998	0.9999999999999259	1.000000000000001

* The same results have been obtained by BLSQS without iterative refinement.

TABLE X

TEST RESULTS OF PROBLEMS (2-A) AND (2-B) IN SINGLE PRECISION ARITHMIC

		GIVEN	BLSQS*	LLSQF	ORTHL
(2-A)	x_1	0.9999268	1.0000010	0.9999926	0.9999998
	x_2	0.9998756	0.9999968	0.9999421	0.9999859
	x_3	1.0007162	1.0000172	1.0027380	1.0000496
	x_4	0.9984465	1.0000467	0.9188493	0.9999377
	x_5	1.0018578	0.9997123	2.4426165	1.0000238
	x_6	0.9991331	1.0003977	-10.0723238	0.9999839
	x_7	1.0000420	0.9998291	33.3302765	1.0000153
(2-B)	x_1	0.9993522	1.0000000	0.9998312	0.9999995
	x_2	1.0001822	0.9999988	1.0005016	0.9999942
	x_3	1.0002918	1.0000086	0.9987872	1.0000296
	x_4	1.0007191	0.9999855	1.0015106	0.9999450
	x_5	0.9994494	1.0000067	0.9993319	1.0000286

* The same results have been obtained by BLSQS without iterative refinement.

TABLE XI

TEST RESULTS OF PROBLEMS (3-A) AND (3-B) IN DOUBLE PRECISION ARITHMETIC

		GIVEN	BLSQS*	LLSQF	ORTHL
(3-A)	x_1	1.000000000000611	0.9999999999999163	0.9999999999997195	1.000000000000649
	x_2	1.999999999999537	2.000000000000065	2.000000000000240	1.999999999999441
	x_3	-1.000000000002217	-0.9999999999996917	-0.9999999999989253	-1.000000000002500
	x_4	3.000000000014941	2.999999999997936	2.999999999992818	3.000000000016684
	x_5	-3.999999999953892	-4.000000000006369	-4.000000000022290	-3.999999999948234
	x_6	-0.538999×10^{-10}	0.814049×10^{-11}	0.284400×10^{-10}	-0.660476×10^{-10}
(3-B)	x_1	-74.91579305899307	-74.91579307444269	-74.91579316041095	-74.91579308345429
	x_2	100.6816561346046	100.6816561559755	100.6816562753221	100.6816561634514
	x_3	-79.80442261521869	-79.80442263226947	-79.80442272701437	-79.80442264221179
	x_4	92.81699663658507	92.81699665660886	92.81699676690292	92.81699666826094
	x_5	-80.05289259765479	-80.05289261577138	-80.05289271597731	-80.05289262632364

* The same results have been obtained by BLSQS without iterative refinement.

TABLE XII

TEST RESULTS OF PROBLEMS (3-A) AND (3-B) IN SINGLE PRECISION ARITHMETIC

		GIVEN	BLSQS	LLSQF	ORTHL
(3-A)	x_1	1.013959	-5.945167	-1 899553	1.001654
	x_2	1.988305	0.763933	0.601×10^{-8}	1.998547
	x_3	-1.053028	-0.278558	4.584720	-1.006414
	x_4	3.354614	0.0	-0.408×10^{-6}	3.042818
	x_5	-2.901019	0.0	-2.896753	-3.866953
	x_6	-1.402582	0.0	0.354×10^{-6}	-0.169660
(3-B)	x_1	1.60539	87.530	43.7141	$.1235\times10^{7}$
	x_2	-5.63163	-124.738	-64.0907	$-.1713\times10^{7}$
	x_3	4.50359	99.256	50.9108	$.1362\times10^{7}$
	x_4	-5.14306	-115.851	-59.1630	$-.1589\times10^{7}$
	x_5	9.02764	109.432	58.1077	$.1442\times10^{7}$

TABLE XIII

COMPARISON OF SIGNIFICANT DIGITS LOST

		GIVEN	BLSQS	LLSQF	ORTHL	Jordan*	Jordan**
(1-A)	D.P.	3.85	3.65	6.05	4.05	4.8	5.0
	S.P.	4.62	6.82	6.22	all		
(1-B)	D.P.	8.25	7.25	9.25	8.05	7.0	8.0
	S.P.	all	all	all	all		
(1-C)	D.P.	8.65	8.65	9.45	8.45	7.5	8.4
	S.P.	all	all	all	all		
(1-D)	D.P.	9.25	9.65	10.05	9.05	8.1	9.0
	S.P.	all	all	all	all		
(1-E)	D.P.	10.25	10.25	11.05	10.05	9.1	10.0
	S.P.	all	all	all	all		
(2-A)	D.P.	3.79	2.56	4.08	1.84	3.0	3.9
	S.P.	3.02	2.16	3.73	1.73		
(2-B)	D.P.	3.25	0.26	2.85	1.12	0.7	1.0
	S.P.	3.02	1.02	2.44	1.42		
(3-A)	D.P.	4.98	3.98	4.82	4.98	--	--
	S.P.	5.85	5.85	all	4.85		

* Test results from Jordan [21] for modified Gram-Schmidt algorithm

** Test results from Jordan [21] for Householder transformations

Table XIV and XV show the rank point obtained for each test. The rank point goes from 1 to 4 for the largest number of significant digits lost to the smallest number of significant digits lost for each program on each test problem. That is, the most accurate program obtains four points, the next accurate one gets three points, and so on. If two tests lost the same number of digits, then they get the same rank point which is the average of the next two rank points. For example, GIVEN and BLSQS on test (1-C) have the same rank point, i.e. (2+3)/2=2.5. Consequently, BLSQS and ORTHL, the implementations of the modified Gram-Schmidt algorithm, are the most accurate on the average, and they are superior to the other programs for testing on Jordan's test problems. This superior agrees with Jordan's test results. GIVEN performs better than LLSQF (Householder transformations).

Since the exact solution of problem (3-B) is not available, the result can not be compared by computing the number of significant digits lost. However, one can see GIVEN is almost as accurate as ORTHL, BLSQS, and LLSQF, and they agree with each other for 8∿10 digits. The squares of the norm of residual vectors, $|\vec{r}|_2^2$, for all programs have been computed as $0.190606170954\times10^{-7}$ approximately. However, the single precision arithmetic lost all digits on problem (3-B). The reason probably are that the roundoff error has occurred when A was read in and that single precision arithmetic should not be used for an ill-conditioned problem.

It is well-known that the usual iterative refinement scheme cannot improve an approximate solution unless the residual vector is computed using some extra precision [4, 5, 24]. In other words, iterative refinement is useless in pure single precision or in pure double precision.

TABLE XIV

THE RANK POINT OF SIGNIFICANT DIGITS LOST
IN DOUBLE PRECISION ARITHMETIC

	Rank Point								Average Point
	(1-A)	(1-B)	(1-C)	(1-D)	(1-E)	(2-A)	(2-B)	(3-A)	
GIVEN	3	2	2.5	3	2.5	2	1	1.5	2.1875
BLSQS	4	4	2.5	2	2.5	3	4	4	3.25
LLSQF	1	1	1	1	1	1	2	3	1.375
ORTHL	2	3	4	4	4	4	3	1.5	3.1875

TABLE XV

THE RANK POINT OF SIGNIFICANT DIGITS LOST
IN SINGLE PRECISION ARITHMETIC

	Rank Point				Average Point
	(1-A)	(2-A)	(2-B)	(3-A)	
GIVEN	4	2	1	2.5	2.375
BLSQS	2	3	4	2.5	2.875
LLSQF	3	1	3	1	2.0
ORTHL	1	4	2	4	2.75

This is true for BLSQS. Surprisingly, the solutions obtained by ORTHL are improved after iterative refinement as shown in Table XVI to Table XVIII, and its final solutions are as accurate as that of BLSQS. Further, the results of ORTHL without refinement are just the same as the results computed by the original version of ORTHOLIN2 which is listed in Appendix J. (Hence the modifications contained in ORTHL have not ruined the behavior of ORTHOLIN2.) This is very unusual, and the author does not have enough time to find out what has happened in ORTHOLIN2/ORTHL. The reason probably is that ORTHL has done the decomposition and/or back substitution in a form that is less stable in some respect than the method used in BLSQS. Users may use BLSQS [or ORTHL with iterative refinement] to get the best solution in pure single/double precision. Otherwise, one should work on ORTHL further until ORTHL is accurate as BLSQS.

TABLE XVI

COMPARISON OF ORTHL WITH AND WITHOUT ITERATIVE REFINEMENT FOR PROBLEMS (1-A) to (1-E) IN DOUBLE PRECISION ARITHMETIC

		(1-A)	(1-B)	(1-C)	(1-D)	(1-E)
With Iterative Refinement	x_1	1.00000000000176	1.00000000587991	1.00000001765005	1.00000007059677	1.00000070598581
	x_2	0.500000000000407	0.500000001975064	0.500000005928552	0.500000023713164	0.500000237140421
	x_3	0.333333333333455	0.333333334182570	0.333333335882485	0.333333343529528	0.333333435299796
	x_4	0.250000000000036	0.250000000372285	0.250000001117476	0.250000004469735	0.250000044699564
	x_5	0.200000000000009	0.200000000132561	0.200000000397901	0.200000001591545	0.200000015916301
Without Iterative Refinement	x_1	0.99999597	0.99999598	0.99999598	0.99999604	0.99999668
	x_2	0.49999866	0.49999866	0.49999866	0.49999868	0.49999889
	x_3	0.33333276	0.33333276	0.33333276	0.33333277	0.33333286
	x_4	0.24999975	0.24999975	0.24999975	0.24999975	0.24999979
	x_5	0.19999991	0.19999991	0.19999991	0.19999991	0.19999993

TABLE XVII

COMPARISON OF ORTHL WITH AND WITHOUT ITERATIVE REFINEMENT FOR PROBLEMS (2-A) to (-B) IN DOUBLE PRECISION ARITHMETIC

		(2-A)	(2-B)	(3-A)	(3-B)
With Iterative Refinement	x_1	1.000000000000000	1.000000000000000	1.000000000000649	-74.9157930845429
	x_2	0.9999999999999995	0.9999999999999996	1.999999999999441	100.68165616-4514
	x_3	1.000000000000006	1.000000000000002	-1.000000000002500	-79.80442264221179
	x_4	0.9999999999999682	0.9999999999999974	3.000000000016684	92.81699666326094
	x_5	1.000000000000073	1.000000000000001	-3.999999999948234	-80.05289262632264
	x_6	0.9999999999999191		-0.660476×10^{-10}	
	x_7	1.000000000000033			
Without Iterative Refinement	x_1	1.000000000009784	0.9999999999950065	1.00000030252136	-74.91746587148
	x_2	0.999999999628141	1.000000000092234	1.99999974265977	100.68397894992
	x_3	1.000000003558426	0.9999999999960641	-1.00000115746025	-79.80626604861
	x_4	0.999999986149639	1.000000000588753	3.00000773111154	92.81914145241
	x_5	1.000000025465319	0.9999999997144280	-3.99997602508280	-80.05484174036
	x_6	0.999999977930424		-0.305937×10^{-4}	
	x_7	1.000000007264618			

TABLE XVIII

COMPARISON OF ORTHL WITH AND WITHOUT ITERATIVE REFINEMENT IN SINGLE PRECISION ARITHMETIC

		(1-A)	(3-A)	(3-B)
With Iterative Refinement	x_1	5.201400	1.001654	$.1235\ 10^7$
	x_2	2.017883	1.998547	$-.1713\ 10^7$
	x_3	1.012878	-1.006414	$.1362\ 10^7$
	x_4	0.555840	3.042818	$-.1589\ 10^7$
	x_5	0.310946	-3.866953	$.1442\ 10^7$
	x_6		-0.169660	
Without Iterative Refinement	x_1	14085.04	$.1120\times10^4$	$.7652\times10^6$
	x_2	-4696.80	$-.9502\times10^3$	$-.1621\times10^7$
	x_3	2011.61	$.4283\times10^4$	$.8434\times10^6$
	x_4	-879.59	$.2860\times10^5$	$-.9822\times10^6$
	x_5	-312.56	$.8869\times10^5$	$.8922\times10^6$
	x_6		$-.1132\times10^6$	

CHAPTER VII

CONCLUSIONS AND RECOMMENDATIONS

From the test results and algorithms discussed in the previous chapters, the following conclusions thus can be derived.

1. Gentleman's modification of Givens transformations has the following advantages which other methods do no have.

   a. Since it processes the design matrix A one row at a time, the storage for the whole design matrix A is not necessary.

   b. Since the design matrices are often sparse, the number of operations required is much smaller in these cases. The reason is that zeros are exploited in Givens transformations.

   c. The effect of new rows is easy to include by taking the advantage of the triangularized structure already present. This is important since the need for updating regression results arises frequently. When data are obtained sequentially, it may be undesirable or impossible to wait for all the data before obtaining some regression results.

   d. Givens transformations can introduce each new row with arbitrary positive or negative weight. Therefore solving weighted least squares problems or deleting rows from a triangularized design matrix is easy although the later can be unstable.

2. From the test results, orthogonal decomposition methods can accurately solve moderately ill-conditioned linear least squares problems in double precision.

3. The method based on Givens transformations is nearly as accurate as the method based on the modified Gram-Schmidt algorithm with iterative refinement, while the modified Gram-Schmidt algorithm obtains the most accurate results.

4. The computed results of Householder transformations method with iterative refinement is a little less accurate than the results of Givens transformations.

5. The performance of each orthogonal decomposition method is getting worse when the residual vector grows larger as in problems (1-A) to (1-E). The number of significant digits lost is greater than the digits lost of a very ill-conditioned problem as (3-A).

6. If mixed precision is available for the modified Gram-Schmidt algorithm and Householder transformations, they should be much more accurate than using pure single precision arithmetic.

7. One must use double precision for ill-conditioned problems and use extra precision for iterative refinement.

For further study, the following recommendations might be a guideline.

1. Deletion of rows from a regression is inherently a numerically unstable process, and if subroutine INCLUD is used with negative weights to do this, then some code should be inserted to detect the instability and restart the decomposition if necessary.

2. If the accuracy obtained by using Givens transformations is not adequate, an iterative improvement can be used, but the storage

required will be increased.

3. Large sparse test problems may be tested to see how much the time is reduced by Givens transformations compared to the time required for large dense problems.

4. Many variations of algorithmic and programming details are possible in implementing Householder transformations, the modified Gram-Schmidt algorithm, or Givens transformations. Tradeoffs possibly involve execution time, accuracy, resistance to underflow and overflow, storage requirements, complexity of code, taking advantage of sparsity of nonzero elements, programming language, portability, etc.

SELECTED BIBLIOGRAPHY

(1) Bauer, F. L., "Elimination with Weighted Row Combinations for Solving Linear Equations and Least Squares Problems." Numerische Mathematik, Vol. 7 (1965), 338-352.

(2) Bjorck, Ake, "Solving Linear Least Squares Problems by Gram-Schmidt Orthogonalization." Nordisk Tidskrift for Informationsbehandling (BIT), Vol. 7 (1967), 1-21.

(3) Bjorck, Ake, "Iterative Refinement of Linear Least Squares Solutions I." BIT, Vol. 7 (1967), 257-278.

(4) Bjorck, Ake, "Iterative Refinement of Linear Least Squares Solutions II." BIT, Vol. 8 (1968), 8-30.

(5) Bjorck, Ake and Gene H. Golub, "Iterative Refinement of Linear Least Square Solutions by Householder Transformation." BIT, Vol. 7 (1967), 322-337.

(6) Bolliger, R. E., Computer Program, Department of Computing and Information Sciences, Oklahoma State University, Stillwater, Oklahoma.

(7) Businger, Peter A. and Gene H. Golub, "Linear Least Squares Solutions by Householder Transformations." Numerische Mathematik, Vol. 7 (1965), 269-276.

(8) Chambers, P. A., "Regression Updating." J. of Amer. Statist. Ass., Vol. 66 (1971), 744-748.

(9) Chandler, John P., Computer Program, Department of Computing and Information Sciences, Oklahoma State University, Stillwater, Oklahoma.

(10) Clayton, D. G., "Gram-Schmidt Orthogonalization." Applied Statistics, Vol. 20 (1971), 335-338.

(11) Fowlkes, E. B., "Some Operators for ANOVA Calculations." Technometrics, Vol. 11 (1969), 511-526.

(12) Gentleman, W. Morven, "Least Squares Computations by Givens Transformations without Square Roots." J. Inst. Maths Applics, Vol. 12 (1973), 329-336.

(13) Gentleman, W. Morven, "Basic Procedures for Large Sparse or Weighted Linear Least Squares Problems." Applied Statistics, Vol. 23 (1974), 448-454.

(14) Golub, Gene H., "Numerical Methods for Solving Linear Least Squares Problems." Numerische Mathematik, Vol. 7 (1965), 206-216.

(15) Golub, Gene H., "Matrix Decompositions and Statistical Calculations." Statistical Computation, R. C. Milton and J. A. Nelder, ed., Academic Press, New York, 1969.

(16) Golub, Gene H. and J. H. Wilkinson, "Note on the Iterative Refinement of Least Squares Solution." Numerische Mathematik, Vol. 9 (1966), 139-148.

(17) Herndon, J. R., "Algorithm 50." Communications of ACM, Vol. 4 (1961), 179.

(18) Householder, A. S., Principles of Numerical Analysis, McGraw-Hill, New York, 1953.

(19) Householder, A. S., "Unitary Triangularization of a Nonsymmetric Matrix." J. Assoc. Comput. Mach., Vol. 5 (1958), 339-342.

(20) IMSL Library, Computer Program, IMSL, Inc., Houston, Texas.

(21) Jordan, T. L., "Experiments of Error Growth Associated with Some Linear Least-Squares Problems." Mathematics of Computation, Vol. 22 (1968), 579-588.

(22) Lawson, Charles L. and Richard J. Hanson, Solving Least Squares Problems. Prentice-Hall, Englewood Cliffs, N.J., 1974.

(23) Longley, J. W., "An Appraisal of Least Squares Program for the Electronic Computer from the Point of View of User." J. Amer. Statist. Ass., Vol. 62, (1967), 819-841.

(24) Martin, R. S., G. Peters, and J. H. Wilkinson, "Symmetric Decomposition of a Positive Definite Matrix." Numerische Mathematik, Vol. 7 (1965), 362-383.

(25) Osborne, E. E., "On Least Squares Solutions of Linear Equation." J. Assoc. Comput. Mach., Vol. 8 (1961), 628-636.

(26) Rice, John R., "Experiments on Gram-Schmidt Orthogonalization." Math. Comp., Vol. 20 (1966), 325-328.

(27) Stewart, G. W., Introduction to Matrix Computations. Academic Press, Inc., New York, 1973.

(28) Walsh, Philip J., "Algorithm 127." Communications of ACM, Vol. 5 (1962), 511-513.

(29) Wampler, Roy H., "An Evaluation of Linear Least Squares Computer Programs." Nat. Bur. of Standards J. Res., Ser. B. Math. Sci., 73 (1969), 59-90.

(30) Wampler, Roy H., "A Report on the Accuracy of Some Widely Used Least Squares Computer Programs." J. Amer. Statist. Ass., Vol. 65 (1970), 549-565.

(31) Wilkinson, J. H., "Error Analysis of Transformations Based on the Use of Matrices of the Form $I-2ww^H$." Error in Digital Computation, L. B. Rall ed., Wiley, New York, Vol 2 (1965), 77-101.

(32) Wilkinson, J. H., The Algebraic Eigenvalue Problem, Oxford Clarendon Press, 1965.

(33) Wilkinson, J. H. and C. Reinsch, "Linear Algebra." Handbook for Automatic Computation, F. L. Bauer, ed., Vol. II, Sprint-Verlag, New York, 1971.

APPENDIX A

PROGRAM LISTING OF GIVEN

```
C*************************************************************************
C                                                                        *
C                                                                        *
C                    S U B R O U T I N E - G I V E N                     *
C                                                                        *
C                                                                        *
C       I. PURPOSES:                                                     *
C                                                                        *
C             THIS SUBROUTINE READS DATA VIA INPUT DEVICE.  THEN,        *
C          IT CALLS SUBROUTINE INCLUD, CONF, SSCDOM, AND REGRES TO       *
C          SOLVE LINEAR LEAST SQUARES PROBLEMS BY IMPLEMENTING THE       *
C          ORTHOGONAL DECOMPOSITION METHOD BASED ON GIVENS               *
C          TRANSFORMATIONS.  FINALLY, IT PRODUCES THE LEAST SQUARES      *
C          SOLUTION VIA OUTPUT DEVICE.                                   *
C                                                                        *
C      II.  SYMBOL LEGEND:                                               *
C                                                                        *
C             NCOL    - NUMBER OF COLUMNS IN DESIGN MATRIX A             *
C             NR      - DIMENSION OF THE ARRAY RBAR;                     *
C                         NR = NCOL*(NCOL+1)/2                           *
C             TOL1    - TOLERANCE FOR DETECTING RANK DEFICIENCIES        *
C             TOL2    - TOLERANCE FOR IDENTIFYING THE CONFOUNDED         *
C                         VARIABLES                                      *
C             ITYPE   - INPUT OPTION INDICATOR                           *
C                         ITYPE=1 FOR NORMAL DESIGN MATRIX               *
C                         ITYPE=2 FOR SPARSE DESIGN MATRIX               *
C                         (NOTE: INPUT DATA SEQUENCES ARE DIFFERENT)     *
C             AROW    - ONE ROW OF THE DESIGN MATRIX A TO BE             *
C                         PROCESSED CURRENTLY                            *
C             BROW    - THE CURRENT ELEMENTS OF RIGHT HAND SIDE B        *
C             WEIGHT  - WEIGHT OF EACH ROW OF A                          *
C             NZERO   - COLUMN INDEX OF THE NONZERO ELEMENT IN THE       *
C                         CURRENT ROW                                    *
C             D       - THE DIAGONAL SCALING MATRIX                      *
C             RBAR    - THE SUPERDIAGONAL ELEMENTS OF R, STORED          *
C                         SEQUENTIALLY BY ROWS                           *
C             TBAR    - THETA BAR, WHERE D**2*TBAR IS THE VECTOR OF      *
C                         ORTHOGONAL COEFFICIENTS                        *
C             SSERR   - THE SUM OF SQUARES ERROR                         *
C             J       - SEE DESCRIPTION IN SUBROUTINE CONF               *
C                                                                        *
C     III.  INPUT PARAMETERS:                                            *
C                                                                        *
C             NCOL, NR, ITYPE.                                           *
C                                                                        *
C      IV.  DATA READ VIA INPUT DEVICE:                                  *
C                                                                        *
C             WEIGHT, AROW, BROW, NZERO.                                 *
C                                                                        *
C       V.  INPUT DATA SEQUENCES:                                        *
C                                                                        *
C             ITYPE=1:                                                   *
C                                                                        *
C                  CARD 1:  WEIGHT OF THE FIRST ROW OF A                 *
C                  CARD 2:  (AROW(I),I=1,NCOL),BROW OF THE FIRST         *
C                    .            ROW OF (A|B)                          *
C                    .                                                   *
```

```
C                    .    (REPEAT CARD 1 AND 2 FOR THE NEXT                *
C                    .       OBSERVATION UNTIL WEIGHT=0)                   *
C                    .                                                     *
C                                                                          *
C                                                                          *
C              ITYPE=2:                                                    *
C                                                                          *
C                    CARD 1:  WEIGHT OF THE FIRST ROW OF A                 *
C                    CARD 2:  BROW                                         *
C                    CARD 3:  NZERO                                        *
C                    CARD 4:  AROW(NZERO) OF THE 1ST ROW OF A              *
C                     .                                                    *
C                     .    (REPEAT CARD 3 AND 4 FOR THE NEXT               *
C                     .       NONZERO AROW(NZERO) UNTIL NZERO=0)           *
C                     .                                                    *
C                     .                                                    *
C                    (REPEAT FROM CARD 1 FOR THE NEXT ROW OF A             *
C                     . UNTIL WEIGHT=0)                                    *
C                     .                                                    *
C                     .                                                    *
C                    NOTE:  1) WEIGHT=0 MEANS END OF INPUT DATA            *
C                           2) NZERO=0 INDICATES END OF EACH ROW           *
C                           3) SET ALL WEIGHT=1 FOR UNWEIGHTED             *
C                                PROBLEMS.                                 *
C                                                                          *
C       VI.  WORK SPACE:                                                   *
C                                                                          *
C                 D(NCOL), RBAR(NR), TBAR(NCOL), AROW(NCOL).               *
C                                                                          *
C      VII.  REFERENCES:                                                   *
C                                                                          *
C                 GENTLEMAN, W. M. "LEAST SQUARES COMPUTATIONS BY GIVENS   *
C                     TRANSFORMATIONS WITHOUT SQUARE ROOTS." J. INST.      *
C                     MATHS. APPLICS, 12 (1973), PP.329-336.               *
C                                                                          *
C                 GENTLEMAN, W. M. "BASIC PROCEDURES FOR LARGE SPARSE OR   *
C                     WEIGHTED LINEAR LEAST SQUARES PROBLEMS." APPLIED     *
C                     STATISTICS, 23 (1974), PP. 448-454.                  *
C                                                                          *
C     VIII.  AUTHOR:                                                       *
C                                                                          *
C                 HSIAOLAN W. LOH, COMPUTING AND INFORMATION SCIENCES,     *
C                     OKLAHOMA STATE UNIVERSITY, STILLWATER, OKLAHOMA.     *
C                                                                          *
C**************************************************************************
      SUBROUTINE GIVEN (NCOL,NR,ITYPE,AROW,D,TBAR,RBAR)
      IMPLICIT REAL*8 (A-H,O,R-Z)
      DIMENSION AROW(NCOL),D(NCOL),TBAR(NCOL),RBAR(NR)
C
C-----SET CONSTANTS.
C
      IN=5
      LP=6
      TOL1=1.D-16
      TOL2=1.D-8
      ZERO=0.
      ONE=1.
C
C-----INITIALIZATION.
C
      N = 0
      SSERR=ZERO
      DO 10 K=1,NCOL
         TBAR(K)=ZERO
         D(K)=ZERI
   10    CONTINUE
```

```
      DO 20 K=1,NR
         RBAR(K)=ZERO
   20    CONTINUE
C
C-----PRINT HEADING.
C
      WRITE (LP,30)
   30 FORMAT (//'    TEST DATA  ==>  (A:B)'//)
C
C-----INPUT WEIGHT FOR THE CURRENT ROW.
C
   40 READ (IN,50) WEIGHT
   50 FORMAT (F14.7)
      IF (ITYPE.EQ.1) GOTO 122
C
C-----INPUT DATA FOR SPARSE MATRIX.
C
      IF (WEIGHT.EQ.ZERO) GOTO 150
      IF (WEIGHT.GT.ZERO) GOTO 60
      N=N-1
      GOTO 70
   60 N=N+1
   70 READ (IN,80) BROW
   80 FORMAT (E12.8)
      DO 90 K=1,NCOL
         AROW(K)=ZERO
   90    CONTINUE
  100 READ (IN,110) NZERO
  110 FORMAT (I2)
      IF ((NZERO.LE.0).OR.(NZERO.GT.NCOL)) GOTO 130
      READ (IN,120) AROW(NZERO)
  120 FORMAT (E12.8)
      GOTO 100
C
C-----INPUT DATA FOR NORMAL MATRIX.
C
  122 IF (WEIGHT.EQ.ZERO) GOTO 150
      IF (WEIGHT.GT.ZERO) GOTO 124
      N=N-1
      GOTO 126
  124 N=N+1
  126 READ (IN,128) (AROW(I),I=1,NCOL),BROW
  128 FORMAT (6E12.8)
C
C-----PRINT CURRENT ROW.
C
  130 WRITE (LP,140) (AROW(I),I=1,NCOL),BROW
  140 FORMAT (6E16.8)
C
C-----INCLUDE THE EFFECT OF THE CURRENT ROW.
C
      CALL INCLUD (NCOL,NR,WEIGHT,AROW,BROW,D,RBAR,TBAR,SSERR)
      GOTO 40
C
C-----PRINT NUMBER OF ROWS AND DIAGONAL MATRIX.
C
  150 WRITE (LP,160) N
  160 FORMAT (//5X,I4,' OBSERVATIONS READ')
      WRITE (LP,170) (D(I),I=1,NCOL)
  170 FORMAT (////'     DIAGONAL MATRIX IS'//(6X,E25.16))
C
C-----FIND CONFOUNDED CONTRAST TO RESOLVE INDETERMINACY.
C
      NFIRST=1
      DO 220 J=1,NCOL
         IF (DABS(D(J)).GE.TOL1) GOTO 220
```

```
C
C          CONFOUNDING DISCOVERED
C
           IF (NFIRST.NE.1) GOTO 180
           NFIRST=0
  180      CALL CONF (NCOL,NR,J,RBAR,AROW)
           WRITE (LP,190) (AROW(I),I=1,NCOL)
  190      FORMAT (////'     CONFOUNDED CONTRASTS'//(6X,E25.16))
C
C          CHOOSE RESOLVING CONSTRAINT
C
           M=J-1
           DO 200 K=1,M
              IF (DABS(AROW(K)).LE.TOL2) GOTO 200
              AROW(K)=ZERO
              GOTO 210
  200         CONTINUE
  210      WEIGHT=ONE
           BROW=ZERO
           CALL INCLUD (NCOL,NR,WEIGHT,AROW,BROW,D,RBAR,TBAR,SSERR)
  220      CONTINUE
C
C-----FIND SUM OF SQUARES DECOMPOSITION AND SUM OF SQUARES ERROR.
C
       CALL SSDCOM (NCOL,D,TBAR,AROW)
       WRITE (LP,230) (AROW(I),I=1,NCOL)
  230  FORMAT (////'     SUM OF SQUARES DECOMPOSITION'//(6X,E25.16))
       WRITE (LP,240) SSERR
  240  FORMAT (////'     SUM OF SQUARES ERROR'//6X,E25.16)
C
C-----FIND SOLUTION VECTOR.
C
       CALL REGRES (NCOL,NR,RBAR,TBAR,AROW)
       WRITE (LP,250) (AROW(I),I=1,NCOL)
  250  FORMAT (////'     REGRESSION COEFFICIENTS'//(16X,E25.16))
       RETURN
       END
```

```
C*********************************************************************
C                                                                     *
C                                                                     *
C                    S U B R O U T I N E - I N C L U D                *
C                                                                     *
C                                                                     *
C       I.  PURPOSE:                                                  *
C                                                                     *
C               THIS SUBROUTINE UPDATES D, RBAR, TBAR, AND SSERR      *
C           TO INCLUDE, WITH SPECIFIED WEIGHT, THE EFFECT OF A NEW    *
C           ROW OF A AND B.                                           *
C               FOR AN INITIAL DECOMPOSITION, D, RBAR, TBAR, AND      *
C           SSERR SHOULD BE SET TO ZERO BEFORE INCLUDING THE FIRST    *
C           ROW.                                                      *
C                                                                     *
C      II.  INPUT VARIABLES:                                          *
C                                                                     *
C              NCOL, NR, WEIGHT, AROW, BROW                           *
C              (SEE DEFINITION IN SUBROUTINE GIVEN)                   *
C                                                                     *
C     III.  OUTPUT VARIABLES:                                         *
C                                                                     *
C              D, RBAR, TBAR, SSERR                                   *
C              (SEE DEFINITION IN SUBROUTINE GIVEN)                   *
C                                                                     *
C*********************************************************************
      SUBROUTINE INCLUD (NCOL,NR,WEIGHT,AROW,BROW,D,RBAR,TBAR,SSERR)
      IMPLICIT REAL*8 (A-H,O,R-Z)
      DIMENSION AROW(NCOL),D(NCOL),TBAR(NCOL),RBAR(NR)
C
C-----SKIP UNNECESSARY TRANSFORMATIONS.  TEST ON EXACT ZEROS MUST
C     BE USED OR STABILITY CAN BE DESTROYED.
C
      DO 20 I=1,NCOL
         IF (WEIGHT.EQ.0) GOTO 30
         IF (AROW(I).EQ.0.) GOTO 20
         XI=AROW(I)
         DI=D(I)
         DPRIME=DI+WEIGHT*XI**2
         CBAR=DI/DPRIME
         SBAR=WEIGHT*XI/DPRIME
         WEIGHT=CBAR*WEIGHT
         D(I)=DPRIME
         NEXTR=(I-1)*(2*NCOL-I)/2+1
         M=I+1
         DO 10 K=M,NCOL
            IF (K.GT.NCOL)GOTO 10
            XK=AROW(K)
            AROW(K)=XK-XI*RBAR(NEXTR)
            RBAR(NEXTR)=CBAR*RBAR(NEXTR)+SBAR*XK
            NEXTR=NEXTR+1
   10       CONTINUE
         XK=BROW
         BROW=XK-XI*TBAR(I)
         TBAR(I)=CBAR*TBAR(I)+SBAR*XK
   20    CONTINUE
      SSERR=SSERR+WEIGHT*BROW**2
   30 RETURN
```

```
C*********************************************************************
C                                                                     *
C                                                                     *
C                    S U B R O U T I N E - C O N F                    *
C                                                                     *
C                                                                     *
C       I.  PURPOSE:                                                  *
C                                                                     *
C             INVOKING THIS SUBROUTINE OBTAINS THE CONTRAST WHICH     *
C           COULD NOT BE ESTIMATED IF D(J) WERE ASSUMED TO BE ZERO.   *
C           THAT IS, OBTAINS THE LINEAR COMBINATION OF THE FIRST J    *
C           COLUMNS WHICH WOULD ZE ZERO.  THIS IS OBTAINED BY SETTING *
C           THE FIRST J-1 ELEMENTS OF CONTRAST TO THE SOLUTION OF THE *
C           TRIANGULAR SYSTEM FORMED BY THE FIRST J-1 ROWS AND        *
C           COLUMNS OF RBAR WITH THE FIRST J-1 ELEMENTS OF THE JTH    *
C           COLUMN AS RIGHT HAND SIDE, SETTING THE JTH ELEMENT OF     *
C           CONTRAST TO -1, AND SETTING THE REMAINING ELEMENTS OF     *
C           CONTRAST TO ZERO.                                         *
C                                                                     *
C      II.  INPUT VARIABLES:                                          *
C                                                                     *
C              NCOL, NR, J, RBAR                                      *
C              (SEE DEFINITION IN SUBROUTINE GIVEN)                   *
C                                                                     *
C     III.  OUTPUT:                                                   *
C                                                                     *
C              CONTRA - THE COEFFICIENTS OF THE CONFOUNDED CONTRAST   *
C                       AMONG THE INDEPENDENT VARIABLES IF THE SYSTEM *
C                       IS RANK DEFICIENT                             *
C                                                                     *
C                                                                     *
C*********************************************************************
      SUBROUTINE CONF (NCOL,NR,J,RBAR,CONTRA)
      IMPLICIT REAL*8 (A-H,O,R-Z)
      DIMENSION RBAR(NR),CONTRA(NCOL)
      L=J+1
      DO 10  I=L,NCOL
         IF (I.GT.NCOL) GOTO 10
         CONTRA(I)=0.
   10    CONTINUE
      CONTRA(J) = -1.
      L = J - 1
      I = L
   20    NEXTR = (I-1) * (2*NCOL-I) / 2 + 1
         CONTRA(I) = RBAR(NEXTR+J-I-1)
         M = I + 1
         DO 30 K=M,L
            CONTRA(I) = CONTRA(I) - RBAR(NEXTR) * CONTRA(K)
            NEXTR = NEXTR + 1
   30       CONTINUE
         I = I - 1
      IF (I.GE.1) GOTO 20
      RETURN
      END
```

```
C*****************************************************************
C                                                                *
C                                                                *
C                  S U B R O U T I N E - S S D C O M             *
C                                                                *
C                                                                *
C       I.  PURPOSE:                                             *
C                                                                *
C            THIS SUBROUTINE COMPUTES THE COMPONENTS OF THE SUM   *
C          OF SQUARES DECOMPOSITION FROM D AND TBAR.             *
C                                                                *
C      II.  INPUT:                                               *
C                                                                *
C           NCOL, D, TBAR                                        *
C           (SEE DEFINITION IN SUBROUTINE GIVEN)                 *
C                                                                *
C     III.  OUTPUT:                                              *
C                                                                *
C             SS    - THE SUM OF SQUARES DECOMPOSITION,          *
C                       I.E. THE SQUARES OF THE ORTHOGONAL       *
C                       COEFFICIENTS                             *
C                                                                *
C*****************************************************************
      SUBROUTINE SSDCOM (NCOL,D,TBAR,SS)
      IMPLICIT REAL*8 (A-H,O,R-Z)
      DIMENSION D(NCOL),TBAR(NCOL),SS(NCOL)
      DO 10 I=1,NCOL
         SS(I) = D(I) * TBAR(I) ** 2
   10    CONTINUE
      RETURN
      END
```

```
C*********************************************************************
C                                                                     *
C                                                                     *
C                    S U B R O U T I N E - R E G R E S                *
C                                                                     *
C                                                                     *
C       I.  PURPOSE:                                                  *
C                                                                     *
C             THIS SUBROUTIE OBTAINS BETA BY BACKSUBSTITUTION IN      *
C           THE TRIANGULAR SYSTEM RBAR AND TBAR.                      *
C                                                                     *
C      II.  INPUT:                                                    *
C                                                                     *
C             NCOL, NR, RBAR, TBAR                                    *
C             (SEE DEFINITION IN SUBROUTINE GIVEN)                    *
C                                                                     *
C     III.  OUTPUT:                                                   *
C                                                                     *
C             BETA - THE REGRESSION COEFFICIENTS                      *
C                                                                     *
C                                                                     *
C*********************************************************************
      SUBROUTINE REGRES (NCOL,NR,RBAR,TBAR,BETA)
      IMPLICIT REAL*8 (A-H,O,R-Z)
      DIMENSION RBAR(NR),TBAR(NCOL),BETA(NCOL)
      I = NCOL
   10    BETA(I) = TBAR(I)
         NEXTR = (I-1) * (2*NCOL-I) / 2 + 1
         M = I + 1
         DO 20 K=M,NCOL
            IF (K.GT.NCOL) GOTO 20
            BETA(I) = BETA(I) - RBAR(NEXTR) * BETA(K)
            NEXTR = NEXTR + 1
   20       CONTINUE
         I = I - 1
      IF (I.GE.1) GOTO 10
      RETURN
      END
```

APPENDIX B

PROGRAM LISTING OF ORTHL

```
      SUBROUTINE ORTHL(A,LAU,NR,NC,B,X,R,LR,IREF,NTRAC,NIX,U,P,PP,D)
      IMPLICIT REAL*8 (A-H,O-Z)
C
C  ORTHL 2.2          A.N.S.I. STANDARD FORTRAN          NOVEMBER 1974
C
C  J. P. CHANDLER, COMPUTER SCIENCE DEPT., OKLAHOMA STATE UNIVERSITY
C
C  LEAST SQUARE SOLUTION OF A*X=B, WHERE -A- IS A MATRIX WITH NR ROWS
C  AND NC COLUMNS (NR.GE.NC), AND B IS A VECTOR WITH NR COMPONENTS.
C
C  F. L. BAUER, NUMERISCHE MATHEMATIK 7 (1965) 338
C
C  GIVEN A MATRIX -A- AND A VECTOR -B-, ORTHL SOLVES FOR THE UNIQUE
C  VECTOR X, IF ANY, WHICH MINIMIZES THE LENGTH OF THE VECTOR A*X-B.
C  ORTHL WILL SOLVE ANY LINEAR LEAST SQUARES FITTING PROBLEM
C  (LINEAR REGRESSION, POLYNOMIAL REGRESSION, ETC.) HAVING A UNIQUE
C  SOLUTION AND, IF STORAGE PERMITS, SHOULD ALWAYS BE USED IN
C  PREFERENCE TO SOLVING THE -NORMAL EQUATIONS- (AH*A*X=AH*B).
C  (AH DENOTES THE TRANSPOSE OF A.)
C
C  FOR A PROBLEM THAT DOES NOT HAVE A UNIQUE SOLUTION (NIX RETURNED
C  NONZERO), CONSULT....      -SOLVING LEAST SQUARES PROBLEMS- BY
C  C. L. LAWSON AND R. J. HANSON (PRENTICE-HALL 1974).
C
C  INPUT QUANTITIES.....   A,LAU,NR,NC,B,LR,IREF,NTRAC
C  OUTPUT QUANTITIES....   X,R,NIX
C  SCRATCH ARRAYS.......   U,P,PP,D
C
C     A       --  THE ARRAY CONTAINING THE INPUT MATRIX -A-
C     LAU     --  THE FIRST DIMENSION OF THE ARRAYS -A- AND -U-
C                      (NOT THE MATRICES -A- AND -U-)
C     NR      --  THE NUMBER OF ROWS IN THE MATRIX -A-
C     NC      --  THE NUMBER OF COLUMNS IN THE MATRIX -A-
C     B       --  THE ARRAY CONTAINING THE INPUT VECTOR -B-
C     X       --  THE ARRAY IN WHICH THE SOLUTION VECTOR IS RETURNED
C     R       --  RETURNS THE ERROR MATRIX (AH*A)**-1
C     LR      --  THE FIRST DIMENSION OF THE ARRAY -R-
C     IREF    --  NONZERO IF ITERATIVE REFINEMENT OF THE SOLUTION IS
C                      TO BE PERFORMED (IF IREF IS ZERO, THE ARRAYS -A-
C                      AND -U- MAY BE THE SAME ARRAY IN THE CALLING
C                      PROGRAM, AND DOUBLE PRECISION IS NOT USED)
C     NTRAC   --  = 0 FOR NORMAL OUTPUT
C                 = 1 TO PRINT OUT THE RESULT OF EACH ITERATION
C                =-1 TO OBTAIN NO OUTPUT
C     NIX     --  RETURNED NONZERO IF THE GIVEN PROBLEM WAS SINGULAR
C     U       --  SCRATCH ARRAY OF AT LEAST NR*NC LOCATIONS
C     P       --  SCRATCH ARRAY OF AT LEAST NR LOCATIONS
C     PP      --  SCRATCH ARRAY OF AT LEAST NC LOCATIONS
C     D       --  SCRATCH ARRAY OF AT LEAST NC LOCATIONS
C
C  THE FOLLOWING CHANGES HAVE BEEN MADE IN BAUER-S ORTHOLIN2 ....
C 1. THE DECOMPOSITION OF OSBORNE IS USED...
C         A=U*R INSTEAD OF BAUER-S A=U*D*R, AND R**-1 IS COMPUTED
C         INSTEAD OF R.
C 2. THE ERROR MATRIX ERR=(AH*A)**-1 IS COMPUTED, WITHOUT FORMING AH*A.
C         (THIS REQUIRES THE USE OF BOTH TRIANGLES OF THE ARRAY R.)
C 3. THE DIAGONAL MATRIX D**-1=UH*U IS SAVED (IN THE ARRAY D) IN ORDER
```

```
C          TO OBTAIN ERR WITHOUT COMPUTING ANY SQUARE ROOTS.
C
C  RELATIONS AMONG THE MATRICES IN THE DECOMPOSITION ....
C  A=U*R      UH*U=D**-1      R=D*UH*A      AH*A=RH*(D**-1)*R
C  R*X=D*UH*B
C
C  OTHER REFERENCES....
C       E. E. OSBORNE, J.S.I.A.M. 12 (1964) 300
C       JOHN R. RICE, MATHEMATICS OF COMPUTATION 20 (1966) 325
C       R. VON HOLDT, PROC. WESTERN JOINT COMPUTER CONF. (1959) 255
C       J. H. WILKINSON AND C. REINSCH, -LINEAR ALGEBRA-
C            (SPRINGER-VERLAG, 1971)
C       T. L. JORDAN, MATHEMATICS OF COMPUTATION 22 (1968) 579
C       R. H. WAMPLER, J. AM. STAT. ASSOC. 65 (1970) 549
C       J. LONGLEY, J. AM. STAT. ASSOC. 62 (1967) 819
C       G. GOLUB IN -STATISTICAL COMPUTATION-, ED. R. C. MILTON AND
C            J. A. NELDER (ACADEMIC PRESS, 1969)
C       A. BJORCK, BIT 7 (1967) 1
C       A. BJORCK, BIT 7 (1967) 257
C       A. BJORCK, BIT 8 (1968) 8
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      DOUBLE PRECISION DS,DT,DU
C
      DIMENSION B(1),X(1),P(1),PP(1),D(1)
      DIMENSION A(LAU,NC),R(LR,NC),U(LAU,NC)
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      IDEBUG=0
      KW=6
      RZERO=0.
      RUNIT=1.
      REFAC=.25
      RQUAR=.25
C
      EPS2=RZERO
      NIX=1
      IF (IDEBUG.EQ.0) GOTO 1000
      WRITE(KW,100)
  100 FORMAT(1H1,'INITIAL (A:B):')
      DO 200 I=1,LAU
  200 WRITE(KW,300) (A(I,J),J=1,NC),B(I)
  300 FORMAT(10X,7E14.4)
C                              SET U EQUAL TO A.
 1000 DO 1010 J=1,NR
        DO 1010 K=1,NC
 1010     U(J,K)=A(J,K)
      IF (IDEBUG.EQ.0) GOTO 1020
      WRITE(KW,1012)
 1012 FORMAT(/' INITIAL U:')
      DO 1014 I=1,NR
 1014 WRITE(KW,1016) (U(I,J),J=1,NC)
 1016 FORMAT(5E26.18)
C                              INITIALIZE R TO THE UNIT MATRIX.
 1020 DO 1040 J=1,NC
        DO 1030 K=1,NC
 1030     R(J,K)=RZERO
 1040   R(J,J)=RUNIT
      IF (IDEBUG.EQ.0) GOTO 1048
      WRITE(KW,1042)
 1042 FORMAT(/' INITIAL R:')
      DO 1044 I=1,NC
 1044 WRITE(KW,1016) (R(I,J),J=1,NC)
C
```

```
C  DECOMPOSE -A- INTO A=U*R , WHERE U IS AN NR BY NC MATRIX WITH
C  ORTHOGONAL COLUMNS AND R IS AN NC BY NC UNIT UPPER TRIANGULAR MATRIX.
C  THE MATRIX R**-1 IS COMPUTED AND STORED IN THE ARRAY R.
C  THE MODIFIED GRAM-SCHMIDT METHOD, WHICH IS STABLE, IS USED TO
C  ORTHOGONALIZE THE COLUMNS OF U.
C
 1048 DO 1130 K=1,NC
        KMU=K-1
        IF (IDEBUG.EQ.1) WRITE(KW,1049) K, KMU
 1049   FORMAT(/' K, KMU =',2I5)
        IF(KMU)1460,1100,1050
 1050   DO 1090 J=1,KMU
          S=RZERO
          DO 1060 L=1,NR
            IF(IDEBUG.EQ.1) WRITE(KW,1052) L,J,K,U(L,J),U(L,K)
 1052       FORMAT(/' #1   L,J,K,U(L,J),U(L,K) =',3I5,2E26.18)
 1060       S=S+U(L,J)*U(L,K)
          S=S/D(J)
          IF (IDEBUG.EQ.1) WRITE(KW,1062) S
 1062     FORMAT(/' #1   S =',E26.18)
          DO 1070 L=1,NR
 1070       U(L,K)=U(L,K)-S*U(L,J)
C
C                              PERFORM THE SAME COLUMN OPERATION ON R.
          DO 1080 L=1,NC
 1080       R(L,K)=R(L,K)-S*R(L,J)
 1090     CONTINUE
          IF (IDEBUG.EQ.0) GOTO 1100
          WRITE(KW,1091)
 1091     FORMAT(/' #1   U =')
          DO 1092 I=1,NR
 1092     WRITE(KW,1016) (U(I,J),J=1,NC)
          WRITE(KW,1094)
 1094     FORMAT(/' #1   R =')
          DO 1095 I=1,NC
 1095     WRITE(KW,1016) (R(I,J),J=1,NC)
C                              COMPUTE THE SQUARED LENGTH OF COLUMN K.
 1100   S=RZERO
        DO 1110 L=1,NR
 1110     S=S+U(L,K)**2
        IF (IDEBUG.EQ.1) WRITE(KW,1112) S
 1112   FORMAT(/' #2   S =',E26.18)
C                              IF THE LENGTH IS ZERO, THE PROBLEM DOES
C                                   NOT HAVE A UNIQUE SOLUTION.
        IF(S)1460,1460,1120
 1120   D(K)=S
 1130   CONTINUE
C                              FORM D*UH*B IN X.
      DO 1150 J=1,NC
        S=RZERO
        DO 1140 K=1,NR
 1140     S=S+U(K,J)*B(K)
        X(J)=S/D(J)
        IF (IDEBUG.EQ.1) WRITE(KW,1142) J,S,D(J),X(J)
 1142   FORMAT(/' (X=D*UH*B)           J,S,D(J),X(J) =',I5,3E26.18)
 1150   CONTINUE
C                              COMPUTE X=(R**-1)*D*UH*B .
      DO 1170 J=1,NC
        S=RZERO
        DO 1160 K=J,NC
 1160     S=S+R(J,K)*X(K)
        X(J)=S
        IF (IDEBUG.EQ.1) WRITE(KW,1162) J,S,X(J)
 1162   FORMAT(/' (X=(R**-1)*D*UH*B)   J,S,X(J) =',I5,2E26.18)
 1170   CONTINUE
C
```

```
      IF(IREF)1180,1390,1180
C
C  ITERATE THE SOLUTION.
C  COMPUTE THE RESIDUAL VECTOR AND STORE IT IN P.
C
 1180 SDOLD=RZERO
 1190 DO 1210 J=1,NR
        DS=B(J)
        DO 1200 K=1,NC
          DT=A(J,K)
          DU=X(K)
 1200     DS=DS-DT*DU
 1210   P(J)=DS
      IF (IDEBUG.EQ.1) WRITE(KW,1212) (P(J),J=1,NR)
 1212 FORMAT(/' REDIDUAL R ='/(5E26.18))
 1218 IF(NTRAC)1240,1240,1220
 1220 WRITE(KW,1230)(X(K),K=1,NC)
 1230 FORMAT(/16H ORTHL .... X =     /(20X,E26.18))
C
C                              COMPUTE PP=D*UH*P .
 1240 DO 1260 J=1,NC
        S=RZERO
        DO 1250 K=1,NR
 1250     S=S+U(K,J)*P(K)
 1260   PP(J)=S/D(J)
      IF(IDEBUG.EQ.1) WRITE(KW,1262) (PP(J),J=1,NC)
 1262 FORMAT(/' PP=D*UH*P ='/5E26.18)
C                              COMPUTE DELTA X = (R**-1)*PP .
      SXOLD=RZERO
      SDX=RZERO
      SDIFF=RZERO
      DO 1280 J=1,NC
        S=RZERO
        DO 1270 K=J,NC
 1270     S=S+R(J,K)*PP(K)
        SXOLD=SXOLD+X(J)**2
        SDX=SDX+S*S
        XSAVE=X(J)
        X(J)=X(J)+S
        SDIFF=SDIFF+(X(J)-XSAVE)**2
        IF (IDEBUG.EQ.1) WRITE(KW,1272) J,S,SXOLD,SDX,XSAVE,X(J),SDIFF
 1272   FORMAT(/' J=',I5,/' S,SXOLD,SDX,XSAVE,X(J),SDIFF ='/5E26.18)
 1280   CONTINUE
C
C                              TEST FOR CONVERGENCE.
      IF(NTRAC)1310,1310,1290
 1290 WRITE(KW,1300)SXOLD,SDX,SDIFF,SDOLD
 1300 FORMAT(/39H ORTHL.      SXOLD, SDX, SDIFF, SDOLD =     ,4E18.5)
C
C                              CHECK (DELTA(N) X) VS. 0.5*X.
 1310 IF(SDX-RQUAR*SXOLD)1340,1320,1320
 1320 WRITE(KW,1330)SDX,SXOLD
 1330 FORMAT(/43H POOR CONVERGENCE IN ORTHL.   SDX, SXOLD = ,2E15.5/1H )
C
C                              CHECK (DELTA(N) X) VS. EPS*X.
 1340 IF(SDIFF-EPS2*SXOLD)1390,1350,1350
 1350 IF(SDIFF)1390,1390,1360
 1360 IF(SDOLD)1380,1380,1370
C                              CHECK (DELTA(N) X) VS.
C                                   SQRT(REFAC)*(DELTA(N-1) X).
 1370 IF(SDIFF-REFAC*SDOLD)1380,1390,1390
 1380 SDOLD=SDIFF
      GO TO 1190
 1390 IF(NTRAC)1410,1410,1400
 1400 WRITE(KW,1230)(X(K),K=1,NC)
C
```

```
C  COMPUTE THE ERROR MATRIX, (R**-1)*D*(R**-1)H , AND STORE IT IN R.
C  COMPUTE THE LOWER TRIANGLE FIRST, THEN SYMMETRIZE THE MATRIX.
C  UP TO THIS POINT THE LOWER TRIANGLE OF THE ARRAY R HAS NOT BEEN USED.
C
 1410 DO 1430 J=1,NC
        NCPJ=NC+J
        DO 1430 KK=J,NC
          K=NCPJ-KK
          S=RZERO
          DO 1420 L=K,NC
 1420       S=S+R(J,L)*R(K,L)/D(L)
 1430     R(K,J)=S
      DO 1440 J=1,NC
        DO 1440 K=J,NC
 1440     R(J,K)=R(K,J)
      IF (IDEBUG.EQ.0) GOTO 1448
      WRITE(KW,1442)
 1442 FORMAT(/' ERROR MATRIX R =')
      DO 1444 I=1,NC
 1444   WRITE(KW,1016) (R(I,J),J=1,NC)
C                              ORTHL FINISHED SUCCESSFULLY.  RETURN.
 1448 NIX=0
 1450 RETURN
C
 1460 WRITE(KW,1470)NIX
 1470 FORMAT( //// 21H ORTHL FAILED (NIX = ,I1, 2H). ,5X,
     *     24H THE SYSTEM IS SINGULAR.  // 1H  )
      GO TO 1450
C
C  END ORTHL.
      END
```

APPENDIX C

PROGRAM LISTING OF BLSQS

```
      SUBROUTINE BLSQS (M,N,MPN,NPU,NRHS,M1,N1,ISING,IFAIL,ETA,TOL,
     *             A,LA,B,LB,X,LX,RES,LRES,QR,LQR,XV,RESV,IPIV,D,Y,
     *             F,G,XMY1,XMY2)
      IMPLICIT REAL*8(A-H,O-Z)
C
C
C.....AUTHOR.  R E BOLLIGER.
C              OKLAHOMA STATE UNIVERSITY.
C
C.....GENERAL DESCRIPTION.
C
C     THIS FORTRAN SUBROUTINE SOLVES THE SYSTEM OF LINEAR EQUATIONS,
C     A * X = B FOR THE BEST LEAST SQUARES SOLUTION.  THIS VERSION
C     IS A TRANSLATION OF SEVERAL ALGOL PROGRAMS BY BJORK (1).  THE
C     MATRIX -A- CONTAINS THE GIVEN SYSTEM OF M LINEAR EQUATIONS IN
C     N UNKNOWNS, WHERE M IS GREATER THAN OR EQUAL TO N AND THE FIRST
C     M1 ARE TO BE STRICTLY SATISFIED.  FOR THE -NRHS- RIGHT HAND
C     SIDES GIVEN IN THE MATRIX -B-, THE BEST LEAST SQUARES SOLUTION
C     TO THE APPROXIMATING SYSTEM IS COMPUTED AND STORED IN THE ARRAY
C     -X-.  THE CORRESPONDING RESIDUALS ARE STORED IN THE ARRAY -RES-.
C     THE CHOICE OF THE RANK N1 OF THE APPROXIMATING SYSTEM DEPENDS
C     ON THE PARAMETER -TOL-.
C
C.....RESTRICTIONS.
C     THE VECTOR -RESV- MUST BE DECLARED TO BE DOUBLE PRECISION,
C     OTHERWISE THE RESULTS OF THIS PROGRAM ARE MEANINGLESS.
C
C.....DIMENSION LIMITATIONS.
C     GIVEN M, N AND -NRHS-, THE CALLING PROGRAM MUST PROVIDE THE
C     FOLLOWING MINIMUM STORAGE LOCATIONS...
C
C     ARRAY NAME    MINIMUM REQUIRED DIMENSION(S)
C     ----------    -----------------------------
C     A             (M,N+1)
C     B             (M,NRHS)
C     X             (N,NRHS)
C     RES           (M,NRHS)
C     QR            (M+N,N)
C     IPIV          (N)
C     XV            (N+1)
C     RESV          (M)
C     D             (N)
C     F             (M+N)
C     G             (M+N)
C     Y             (N)
C     XMY1          (M)
C     XMY2          (N)
C     THIS MEANS THAT AT LEAST
C          2*M+7*N+N**2+2M*N+2*NRHS*(M+2N)+1
C     STORAGE LOCATIONS MUST BE RESERVED.
C
C.....SPECIAL MACHINE REQUIREMENTS.
C     THE PARAMETERS -ETA-, -TOL-, -FOUR- AND -SIXFO- ARE
C     MACHINE DEPENDENT.  THE BEST VALUES OF -TOL-,
C     -FOUR- AND-SIXFO- FOR THE IBM 360 ARE UN-
C     DETERMINED AT THIS TIME.
C
```

```
C.....SUBROUTINES CALLED.
C     THIS PROGRAM CALLS THE SUBROUTINES
C     -SOLVE-, -DECOM-, AND -ACSOL-.  EACH OF
C     THESE PROGRAMS ARE CONTAINED IN THE
C     BLSQS PACKAGE.
C
C.....CALLING SEQUENCE.
C
C     CALL BLSQS(M,N,MPN,NPU,NRHS,M1,N1,ISING,IFAIL,ETA,TOL,
C    *           A,LA,B,LB,X,LX,RES,LRES,QR,LQR,XV,RESV,IPIV,D,Y,
C               F,G,XMY1,XMY2)
C
C.....PARAMETER DESCRIPTION
C     NAME          MEANING OR USE
C     ----          --------------
C     M             NUMBER OF EQUATIONS TO BE SOLVED
C     N             NUMBER OF UNKNOWNS
C     MPN           EQUAL TO N + M
C     NPU           EQUAL TO N + 1
C     NRHS          NUMBER OF RIGHT HAND SIDES
C     M1            NUMBER OF EQUATIONS TO BE STRICTLY SATISFIED
C     N1            RANK OF THE A MATRIX (DETERMINED BY TOL)
C     ISING         FAILURE EXIT PARAMETER IN DECOM
C     IFAIL         FAILURE EXIT PARAMETER IN ACSOL
C     ETA           RELATIVE MACHINE TOLERANCE
C     TOL           PARAMETER USED TO DETERMINE RANK OF A
C     A             ARRAY CONTAINING SYSTEM TO BE SOLVED
C     LA            SEE (**) BELOW
C     B             ARRAY OF RIGHT HAND SIDES
C     LB            SEE (**) BELOW
C     X             ARRAY OF SOLUTION VECTORS
C     LX            SEE (**) BELOW
C     RES           ARRAY OF RESIDUAL VECTORS
C     LRES          SEE (**) BELOW
C     QR            ARRAY CONTAINING DECOMPOSITION OF A
C     LQR           SEE (**) BELOW
C     XV            A SOLUTION VECTOR
C     RESV          A RESIDUAL VECTOR
C
C     THE ARRAYS D, Y, F, G, IPIV, XMY1, AND
C     XMY2 ARE USED THROUGHOUT THE PROGRAM
C     FOR COMPUTATIONAL PURPOSES AND NEED NOT
C     CONCERN THE USER.
C
C     **--FOR THE ARRAY DESCRIBED IN THE LINE ABOVE THIS ONE
C     THIS PARAMETER IS EQUAL TO THE FIRST DIMENSION
C     OF THE ARRAY SPECIFIED IN THE CALLING PROGRAM.
C     FOR EXAMPLE---
C     DIMENSION A(100,11)
C     LA=100
C     CALL BLSQS (...A,LA,...)
C
C.....REFERENCES.
C
C     1.  A. BJORK, BIT 7(1967) 257-278 AND 8(1968) 8-30.
C
C
      DIMENSION A(LA,NPU), QR(LQR,N), F(MPN),G(MPN), XV(NPU)
      DIMENSION RESV(M), XMY1(M), XMY2(N), D(N), Y(N), IPIV(N)
      DIMENSION B(LB,NRHS), X(LX,NRHS), RES(LRES,NRHS)
C
C                              DEFINE QR MATRIX
      IOUT=6
      IF(M-N)10,30,30
   10 WRITE(IOUT,20)
   20 FORMAT(1HO,51HNUMBER OF EQUATIONS IS LESS THAN NUMBER OF UNKNOWNS)
```

```
      GO TO 80
   30 DO 40 J=1,N
      DO 40 I=1,M
   40 QR(I,J)=A(I,J)
      CALL DECOM       (M,N,M1,N1,ISING,ETA,TOL,IPIV,D,QR,LQR)
C
C                              BEGIN (IV)TH RIGHT HAND SIDE
      DO 70 IV=1,NRHS
      DO 50 I=1,M
   50 A(I,NPU)=B(I,IV)
      MPU=M+1
      CALL ACSOL         (M,N,M1,N1,MPN,NPU,A,LA,QR,LQR,D,IPIV,
     *                    XV,RESV,F,G,Y,XMY1,XMY2,IFAIL,ETA)
C
C                              STORE SOLUTIONS AND RESIDUALS
      DO 60 J=1,N
   60 X(J,IV)=XV(J)
      M1PU=M1+1
      DO 70 I=M1PU,M
   70 RES(I,IV)=RESV(I)
   80 RETURN
      END
      SUBROUTINE DECOM(M,N,M1,N1,ISING,ETA,TOL,IPIV,D,QR,LQR)
      IMPLICIT REAL*8(A-H,O-Z)
C
C     THIS SUBROUTINE USES THE MODIFIED GRAM-SCHMIDT
C     ALGORITHM WITH PIVOTING TO OBTAIN THE
C     DECOMPOSITION OF THE MATRIX STORED IN QR
C     NEEDED FOR THE ITERATIVE REFINEMENT.  IF THE
C     N1 FIRST ROWS OF QR MODIFIED BY ROUNDING
C     ERRORS ARE LINEARLY DEPENDENT, THE VARIABLE
C     ISING IS SET EQUAL TO ONE AND THE DECOMPOSITION IS
C     NOT COMPLETED.  ON NORMAL EXIT, ISING HAS THE
C     VALUE ZERO.
C
C     AUTHORS NOTE---  THE COMPUTATION
C     OF THE BOOLEAN VARIABLE -NOT FINIS- IS,
C     OF COURSE, NOT NECESSARY, EXCEPT TO
C     PROVIDE CONTINUITY BETWEEN THE
C     FORTRAN AND ALGOL VERSIONS OF THIS
C     ALGORITHM.
C
      DIMENSION QR(LQR,N), D(N), IPIV(N)
      IOUT=6
      ZERO=0.0
      UNITY=1.0
      TOL2=TOL**2
      MV=1
      MH=M1
C                              FSUM=.TRUE.
      IFSUM=1
      N1=N
      MS=M
C                              FINIS=.FALSE.
      IFIN=0
      DO 10 J=1,N
   10 IPIV(J)=J
C                              BEGIN STEP NUMBER -IS-
C                              OF THE DECOMPOSITION
      DO 520 IS=1,N
      K=M+IS
      IF(IS-M1-1)30,20,30
   20 MV=M1+1
      MH=M
C                              FSUM=.TRUE.
      IFSUM=1
```

```
C                                   COMPUTE -NOT FINIS-
   30 IF(IFIN)50,40,50
   40 NFIN=1
      GO TO 60
   50 NFIN=0
   60 IF(NFIN-1)210,70,210
C                                   BEGIN PIVOT SEARCH
C                                   STATEMENT NR 70 IS THE LABEL -PIV-...
   70 DS=ZERO
      DO 120 J=IS,N
      IF(IFSUM-1)100,80,100
   80 SUM=ZERO
      DO 90 I=MV,MH
   90 SUM=SUM+QR(I,J)*QR(I,J)
      D(J)=SUM
  100 IF(DS-D(J))110,120,120
  110 DS=D(J)
      IP=J
  120 CONTINUE
      IF(IFSUM-1)140,130,140
  130 DM=DS
  140 IF(DS-ETA*DM)150,160,160
  150 IFSUM=1
      GO TO 170
  160 IFSUM=0
  170 IF(IFSUM-1)180,70,180
  180 IF(IP-IS)190,220,190
C                                   BEGIN COLUMN INTERCHANGE
  190 I=IPIV(IP)
      IPIV(IP)=IPIV(IS)
      IPIV(IS)=I
      D(IP)=D(IS)
      KMU=K-1
      DO 200 I=1,KMU
      C=QR(I,IP)
      QR(I,IP)=QR(I,IS)
  200 QR(I,IS)=C
C                                   END COLUMN INTERCHANGE
C                                   END PIVOT SEARCH
      GO TO 220
C                                   STATEMENT NR 210 IS THE LABEL -NDS-...
  210 MH=K-1
      MS=MH
  220 IF(IFIN-1)230,240,230
  230 C=ZERO
      GO TO 250
  240 C=UNITY
  250 SUM=ZERO
      DO 260 I=MV,MH
  260 SUM=SUM+QR(I,IS)*QR(I,IS)
      SUM=SUM+C
      D(IS)=SUM
      DS=D(IS)
C                                   COMPUTE -NOT FINIS-
      IF(IFIN)280,270,280
  270 NFIN=1
      GO TO 290
  280 NFIN=0
  290 IF(NFIN-1)400,300,400
  300 IF(IS-M1)400,400,310
  310 IF(DS-TOL2*D(M1+1))320,320,400
  320 IFIN=1
      N1=IS-1
      MV=M+1
      DO 390 IP=IS,N
C                                   CHECK FOR M1=0
```

```
      IF(M1)370,370,330
  330 DO 340 I=1,M1
  340 QR(I,IP)=ZERO
      DO 360 J=1,M1
      SUM=ZERO
      DO 350 I=1,M
  350 SUM=SUM+QR(I,J)*QR(I,IP)
      C=SUM/D(J)
      DO 360 I=1,M1
  360 QR(I,IP)=QR(I,IP)-C*QR(I,J)
  370 MPU=M+1
      MPN1=M+N1
      DO 390 JJ=MPU,MPN1
      J=MPU+MPN1-JJ
      SUM=ZERO
      DO 380 I=J,MPN1
      ILM=I-M
  380 SUM=SUM+QR(J,ILM)*QR(I,IP)
  390 QR(J,IP)=-SUM
      GO TO 210
  400 IF(DS)430,410,430
C                                 HERE FOR SINGULAR EXIT
  410 ISING=1
      WRITE(IOUT,420)
  420 FORMAT(24HOEXIT SINGULAR IN DECOMP)
      GO TO 530
  430 QR(K,IS)=-UNITY
      ISPU=IS+1
      IF(ISPU-N)440,440,520
C                                 BEGIN ORTHOGONALIZATION
  440 DO 510 J=ISPU,N
      SUM=ZERO
      DO 450 I=MV,MH
  450 SUM=SUM+QR(I,J)*QR(I,IS)
      RSJ=SUM/DS
      QR(K,J)=RSJ
      DO 460 I=1,MS
  460 QR(I,J)=QR(I,J)-RSJ*QR(I,IS)
C                                 COMPUTE -NOT FINIS-
      IF(IFIN)470,480,470
  470 NFIN=0
      GO TO 490
  480 NFIN=1
  490 CONTINUE
      IF(NFIN-1)520,500,520
  500 D(J)=D(J)-DS*RSJ**2
  510 CONTINUE
  520 CONTINUE
C                                 END ORTHOGONALIZATION
C                                 END STEP NUMBER -IS-
      ISING=0
  530 RETURN
      END
      SUBROUTINE ACSOL (M,N,M1,N1,MPN,NPU,A,LA,QR,LQR,D,IPIV,
     *                  XV,RESV,F,G,Y,XMY1,XMY2,IFAIL,ETA)
      IMPLICIT REAL*8(A-H,O-Z)
C
C     THIS SUBROUTINE USES THE DECOMPOSITION
C     STORED IN QR FOR THE ITERATIVE REFINEMENT
C     OF THE SOLUTION CORRESPONDING TH THE RIGHT
C     HAND SIDE GIVEN IN THE (N+1)ST COLUMN OF
C     A.  IF THE SOLUTION FAILS TO IMPROVE
C     SUFFICIENTLY, THE VARIABLE IFAIL IS SET
C     EQUAL TO ONE AT EXIT.  OTHERWISE, IFAIL
C     IS ZERO.
C
```

```
      DIMENSION A(LA,NPU), QR(LQR,N), F(MPN),G(MPN), XV(NPU)
      DIMENSION RESV(M), XMY1(M), XMY2(N), D(N), Y(N), IPIV(N)
      DPNUL=0.0
      ZERO=0.0
      UNITY=1.0
      IOUT=6
C                                BJORKS CHOICE FOR THIS PARAMETER
      SIXFO=64.0
C                                BJORKS CHOICE FOR THIS PARAMETER
      FOUR=4.0
      XV(NPU)=-UNITY
      ETA2=ETA**2
      DO 10 I=1,M
      F(I)=A(I,NPU)
      G(I)=ZERO
      RESV(I)=DPNUL
   10 XMY1(I)=ZERO
      DO 20 IS=1,N
      XV(IS)=ZERO
      JAYE=M+IS
      F(JAYE)=ZERO
      G(JAYE)=ZERO
   20 XMY2(IS)=ZERO
      K=0
      ENDR2=ZERO
      ENDX2=ZERO
C                                BEGIN KTH ITERATION STEP
   30 ENDR1=ENDR2
      ENDX1=ENDX2
      ENDR2=ZERO
      ENDX2=ZERO
      IF(K)40,280,40
C                                BEGIN NEW RESIDUALS
   40 DO 50 I=1,M
      ALPHA=F(I)
      RESV(I)=RESV(I)+ALPHA
   50 XMY1(I)=XMY1(I)+G(I)
C     WRITE(IOUT,60)K
C  60 FORMAT(1HO,20HFOR ITERATION NUMBER,I2,20H  RESIDUAL VECTOR IS,//)
C     DO 70 I=1,M
C  70 WRITE(IOUT,80)RESV(I)
C  80 FORMAT(1HO,D22.15)
      DO 130 IS=1,N
      J=M+IS
      IP=IPIV(IS)
      XV(IP)=XV(IP)+F(J)
      XMY2(IP)=XMY2(IP)+G(J)
C                                ** A DOUBLE PRECISION INNER PRODUCT **
      DPSUM=DPNUL
      DO 90 I=1,M
      ALPHA=A(I,IP)
      BETA=XMY1(I)
   90 DPSUM=DPSUM+ALPHA*BETA
      ALPHA=XV(IP)
      DPSUM=DPSUM-ALPHA
      G(J)=-DPSUM
      IF(IS-N1)110,110,100
  100 F(J)=ZERO
      GO TO 130
C                                ** A DOUBLE PRECISION INNER PRODUCT **
  110 DPSUM=DPNUL
      DO 120 I=1,M
      ALPHA=A(I,IP)
  120 DPSUM=DPSUM+ALPHA*RESV(I)
      F(J)=-DPSUM
  130 CONTINUE
```

```
      WRITE(IOUT,140)K
  140 FORMAT(1HO,20HFOR ITERATION NUMBER,I2,20H  SOLUTION VECTOR IS,//)
      DO 150 I=1,N
  150 WRITE(IOUT,160)XV(I)
  160 FORMAT(1HO,E15.7)
      DO 250 I=1,M
      IF(I-M1)180,180,170
  170 C=RESV(I)
      GO TO 190
  180 C=DPNUL
C                              ** A DOUBLE PRECISION INNER PRODUCT **
  190 DPSUM=DPNUL
      DO 200 J=1,NPU
      ALPHA=A(I,J)
      BETA=XV(J)
  200 DPSUM=DPSUM+ALPHA*BETA
      DPSUM=DPSUM+C
      F(I)=-DPSUM
      IF(I-M1)210,210,220
  210 C=DPNUL
      GO TO 230
  220 C=XMY1(I)
C                              ** A DOUBLE PRECISION INNER PRODUCT **
  230 DPSUM=DPNUL
      DO 240 J=1,N
      ALPHA=A(I,J)
      BETA=XMY2(J)
  240 DPSUM=DPSUM+ALPHA*BETA
      DPSUM=DPSUM+C
  250 G(I)=-DPSUM
      N1PU=N1+1
      DO 270 JJ=N1PU,N
      IS=N1PU+N-JJ
      SUM=ZERO
      MPIS=M+IS
      DO 260 I=1,MPIS
  260 SUM=SUM+QR(I,IS)*G(I)
      JAYE=M+IS
  270 G(JAYE)=SUM
C                              END NEW RESIDUALS
  280 CALL SOLVE        (M,N,M1,N1,MPN,QR,LQR,D,Y,F)
      MPU=M+1
      N1PU=N1+1
      IF(N1PU-N)290,290,320
  290 DO 310 IS=N1PU,N
      J=M+IS
      SUM=ZERO
      DO 300 I=MPU,J
  300 SUM=SUM+QR(I,IS)*F(I)
      SUM=SUM+G(J)
      CSP=SUM/D(IS)
      DO 310 I=1,J
  310 F(I)=F(I)-CSP*QR(I,IS)
  320 MPN=M+N
      DO 350 J=MPU,MPN
      IF(J-M-N1)340,340,330
  330 G(J)=ZERO
      GO TO 350
  340 G(J)=G(J)+F(J)
  350 CONTINUE
      CALL SOLVE        (M,N,M1,N1,MPN,QR,LQR,D,Y,G)
      DO 360 I=1,M
  360 ENDR2=ENDR2+F(I)**2
      DO 370 I=MPU,MPN
      IF (F(I).LT.1.E-40) GOTO 370
      ENDX2=ENDX2+F(I)**2
```

```
  370 CONTINUE
      IF(K)390,380,390
  380 ENR=ENDR2
      ENX=ENDX2
C                                 END KTH ITERATION
  390 K=K+1
C                                 KTH ITERATION TO BE DONE AT LEAST TWICE
      IF(K-1)30,30,400
C                                 TEST FOR FURTHER ITERATION
  400 IF(SIXFO*ENDX2-ENDX1)410,420,420
  410 IF(ENDX2-ETA2*ENX)420,420,30
  420 IF(SIXFO*ENDR2-ENDR1)430,440,440
  430 IF(ENDR2-ETA2*ENR)440,440,30
C                                 TEST FOR FAILURE EXIT
  440 IF(ENDR2-FOUR*ETA2*ENR)480,480,450
  450 IF(ENDX2-FOUR*ETA2*ENX)480,480,460
  460 IFAIL=1
C                                 HERE FOR FAILURE EXIT
      WRITE(IOUT,470)
  470 FORMAT(19HOEXIT FAIL IN ACSOL)
      GO TO 490
  480 IFAIL=1
C                                 HERE FOR NORMAL EXIT
  490 RETURN
      END
      SUBROUTINE SOLVE (M,N,M1,N1,MPN,QR,LQR,D,Y,F)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION QR(LQR,N), F(MPN), Y(N), D(N)
      ZERO=0.0
      MV=1
      MH=M1
      DO 100 IS=1,N1
      J=M+IS
      IF(IS-M1-1)20,10,20
   10 MV=M1+1
      MH=M
   20 ISM1=IS-1
      SUM=ZERO
      IF(ISM1)50,50,30
   30 DO 40 I=1,ISM1
      MPI=M+I
   40 SUM=SUM+QR(MPI,IS)*Y(I)
   50 Y(IS)=SUM-F(J)
      Y(IS)=-Y(IS)
      IF(IS-M1)70,70,60
   60 C=-Y(IS)
      GO TO 80
   70 C=ZERO
   80 SUM=ZERO
      DO 90 I=MV,MH
   90 SUM=SUM+QR(I,IS)*F(I)
      SUM=SUM+C
      C=SUM / D(IS)
      F(J)=C
      DO 100 I=MV,M
  100 F(I)=F(I)-C*QR(I,IS)
      IF(M1)150,150,110
  110 DO 120 I=1,M1
  120 F(I)=ZERO
      DO 140 IS=1,M1
      SUM=ZERO
      DO 130 I=1,M
  130 SUM=SUM+QR(I,IS)*F(I)
      SUM=SUM-Y(IS)
      C=SUM / D(IS)
      DO 140 I=1,M1
```

```
140 F(I)=F(I)-C*QR(I,IS)
150 MPU=M+1
    MPN1=M+N1
    DO 170 JJ=MPU,MPN1
    J=MPN1+MPU-JJ
    SUM=ZERO
    DO 160 I=J,MPN1
    ILM=I-M
160 SUM=SUM+QR(J,ILM)*F(I)
170 F(J)=-SUM
    RETURN
    END
```

APPENDIX D

PROGRAM LISTING OF LLSQF

```
      SUBROUTINE LLSQF(A,IA,M,N,B,TOL,KBASIS,X,H,IP,IER)
      INTEGERIA,M,N,KBASIS,IP(N)
      REALA(IA,N),B(M),TOL,X(N),H(N)
      INTEGERI,IER,J,JCOL,JJ,JSTART,K,KP1,L,LDIAG,LMAX
      REALBB,DLOSS,DLOSSJ,RCOND,RCONDJ,RNORM,TMP,XNORM
      REALSASUM,SDOT,SNRM2
      LDIAG=MINO(M,N)
      IER=129
      IF(LDIAG.LE.O)GOTO9OOO
      IER=13O
      IF(TOL.GT.1.O)GOTO9OOO
      IER=O
      JSTART=MAXO(KBASIS+1,1)
      DO35J=1,LDIAG
      IP(J)=J
      IF(J.LE.KBASIS)GOTO3O
      LMAX=J
      IF(J.EQ.JSTART)GOTO1O
      DLOSSJ=1.O
      IF(BB.EQ.O.O)GOTO3O
      TMP=BB
      BB=BB*SQRT(AMAX1(1.O-(B(J-1)/BB)**2,O.O))
      IF(BB.EQ.O.O)GOTO3O
      DLOSSJ=BB/TMP
      DO5L=J,N
      IF(H(L).EQ.O.O)GOTO5
      TMP=H(L)
      H(L)=H(L)*SQRT(AMAX1(1.O-(A(J-1,L)/H(L))**2,O.O))
      DLOSSJ=AMIN1(DLOSSJ,H(L)/TMP)
      TMP=X(L)
      X(L)=O.O
      IF(H(L).EQ.O.O)GOTO5
      X(L)=TMP-A(J-1,L)*B(J-1)
      IF(H(LMAX).EQ.O.O)LMAX=L
      IF(ABS(X(L))/H(L).GT.ABS(X(LMAX))/H(LMAX))LMAX=L
5     CONTINUE
      DLOSS=DLOSS*DLOSSJ
      TMP=1O.O+DLOSS
      IF(TMP.GT.1O.O)GOTO2O
1O    BB=SNRM2(M-J+1,B(J),1)
      IF(BB.EQ.O.O)GOTO3O
      DO15L=J,N
      H(L)=SNRM2(M-J+1,A(J,L),1)
      X(L)=O.O
      IF(H(L).EQ.O.O)GOTO15
      X(L)=SDOT(M-J+1,A(J,L),1,B(J),1)
      IF(H(LMAX).EQ.O.O)LMAX=L
      IF(ABS(X(L))/H(L).GT.ABS(X(LMAX))/H(LMAX))LMAX=L
15    CONTINUE
      DLOSS=1.O
2O    CONTINUE
      IP(J)=LMAX
      IF(LMAX.EQ.J)GOTO3O
      DO25I=1,M
      TMP=A(I,J)
      A(I,J)=A(I,LMAX)
      A(I,LMAX)=TMP
```

```
   25 CONTINUE
      H(LMAX)=H(J)
      X(LMAX)=X(J)
30    JCOL=MINO(J+1,N)
      CALLSVHS12(1,J,J+1,M,A(1,J),1,H(J),A(1,JCOL),1,IA,N-J)
      CALLSVHS12(2,J,J+1,M,A(1,J),1,H(J),B,1,M,1)
35    CONTINUE
      RCOND=0.0
      K=0
      RNORM=0.0
      XNORM=0.0
      DO55J=1,LDIAG
      IF(ABS(A(J,J)).EQ.0.0)GOTO60
      IF(TOL.LT.0.0)GOTO50
      RNORM=AMAX1(RNORM,SASUM(J,A(1,J),1))
      X(J)=1.0/A(J,J)
      IF(J.LT.2)GOTO45
      I=J
      DO40L=2,J
      I=I-1
      X(I)=-SDOT(J-I,X(I+1),1,A(I,I+1),IA)/A(I,I)
   40 CONTINUE
45    CONTINUE
      XNORM=AMAX1(XNORM,SASUM(J,X,1))
      RCONDJ=1.0/(RNORM*XNORM)
      IF(TOL.GE.RCONDJ)GOTO
      RCOND=RCONDJ
50    K=J
55    CONTINUE
60    KP1=K+1
      KBASIS=K
      DO65J=1,N
65    X(J)=0.0
      IF(KBASIS.EQ.0)GOTO90
      X(K)=B(K)/A(K,K)
      IF(K.LT.2)GOTO75
      I=K
      DO70L=2,K
      I=I-1
      X(I)=(B(I)-SDOT(K-I,X(I+1),1,A(I,I+1),IA))/A(I,I)
70    CONTINUE
75    J=LDIAG+1
      DO80JJ=1,LDIAG
      J=J-1
      L=IP(J)
      IF(L.EQ.J)GOTO80
      TMP=X(L)
      X(L)=X(J)
      X(J)=TMP
80    CONTINUE
      DO85I=1,K
85    B(I)=0.0
90    J=LDIAG+1
      DO95JJ=1,LDIAG
      J=J-1
      CALLSVHS12(2,J,J+1,M,A(1,J),1,H(J),B,1,M,1)
95    CONTINUE
      IF(TOL.GE.0.0)TOL=RCOND
      GOTO9005
9000  CONTINUE
      CALLUERTST(IER,'LLSQF ')
 9005 RETURN
      END

      SUBROUTINE UGETIO(IOPT,NIN,NOUT)
```

```
      INTEGERIOPT,NIN,NOUT
      INTEGERNIND,NOUTD
      DATANIND/5/,NOUTD/6/
      IF(IOPT.EQ.3)GOTO10
      IF(IOPT.EQ.2)GOTO5
      IF(IOPT.NE.1)GOTO9005
      NIN=NIND
      NOUT=NOUTD
      GOTO9005
5     NIND=NIN
      GOTO9005
10    NOUTD=NOUT
9005  RETURN
      END

      SUBROUTINE UERTST(IER,NAME)
      INTEGERIER
      CHARACTER*2NAME(3)
      CHARACTER*2NAMSET(3),NAMEQ(3)
      CHARACTER*1 IEQ
      DATANAMSET/'UE','RS','ET'/
      DATANAMEQ/'  ','  ','  '/
      DATALEVEL/4/,IEQDF/0/,IEQ/'='/
      IF(IER.GT.999)GOTO25
      IF(IER.LT.-32)GOTO55
      IF(IER.LE.128)GOTO5
      IF(LEVEL.LT.1)GOTO30
      CALLUGETIO(1,NIN,IOUNIT)
      IF(IEQDF.EQ.1)WRITE(IOUNIT,35)IER,NAMEQ,IEQ,NAME
      IF(IEQDF.EQ.0)WRITE(IOUNIT,35)IER,NAME
      GOTO30
5     IF(IER.LE.64)GOTO10
      IF(LEVEL.LT.2)GOTO30
      CALLUGETIO(1,NIN,IOUNIT)
      IF(IEQDF.EQ.1)WRITE(IOUNIT,40)IER,NAMEQ,IEQ,NAME
      IF(IEQDF.EQ.0)WRITE(IOUNIT,40)IER,NAME
      GOTO30
10    IF(IER.LE.32)GOTO15
      IF(LEVEL.LT.3)GOTO30
      CALLUGETIO(1,NIN,IOUNIT)
      IF(IEQDF.EQ.1)WRITE(IOUNIT,45)IER,NAMEQ,IEQ,NAME
      IF(IEQDF.EQ.0)WRITE(IOUNIT,45)IER,NAME
      GOTO30
15    CONTINUE
      DO20I=1,3
      IF(NAME(I).NE.NAMSET(I))GOTO25
20    CONTINUE
      LEVOLD=LEVEL
      LEVEL=IER
      IER=LEVOLD
      IF(LEVEL.LT.0)LEVEL=4
      IF(LEVEL.GT.4)LEVEL=4
      GOTO30
25    CONTINUE
      IF(LEVEL.LT.4)GOTO30
      CALLUGETIO(1,NIN,IOUNIT)
      IF(IEQDF.EQ.1)WRITE(IOUNIT,50)IER,NAMEQ,IEQ,NAME
      IF(IEQDF.EQ.0)WRITE(IOUNIT,50)IER,NAME
30    IEQDF=0
      RETURN
   35 FORMAT(19H *** TERMINAL ERROR,10X,7H(IER = ,I3,20H) FROM IMSL ROUT
     *INE ,3A2,A1,3A2)
   40 FORMAT(36H *** WARNING WITH FIX ERROR  (IER = ,I3,20H) FROM IMSL R
     *OUTINE ,3A2,A1,3A2)
   45 FORMAT(18H *** WARNING ERROR,11X,7H(IER = ,I3,20H) FROM IMSL ROUTI
```

```
     *NE ,3A2,A1,3A2)
   50 FORMAT(20H *** UNDEFINED ERROR,9X,7H(IER = ,I5,20H) FROM IMSL ROUT
     *INE ,3A2,A1,3A2)
   55 IEQDF=1
      DO60I=1,3
60    NAMEQ(I)=NAME(I)
65    RETURN
      END


      REAL FUNCTION SDOT(N,SX,INCX,SY,INCY)
      INTEGERN,INCX,INCY
      REALSX(1),SY(1)
      INTEGERI,M,MP1,NS,IX,IY
      SDOT=0.0E0
      IF(N.LE.0)RETURN
      IF(INCX.EQ.INCY)IF(INCX-1)5,15,35
5     CONTINUE
      IX=1
      IY=1
      IF(INCX.LT.0)IX=(-N+1)*INCX+1
      IF(INCY.LT.0)IY=(-N+1)*INCY+1
      DO10I=1,N
      SDOT=SDOT+SX(IX)*SY(IY)
      IX=IX+INCX
      IY=IY+INCY
10    CONTINUE
      RETURN
15    M=N-(N/5)*5
      IF(M.EQ.0)GOTO25
      DO20I=1,M
      SDOT=SDOT+SX(I)*SY(I)
20    CONTINUE
      IF(N.LT.5)RETURN
25    MP1=M+1
      DO30I=MP1,N,5
      SDOT=SDOT+SX(I)*SY(I)+SX(I+1)*SY(I+1)+SX(I+2)*SY(I+2)+SX(I+3)*SY(I
     *+3)+SX(I+4)*SY(I+4)
30    CONTINUE
      RETURN
35    CONTINUE
      NS=N*INCX
      DO40I=1,NS,INCX
      SDOT=SDOT+SX(I)*SY(I)
   40 CONTINUE
      RETURN
      END


      REAL FUNCTION SNRM2(N,SX,INCX)
      INTEGERN,INCX
      REALSX(1)
      INTEGERI,J,NEXT,NN
      REALCUTLO,CUTHI,HITEST,SUM,XMAX,ZERO,ONE
      DATAZERO,ONE/0.0E0,1.0E0/
      DATACUTLO,CUTHI/4.441E-16,1.304E19/
      IF(N.GT.0)GOTO5
      SNRM2=ZERO
      GOTO70
5     ASSIGN15TONEXT
      SUM=ZERO
      NN=N*INCX
      I=1
10    GOTONEXT,(15,20,35,40)
15    IF(ABS(SX(I)).GT.CUTLO)GOTO55
      ASSIGN20TONEXT
```

```
      XMAX=ZERO
20    IF(SX(I).EQ.ZERO)GOTO65
      IF(ABS(SX(I)).GT.CUTLO)GOTO55
      ASSIGN35TONEXT
      GOTO30
25    I=J
      ASSIGN40TONEXT
      SUM=(SUM/SX(I))/SX(I)
   30 XMAX=ABS(SX(I))
      GOTO45
35    IF(ABS(SX(I)).GT.CUTLO)GOTO50
40    IF(ABS(SX(I)).LE.XMAX)GOTO45
      SUM=ONE+SUM*(XMAX/SX(I))**2
      XMAX=ABS(SX(I))
      GOTO65
45    SUM=SUM+(SX(I)/XMAX)**2
      GOTO65
50    SUM=(SUM*XMAX)*XMAX
   55 HITEST=CUTHI/FLOAT(N)
      DO60J=I,NN,INCX
      IF(ABS(SX(J)).GE.HITEST)GOTO25
60    SUM=SUM+SX(J)**2
      SNRM2=SQRT(SUM)
      GOTO70
65    CONTINUE
      I=I+INCX
      IF(I.LE.NN)GOTO10
      SNRM2=XMAX*SQRT(SUM)
70    CONTINUE
      RETURN
      END


      SUBROUTINE SVHS12(MODE,LP,L1,M,U,INCU,UP,C,INCC,ICV,NCV)
      INTEGERMODE,LP,L1,M,INCU,INCC,ICV,NCV
      REALU(1),UP,C(1)
      INTEGERIJ,ILP,IL1,IM,INCR,I2,I3,I4,J
      REALONE,CL,CLINV,SM1
      ONE=1.
      IF(0.GE.LP.OR.LP.GE.L1.OR.L1.GT.M)GOTO9005
      ILP=(LP-1)*INCU+1
      IL1=(L1-1)*INCU+1
      IM=(M-1)*INCU+1
      CL=ABS(U(ILP))
      IF(MODE.EQ.2)GOTO15
      DO5IJ=IL1,IM,INCU
5     CL=AMAX1(ABS(U(IJ)),CL)
      IF(CL.LE.0.0)GOTO9005
      CLINV=ONE/CL
      SM=(U(ILP)*CLINV)**2
      DO10IJ=IL1,IM,INCU
 10   SM=SM+(U(IJ)*CLINV)**2
      SM1=SM
      CL=CL*SQRT(SM1)
      IF(U(ILP).GT.0.0)CL=-CL
      UP=U(ILP)-CL
      U(ILP)=CL
      GOTO20
15    IF(CL.LE.0.0)GOTO9005
20    IF(NCV.LE.0)GOTO9005
      B=UP*U(ILP)
      IF(B.GE.0.0)GOTO9005
      B=ONE/B
      I2=1-ICV+INCC*(LP-1)
      INCR=INCC*(L1-LP)
      DO35J=1,NCV
```

```
      I2=I2+ICV
      I3=I2+INCR
      I4=I3
      SM=C(I2)*UP
      DO25IJ=IL1,IM,INCU
      SM=SM+C(I3)*U(IJ)
      I3=I3+INCC
25    CONTINUE
      IF(SM.EQ.0.0)GOTO35
      SM=SM*B
      C(I2)=C(I2)+SM*UP
      DO30IJ=IL1,IM,INCU
      C(I4)=C(I4)+SM*U(IJ)
      I4=I4+INCC
   30 CONTINUE
35    CONTINUE
9005  RETURN
      END

      REAL FUNCTION SASUM(N,SX,INCX)
      INTEGERN,INCX
      REALSX(1)
      INTEGERI,M,MP1,NS
      SASUM=0.0E0
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GOTO10
      NS=N*INCX
      DO5I=1,NS,INCX
      SASUM=SASUM+ABS(SX(I))
5     CONTINUE
      RETURN
10    M=N-(N/6)*6
      IF(M.EQ.0)GOTO20
      DO15I=1,M
      SASUM=SASUM+ABS(SX(I))
15    CONTINUE
      IF(N.LT.6)RETURN
20    MP1=M+1
      DO25I=MP1,N,6
      SASUM=SASUM+ABS(SX(I))+ABS(SX(I+1))+ABS(SX(I+2))+ABS(SX(I+3))+ABS(
     *SX(I+4))+ABS(SX(I+5))
   25 CONTINUE
      RETURN
      END

      DOUBLE PRECISION FUNCTION DDOT(N,DX,INCX,DY,INCY)
      DOUBLEPRECISIONDX(1),DY(1)
      INTEGERN,INCX,INCY
      INTEGERI,M,MP1,NS,IX,IY
      DDOT=0.D0
      IF(N.LE.0)RETURN
      IF(INCX.EQ.INCY)IF(INCX-1)5,15,35
5     CONTINUE
      IX=1
      IY=1
      IF(INCX.LT.0)IX=(-N+1)*INCX+1
      IF(INCY.LT.0)IY=(-N+1)*INCY+1
      DO10I=1,N
      DDOT=DDOT+DX(IX)*DY(IY)
      IX=IX+INCX
      IY=IY+INCY
10    CONTINUE
      RETURN
15    M=N-(N/5)*5
```

```
      IF(M.EQ.0)GOTO25
      DO20I=1,M
      DDOT=DDOT+DX(I)*DY(I)
   20 CONTINUE
      IF(N.LT.5)RETURN
25    MP1=M+1
      DO30I=MP1,N,5
      DDOT=DDOT+DX(I)*DY(I)+DX(I+1)*DY(I+1)+DX(I+2)*DY(I+2)+DX(I+3)*DY(I
     *+3)+DX(I+4)*DY(I+4)
30    CONTINUE
      RETURN
35    CONTINUE
      NS=N*INCX
      DO40I=1,NS,INCX
      DDOT=DDOT+DX(I)*DY(I)
40    CONTINUE
      RETURN
      END


      DOUBLE PRECISION FUNCTION DNRM2(N,SX,INCX)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION SX(1)
      DATAZERO,ONE/0.0E0,1.0E0/
      DATACUTLO,CUTHI/4.441E-16,1.304E19/
      IF(N.GT.0)GOTO5
      DNRM2=ZERO
      GOTO70
5     ASSIGN15TONEXT
      SUM=ZERO
      NN=N*INCX
      I=1
10    GOTONEXT,(15,20,35,40)
15    IF(DABS(SX(I)).GT.CUTLO)GOTO55
      ASSIGN20TONEXT
      XMAX=ZERO
20    IF(SX(I).EQ.ZERO)GOTO65
      IF(DABS(SX(I)).GT.CUTLO)GOTO55
      ASSIGN35TONEXT
      GOTO30
25    I=J
      ASSIGN40TONEXT
      SUM=(SUM/SX(I))/SX(I)
   30 XMAX=ABS(SX(I))
      GOTO45
35    IF(DABS(SX(I)).GT.CUTLO)GOTO50
40    IF(DABS(SX(I)).LE.XMAX)GOTO45
      SUM=ONE+SUM*(XMAX/SX(I))**2
      XMAX=DABS(SX(I))
      GOTO65
45    SUM=SUM+(SX(I)/XMAX)**2
      GOTO65
50    SUM=(SUM*XMAX)*XMAX
   55 HITEST=CUTHI/FLOAT(N)
      DO60J=I,NN,INCX
      IF(ABS(SX(J)).GE.HITEST)GOTO25
60    SUM=SUM+SX(J)**2
      DNRM2=DSQRT(SUM)
      GOTO70
65    CONTINUE
      I=I+INCX
      IF(I.LE.NN)GOTO10
      DNRM2=XMAX*DSQRT(SUM)
70    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DVHS12(MODE,LP,L1,M,U,INCU,UP,C,INCC,ICV,NCV)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION U(1),C(1)
      ONE=1.
      IF(O.GE.LP.OR.LP.GE.L1.OR.L1.GT.M)GOTO9005
      ILP=(LP-1)*INCU+1
      IL1=(L1-1)*INCU+1
      IM=(M-1)*INCU+1
      CL=DABS(U(ILP))
      IF(MODE.EQ.2)GOTO15
      DO5IJ=IL1,IM,INCU
5     CL=DMAX1(DABS(U(IJ)),CL)
      IF(CL.LE.O.O)GOTO9005
      CLINV=ONE/CL
      SM=(U(ILP)*CLINV)**2
      DO1OIJ=IL1,IM,INCU
10    SM=SM+(U(IJ)*CLINV)**2
      SM1=SM
      CL=CL*DSQRT(SM1)
      IF(U(ILP).GT.O.O)CL=-CL
      UP=U(ILP)-CL
      U(ILP)=CL
      GOTO2O
15    IF(CL.LE.O.O)GOTO9005
20    IF(NCV.LE.O)GOTO9005
      B=UP*U(ILP)
      IF(B.GE.O.O)GOTO9005
      B=ONE/B
      I2=1-ICV+INCC*(LP-1)
      INCR=INCC*(L1-LP)
      DO35J=1,NCV
      I2=I2+ICV
      I3=I2+INCR
      I4=I3
      SM=C(I2)*UP
      DO25IJ=IL1,IM,INCU
      SM=SM+C(I3)*U(IJ)
      I3=I3+INCC
25    CONTINUE
      IF(SM.EQ.O.O)GOTO35
      SM=SM*B
      C(I2)=C(I2)+SM*UP
      DO3OIJ=IL1,IM,INCU
      C(I4)=C(I4)+SM*U(IJ)
      I4=I4+INCC
   30 CONTINUE
35    CONTINUE
9005  RETURN
      END


      DOUBLE PRECISION FUNCTION DASUM(N,SX,INCX)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION SX(1)
      DASUM=O.OEO
      IF(N.LE.O)RETURN
      IF(INCX.EQ.1)GOTO1O
      NS=N*INCX
      DO5I=1,NS,INCX
      DASUM=DASUM+DABS(SX(I))
5     CONTINUE
      RETURN
10    M=N-(N/6)*6
      IF(M.EQ.O)GOTO2O
```

```
      DO15I=1,M
      DASUM=DASUM+DABS(SX(I))
15    CONTINUE
      IF(N.LT.6)RETURN
20    MP1=M+1
      DO25I=MP1,N,6
      DASUM=DASUM+DABS(SX(I))+DABS(SX(I+1))+DABS(SX(I+2))+
     *DABS(SX(I+3))+DABS(SX(I+4))+DABS(SX(I+5))
25    CONTINUE
      RETURN
      END
```

APPENDIX E

PROGRAM LISTING OF INVHIL

```
      SUBROUTINE INVHIL (NN,S,LS)
C
C  PRODUCES THE INVERSE OF AN N BY N FINITE SEGMENT OF THE HILBERT
C  MATRIX.     H(I,J)=1/(I+J-1)
C
C  J. HERNDON AND P. NAUR, ALGORIGHM 50, COMMUNICATIONS OF THE A.C.M.
C
C  USAGE.....
C  NN SPECIFIES THE ORDER OF THE MATRIX TO BE PRODUCED.
C  S IS THE DOUBLE PRECISION ARRAY IN WHICH THE MATRIX IS RETURNED.
C  LS IS THE FIRST DIMENSION OF THE ARRAY S IN THE CALLING PROBRAM.
C
C  EXAMPLE.....
C     DOUBLE PRECISION S
C     DIMENSION S(10,10)
C     LS=10
C     N=6
C     CALL INVHIL (N,S,LS)
C     CALL EXIT
C     END
C
C  J. P. CHANDLER, F.S.U. PHYSICS DEPT.
C
      DOUBLE PRECISION S,W,AN,AJ,AK,AL,UNITY,HALF,DD,THRSH,ABSDD,DEF
      DOUBLE PRECISION DMOD
C
      DIMENSION S(LS,NN)
C
      KW=6
      UNITY=1.D0
      THRSH=.01D0
      HALF=.5D0
      N=NN
      W=N*N
      S(1,1)=W
      IF(N-2)200,10,10
   10 AN=N
      DO 20 J=2,N
      AJ=J
      W=W*((AN+AJ-UNITY)*(AN-AJ+UNITY)/(AJ-UNITY)**2)**2
   20 S(J,J)=W
      NMU=N-1
      DO 30 J=1,NMU
      JPU=J+1
      DO 30 K=JPU,N
      L=K-1
      AL=L
   30 S(J,K)=-S(J,L)*(AN+AL)*(AN-AL)/AL**2
      DO 40 J=2,N
      AJ=J
      DO 40 K=1,J
      AK=K
      S(K,J)=S(K,J)/(AJ+AK-UNITY)
   40 S(J,K)=S(K,J)
C                              ROUND OFF ALL ELEMENTS TO THE NEAREST
C                                   INTEGER.
      DO 170 J=1,N
```

```
      DO 170 K=1,J
      DD=DMOD(S(J,K),UNITY)
C                                   IF(DABS(DD)-THRSH)
      ABSDD=DD
      IF(ABSDD)50,60,60
   50 ABSDD=-ABSDD
   60 IF(ABSDD-THRSH)120,120,70
C                                   IF(DABS(DABS(DD)-UNITY)-THRSH
   70 DEF=ABSDD-UNITY
      IF(DEF)80,90,90
   80 DEF=-DEF
   90 IF(DEF-THRSH)120,120,100
  100 WRITE(KW,110)N,J,K,DD
  110 FORMAT(' POOR ACCURACY IN INVHIL FOR N = ',I3,', J = ',I3,
     *    ' J = ',I3,'.   DEFECT = ',D12.5)
C
C                                   IF(DABS(DD)-HALF)
  120 IF(ABSDD-HALF)160,160,130
C                                   DD=DD-DSIGN(UNITY,DD)
  130 DEF=UNITY
      IF(DD)140,150,150
  140 DEF=-DEF
  150 DD=DD-DEF
  160 S(J,K)=S(J,K)-DD
  170 S(K,J)=S(J,K)
      DO 180 J=1,N
  180 WRITE(KW,190)N,J,(S(J,K),K=1,N)
  190 FORMAT(/' INVHIL.  N = ',I3,5X,'J = ',I3/(1X,5D21.13))
  200 RETURN
      END
```

APPENDIX F

TEST PROGRAM FOR GIVEN

```
C*************************************************************************
C                                                                        *
C                        TEST PROGRAM FOR GIVEN                          *
C                                                                        *
C*************************************************************************
C
C                                  THE IMPLICIT STATEMENT IS USED FOR
C                                  DOUBLE PRECISION ARITHMETIC ONLY.
      IMPLICIT REAL*8(A-H,O-Z)
C
C                                  SET DIMENSIONS FOR WORK ARRAYS.
C
      DIMENSION AROW(10),D(10),TBAR(10),RBAR(45)
C
C                                  NCOL=NUMBER OF COLUMNS IN DESIGN MATRIX.
      NCOL=6
C
C                                  NR = NCOL*(NCOL-1)/2
      NR=15
C
C                                  ITYPE=1 FOR DENSE DESIGN MATRIX.
C                                  ITYPE=2 FOR SPARSE DESIGN MATRIX.
      ITYPE=1
C
      CALL GIVEN (NCOL,NR,ITYPE,AROW,D,TBAR,RBAR)
      STOP
      END
```

APPENDIX G

TEST PROGRAM FOR ORTHL

```
C*************************************************************************
C                                                                        *
C                        TEST PROGRAM FOR ORTHL                          *
C                                                                        *
C*************************************************************************
C
C                                  THE IMPLICIT STATEMENT IS USED FOR
C                                  DOUBLE PRECISION ARITHMETIC ONLY.
      IMPLICIT REAL*8 (A-H,O-Z)
C                                  SET DIMENSIONS FOR WORK ARRAYS.
      DIMENSION A(10,10),R(10,10),X(10),B(10),U(10,10),
     *          PP(10),D(10),RES(10)
C
C                                  INPUT DEVICE NUMER
      IN=5
C                                  THE FIRST DIMENSION OF A
      LAU= 10
C                                  NUMBER OF ROWS IN THE INPUT MATRIX -A-
      NR= 6
C                                  NUMBER OF COLUMNS IN -A-
      NC=6
C                                  THE FIRST DIMENSION OF R
      LR=6
C                                  IREF=0 FOR NO ITERATIVE REFINEMENT
C                                  IREF=1 FOR ITERATIVE REFINEMENT
      IREF=1
C                                  NTRAC=0 FOR NORMAL OUTPUT
C                                  NTRAC=1 FOR PRINT OUT THE RESULT OF
C                                          EACH ITERATION
      NTRAC=1
C                                  INPUT THE MATRIX -A- AND RIGHT HAND
C                                    SIDE -B-
      DO 20 I=1,NR
         READ (IN,10) (A(I,J),J=1,NC),B(I)
   10    FORMAT (7F6.1)
   20    CONTINUE
C
      CALL ORTHL (A,LAU,NR,NC,B,X,R,LR,IREF,NTRAC,NIX,U,P,PP,D)
      STOP
      END
```

APPENDIX H

TEST PROGRAM FOR BLSQS

```
C*********************************************************************
C                                                                     *
C                        TEST PROGRAM FOR BLSQS                       *
C                                                                     *
C*********************************************************************
C
C                                THE IMPLICIT STATEMENT IS USED FOR
C                                DOUBLE PRECISION ARITHMETIC ONLY.
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION QR(20,6),A(10,7),B(10,6),X(10,10),F(12),G(12)
      DIMENSION RESV(6),XMY1(6),XV(7),XMY2(6),IPIV(6),D(6),Y(6)
      DIMENSION AA(6,6),RES(10)
C     DOUBLE PRECISION RESV
C                                INPUT DEVICE NUMBER
      IN=5
C                                OUTPUT DEVICE NUMBER
      IOUT=6
C                                FIRST DIMENSION OF A
      LA=10
C                                FIRST DIMENSION OF B
      LB=10
C                                FIRST DIMENSION OF RES
      LRES=10
C                                FIRST DIMENSION OF QR
      LQR=20
C                                FIRST DIMENSION OF X
      LX=10
C                                NUMBER OF RIGHT HAND SIDES
      NRHS=1
C                                SINGLE PRECISION IBM 360
C                                RELATIVE MACHINE TOLERANCE
      ETA=1.0E-8
C                                -TOL- DETERMINES SYSTEM RANK
      TOL=1.0E-7
C                                NUMBER OF CONSTRAINTS
      M1=0
C                                NUMBER OF EQUATIONS
      M=6
C                                NUMBER OF UNKNOWNS
      N=6
C                                REQUIRED SUBROUTINE PARAMETERS
      NPU=N+1
      MPN=M+N
C                                INPUT A AND B
      DO 20 I=1,M
         READ (IN,10) (A(I,J),J=1,N),B(I,1)
   10    FORMAT(7F6.0)
   20 CONTINUE
C
C
      CALL BLSQS       (M,N,MPN,NPU,NRHS,M1,N1,ISING,IFAIL,ETA,TOL,
     *             A,LA,B,LB,X,LX,RES,LRES,QR,LQR,XV,RESV,IPIV,D,Y,
     *             F,G,XMY1,XMY2)
C
C                                PRINT OUT RESULT
      WRITE(IOUT,30)
```

```
 30 FORMAT(15H  -A- MATRIX---,//)
    DO 40 I=1,M
 40 WRITE(IOUT,50)(A(I,J),J=1,M)
 50 FORMAT(1HO,5E15.7)
    WRITE(IOUT,60)
 60 FORMAT(1HO,18HRIGHT HAND SIDE---,//)
    WRITE(IOUT,70)(B(I,1),I=1,M)
 70 FORMAT(1HO,6E15.7)
    WRITE(IOUT,80)
 80 FORMAT(1HO,18HSOLUTION VECTOR---,//)
    WRITE(IOUT,90)(X(I,1),I=1,N)
 90 FORMAT(1HO,3X,E25.16)
    WRITE(IOUT,100)N1
100 FORMAT(1HO,17HSYSTEM RANK IS---,I4,//)
    STOP
    END
```

APPENDIX I

TEST PROGRAM FOR LLSQF

```
C*************************************************************************
C                                                                        *
C                          TEST PROGRAM FOR LLSQF                        *
C                                                                        *
C*************************************************************************
C
C                                  THE IMPLICIT STATEMENT IS USED FOR
C                                  DOUBLE PRECISION ARITHMETIC ONLY.
      IMPLICIT REAL*8 (A-H,O-Z)
C                                  SET DIMENSIONS
      DIMENSION A(10,5),B(10),X(5),H(5),IP(5)
C                                  INPUT DEVICE NUMBER
      IN=5
C                                  OUTPUR DEVICE NUMBER
      LP=6
C                                  FIRST DIMENSION OF A
      IA=10
C                                  NUMBER OF ROWS IN INPUT MATRIX -A-
      M=6
C                                  NUMBER OF COLUMNS IN -A-
      N=5
C                                  TOL DETECTS RANK DEFICIENCE
      TOL=0.0
C                                  RANK OF -A-
      KBASIS=6
C                                  INPUT -A- AND -B-
      DO 20 I=1,M
         READ (IN,10) (A(I,J),J=1,N),B(I)
   10    FORMAT (7F6.0)
   20 CONTINUE
C
C
      CALL LLSQF(A,IA,M,N,B,TOL,KBASIS,X,H,IP,IER)
C
C                                 IF A CONDITION NUMBER IS CALCULATED,
C                                 ITS RECIPROCAL IS RETURNED IN TOL.
C                                 OTHERWISE, TOL IS NOT CHANGED.
      WRITE (LP,30) TOL
   30 FORMAT (/' TOL = ',E18.8)
C                                 PRINT OUT SOLUTION VECTOR
      WRITE (LP,40) (X(I),I=1,N)
   40 FORMAT (/' X   = ',/(7X,E25.16))
      STOP
      END
```

APPENDIX J

PROGRAM LISTING OF THE ORIGINAL VERSION OF ORTHOLIN2 WITHOUT ITERATIVE REFINEMENT

```
      SUBROUTINE ORTHL (A,LAU,NR,NC,B,X,R,LR,IREF,NTRAC,NIX,U,P,PP,D)
C
C  ORIGINAL VERSION OF ORTHOLIN2 BY F. L. BARER
C  ITERATIVE IMPROVEMENT HAS NOT YET BEEN IMPLEMENTED.
C  J. P. CHANDLER, COMPUTER SCIENCE DEPT., OKLAHOMA STATE UNIVERSITY
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION B(1),X(1),P(1),PP(1),D(1)
      DIMENSION A(LAU,NC),R(LR,NC),U(LAU,NC)
C
      KW=6
      RZERO=0
      NIX=1
C                                MOVE A INTO U.
      DO 1 J=1,NR
      DO 1 K=1,NC
    1 U(J,K)=A(J,K)
C                                COMPUTE U AND R.
      DO 2 J=1,NC
      S=RZERO
      DO 3 K=1,NR
      T=U(K,J)
      P(K)=T
    3 S=S+T*T
      IF(S.NE.RZERO) GO TO 20
      WRITE(KW,21)J
   21 FORMAT(/' R(J,J) IS ZERO IN ORTHL FOR J =',I3)
      RETURN
   20 R(J,J)=S
      T=RZERO
      DO 4 K=1,NR
    4 T=T+P(K)*B(K)
      X(J)=T
      JPU=J+1
      IF(JPU.GT.NC) GO TO 2
      DO 5 L=JPU,NC
      T=RZERO
      DO 6 K=1,NR
    6 T=T+P(K)*U(K,L)
      R(J,L)=T
      T=T/S
      DO 9 K=1,NR
    9 U(K,L)=U(K,L)-P(K)*T
    5 CONTINUE
    2 CONTINUE
```

```
C                                       DO THE BACK SUBSTITUTION.
      DO 10 JJ=1,NC
      J=NC+1-JJ
      T=R(J,J)
      S=X(J)
      JPU=J+1
      IF(JPU.GT.NC) GO TO 10
      DO 11 K=JPU,NC
   11 S=S-R(J,K)*X(K)
   10 X(J)=S/T
      NIX=0
      RETURN
      END
```

VITA

Hsiao-Lan Wang Loh

Candidate for the Degree of

Master of Science

Thesis: GIVENS TRANSFORMATIONS FOR LEAST SQUARES

Major Field: Computing and Information Sciences

Biographical:

Personal Data: Born in Taipei, Taiwan, the Republic of China, December 9, 1955, the daughter of Mr. and Mrs. Chiang-Lin Wang.

Education: Received Bachelor of Arts in Educational Psychology from Fu-Jen Catholic University, Taiwan, in 1977; completed requirement for the Master of Science degree at Oklahoma State University in December, 1983.

Professional Experience: Programmer at China Electronics Corporation, Taipei, Taiwan, July, 1979 to Feburary, 1980; Graduate Teaching Assistant, Department of Computing and Information Sciences, Oklahoma State University, Stillwater, Oklahoma, January, 1982 to December, 1983.