CS12

NUMERICAL METHODS FOR SOLVING LINEAR LEAST SQUARES PROBLEMS

BY

G. GOLUB

AN ALGOL PROCEDURE FOR FINDING LINEAR LEAST SQUARES SOLUTIONS

BY

PETER BUSINGER

TECHNICAL REPORT CS12
AUGUST 28, 1964

COMPUTER SCIENCE DIVISION
School of Humanities and Sciences
STANFORD UNIVERSITY

ABSTRACT

NUMERICAL METHODS FOR SOLVING LINEAR LEAST SQUARES PROBLEMS [*]

by

G. Golub

A common problem in a Computer Laboratory is that of finding linear least squares solutions. These problems arise in a variety of areas and in a variety of contexts. Linear least squares problems are particularly difficult to solve because they frequently involve large quantities of data, and they are ill-conditioned by their very nature. In this paper, we shall consider stable numerical methods for handling these problems. Our basic tool is a matrix decomposition based on orthogonal Householder transformations.

 This report was supported in part by Office of Naval Research Contract Nonr-225(37) (NR 044-11) at Stanford University.

1. Introduction.

Let A be a given $m \times n$ real matrix of rank r, and $\underline{b}$ a given vector. We wish to determine a vector $\hat{\underline{x}}$ such that

$$\| \underline{b} - A \hat{\underline{x}} \| = \min. \tag{1.1}$$

where $\| \cdots \|$ indicates the euclidean norm. If $m \geq n$ and $r < n$ then there is no unique solution. Under these conditions, we require simultaneously to (1.1) that

$$\| \hat{\underline{x}} \| = \min. \tag{1.2}$$

Condition (1.2) is a very natural one for many statistical and numerical problems.

If $m \geq n$ and $r = n$, then it is well known (cf. [4]) that $\hat{\underline{x}}$ satisfies the equation

$$A^T A \underline{x} = A^T \underline{b} \tag{1.3}$$

Unfortunately, the matrix $A^T A$ is frequently ill-conditioned [6] and influenced greatly by roundoff errors. The following example of Läuchli [3] illustrates this well. Suppose

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ \epsilon & 0 & 0 & 0 & 0 \\ 0 & \epsilon & 0 & 0 & 0 \\ 0 & 0 & \epsilon & 0 & 0 \\ 0 & 0 & 0 & \epsilon & 0 \\ 0 & 0 & 0 & 0 & \epsilon \end{bmatrix} ,$$

then

$$A^TA = \begin{bmatrix} 1+\epsilon^2 & 1 & 1 & 1 & 1 \\ 1 & 1+\epsilon^2 & 1 & 1 & 1 \\ 1 & 1 & 1+\epsilon^2 & 1 & 1 \\ 1 & 1 & 1 & 1+\epsilon^2 & 1 \\ 1 & 1 & 1 & 1 & 1+\epsilon^2 \end{bmatrix} \qquad (1.4)$$

Clearly for $\epsilon \neq 0$, the rank of A^TA is five since the eigenvalues of A^TA are $5+\epsilon^2, \epsilon^2, \epsilon^2, \epsilon^2, \epsilon^2$.

Let us assume that the elements of A^TA are computed using double precision arithmetic, and then rounded to single precision accuracy. Now let η be the largest number on the computer such that $fl(1.0+\eta) \equiv 1.0$ where $fl(...)$ indicates the floating point computation. Then if $\epsilon < \frac{\sqrt{\eta}}{2}$, the rank of the computed representation of (1.4) will be one. Consequently, no matter how accurate the linear equation solver, it is impossible to solve the normal equations (1.3).

In [2], Householder stressed the use of orthogonal transformations for solving linear least squares problems. In this paper, we shall exploit these transformations and show their use in a variety of least squares problems.

2. A Matrix Decomposition.

Throughout this section, we shall assume $m \geq n = r$.

Since the euclidean norm of a vector is unitarily invariant,

$$\| \underline{b} - A \underline{x} \| = \| \underline{c} - Q A \underline{x} \|$$

where $\underline{c} = Q\underline{b}$ and Q is an orthogonal matrix. We choose Q so that

$$QA = R = \begin{pmatrix} \tilde{R} \\ \cdots \\ 0 \end{pmatrix} \Big\} (m-n) \times n \qquad (2.1)$$

where $\tilde{R}$ is an upper triangular matrix.

Clearly,

$$\hat{\underline{x}} = R^{-1} \tilde{\underline{c}}$$

where $\tilde{\underline{c}}$ is the first n components of $\underline{c}$ and consequently,

$$\| \underline{b} - A \hat{\underline{x}} \| = \left(\sum_{j=m+1}^{n} c_j^2 \right)^{\frac{1}{2}} .$$

Since $\tilde{R}$ is an upper triangular matrix and $\tilde{R}^T \tilde{R} = A^T A$, $\tilde{R}^T \tilde{R}$ is simply the Choleski decomposition of $A^T A$.

There are a number of ways to achieve the decomposition (2.1); e.g., one could apply a sequence of plane rotations to annihilate the elements below the diagonal of A. A very effective method to realize the decomposition (2.1) is via Householder transformations [2]. Let $A = A^{(1)}$, and let $A^{(2)}, A^{(3)}, \ldots, A^{(n+1)}$ be defined as follows:

$$A^{(k+1)} = P^{(k)} A^{(k)} \qquad (k = 1,2, \ldots, n) \ .$$

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

$$P^{(k)} = I - 2\underline{w}^{(k)} \underline{w}^{(k)T}$$

for suitable $\underline{w}^{(k)}$ such that $\underline{w}^{(k)T} \underline{w}^{(k)} = 1$.
A derivation of $P^{(k)}$ is given in [9]. In order to simplify the calculations, we redefine $P^{(k)}$ as follows:

$$P^{(k)} = I - \beta_k \underline{u}^{(k)} \underline{u}^{(k)T}$$

where

$$\sigma_k = \left(\sum_{i=k}^{m} (a_{i,k}^{(k)})^2 \right)^{\frac{1}{2}}$$

$$\beta_k = [\sigma_k(\sigma_k + |a_{k,k}^{(k)}|)]^{-1}$$

$$u_i^{(k)} = 0 \qquad \text{for} \quad i < k$$

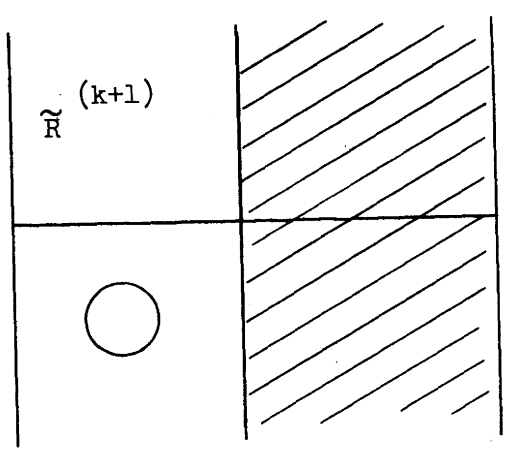
$$u_i^{(k)} = \operatorname{sgn}(a_{k,k}^{(k)}) \ (\sigma_k + |a_{k,k}^{(k)}|)$$

$$u_i^{(k)} = a_{i,k}^{(k)} \qquad \text{for} \quad i > k \ .$$

Thus $$A^{(k+1)} = A^{(k)} - \underline{u}^{(k)} \ (\beta_k \ \underline{u}^{(k)^T} A^{(k)}) \ .$$

After $P^{(k)}$ has been applied to $A^{(k)}$, $A^{(k+1)}$ appears as follows:

$A^{(k+1)} =$



where $\tilde{R}^{(k+1)}$ is a $k \times k$ upper triangular matrix which is unchanged by subsequent transformations. Now $a_{k,k}^{(k+1)} = -(\text{sgn } a_{k,k}^{(k)})\ \sigma_k$ so that the rank of A is less than n if $\sigma_k = 0$. Clearly,

$$R = A^{(n+1)}$$

and

$$Q = P^{(n)}\ P^{(n-1)}\ \ldots\ P^{(1)}$$

although one need not compute Q explicitly.

3. The Practical Procedure.

Wilkinson [10] has shown that the Choleski decomposition is stable for a positive definite matrix even if no interchanges of rows and columns are performed. Since we are in effect performing a Choleski decomposition of A^TA, no interchanges of the columns of A are needed in most situations. However, in order to ensure the utmost accuracy one should choose the columns of A by some strategy. In what follows, we shall refer to the matrix $A^{(k)}$ even if some of the columns have been interchanged.

One possibility is to choose at the k^{th} stage the column of $A^{(k)}$ which will maximize $|a_{k,k}^{(k+1)}|$. This is equivalent to searching for the maximum diagonal element in the Choleski decomposition of A^TA.

Let $$s_j^{(k)} = \sum_{i=k}^{m} (a_{i,j}^{(k)})^2 \qquad \text{for } j = k, k+1, \ldots, n.$$

Then since $|a_{k,k}^{(k+1)}| = \sigma_k$, one should choose that column for which $s_j^{(k)}$ is maximized. After $A^{(k+1)}$ has been computed, one can compute $s_j^{(k+1)}$ as follows:

$$s_j^{(k+1)} = s_j^{(k)} - (a_{k,j}^{(k+1)})^2 \qquad (j = k+1, \ldots, m)$$

since the orthogonal transformations leave the column lengths invariant. Naturally, the $s_j^{(k)}$'s must be interchanged if the columns of $A^{(k)}$ are interchanged. Although it is possible to compute σ_k directly from the $s_j^{(k)}$'s, it is best to compute σ_k at each stage using double precision inner products to ensure maximal accuracy.

The strategy described above is most appropriate when one has a sequence of vectors $\underline{b}_1, \underline{b}_2, \ldots, \underline{b}_p$ for which one desires a least squares estimate. In many problems, there is one vector $\underline{b}$ and one wishes to express it in as few columns of A as possible. This is the stagewise multiple regression problem. We cannot solve this problem, but we shall show how one can choose that column of $A^{(k)}$ for which the sum of squares of residuals is maximally reduced at the k^{th} stage.

Let $\underline{c}^{(1)} = \underline{b}$ and $\underline{c}^{(k+1)} = P^{(k)} \underline{c}^{(k)}$. Now

$$\| \underset{\sim}{c}^{(k)} - R^{(k)} \hat{\underline{x}}^{(k-1)} \| = \left(\sum_{j=k}^{m} (c_j^{(k)})^2 \right)^{\frac{1}{2}}$$

where $\hat{\underline{x}}^{(k-1)}$ is the least squares estimate based on $(k-1)$ columns of A, and $\underset{\sim}{c}^{(k)}$ is the first $(k-1)$ elements of $\underline{c}^{(k)}$. Then since length is preserved

under an orthogonal transformation, we wish to find that column of $A^{(k)}$ which will maximize $|c_k^{(k+1)}|$.

Let $$t_j^{(k)} = \sum_{i=k}^{m} a_{i,j}^{(k)} \; c_i^{(k)} \qquad \text{for } j = k, k+1, \ldots, m.$$

Then since $|c_k^{(k+1)}| = \left| \sum_{i=k}^{m} a_{i,k}^{(k)} \; c_i^{(k)} / \sigma_k \right|$ one should choose that column of $A^{(k)}$ for which $(t_j^{(k)})^2 / s_j^{(k)}$ is maximized. After $P^{(k)}$ is applied to $A^{(k)}$, one can adjust $t_j^{(k)}$ as follows:

$$t_j^{(k+1)} = t_j^{(k)} - a_{k,j}^{(k+1)} \; c_k^{(k+1)} .$$

In many statistical applications, if $(t_j^{(k)})^2 / s^{(k)}$ is sufficiently small then no further transformations are performed.

Once the solution to the equations has been obtained then it is possible to obtain an improved solution by a simple iterative technique. This technique, however, requires that the orthogonal transformations be saved during their application. The best method for storing the transformation is to store the elements of $\underline{u}^{(k)}$ below the diagonal of the k^{th} column of $A^{(k+1)}$.

Let $\bar{\underline{x}}$ be the initial solution obtained, and let $\hat{\underline{x}} = \bar{\underline{x}} + \underline{e}$.

Then $$\| \underline{b} - A \hat{\underline{x}} \| = \| \underline{r} - A \underline{e} \|$$

where $\underline{r} = \underline{b} - A \bar{\underline{x}}$, the residual vector .

Thus the correction vector $\underline{e}$ is itself the solution to a linear least squares problem. Once A has been decomposed then it is a fairly simple matter to compute $\underline{r}$ and solve for $\underline{e}$. Since $\underline{e}$ critically depends upon the residual vector, the components of $\underline{r}$ should be computed using double precision inner products and then rounded to single precision accuracy. Naturally, one should continue to iterate as long as improved estimates of $\hat{\underline{x}}$ are obtained.

The above iteration technique will converge only if the initial approximation to $\hat{\underline{x}}$ is sufficiently accurate. Let

$$\underline{x}^{(q+1)} = \underline{x}^{(q)} + \underline{e}^{(q)} \qquad (q = 0, 1, \ldots)$$

with $\underline{x}^{(0)} = \underline{0}$.

Then if $\| \underline{e}^{(1)} \| / \| \underline{x}^{(1)} \| > c$ and if $c < 1/2$, i.e., "at least one bit of the initial solution is correct," one should not iterate since there is little likelihood that the iterative method will converge. Since convergence tends to be linear, one should terminate the procedure as soon as $\| \underline{e}^{(k+1)} \| > c \| \underline{e}^{(k)} \|$.

4. A Numerical Example.

In Table I, we give the results of an extensive calculation. The matrix consists of the first 5 columns of the inverse of the 6×6 Hilbert matrix. The calculations were performed in single precision arithmetic. The columns were chosen so that the diagonal elements were maximized at each stage. The iteration procedure was terminated as soon as $\| \underline{e}^{(k+1)} \| > 0{,}25 \| \underline{e}^{(k)} \|$. Three iterations were performed but since $\| \underline{e}^{(2)} \| > 0.25 \| \underline{e}^{(1)} \|$, $\underline{x}^{(2)}$ was taken to be the correct solution.

In Table II, we show the results of using double precision inner products on the same problem. Note that the first iterate in Table I is approximately as accurate as the first iterate in Table II. The double precision inner product routine converged to a solution for which all figures were accurate. The normal equations were formed using double precision inner products but even with a very accurate linear equation solver described by McKeeman [5] no solution could be obtained.

TABLE I

	A				B
3.6000000000D+01	-6.3000000000D+02	3.3600000000D+03	-7.5600000000D+03	7.5600000000D+03	4.6300000000D+02
-6.3000000000D+02	1.4700000000D+04	-8.8200000000D+04	2.1168000000D+05	-2.2050000000D+05	-1.3860000000D+04
3.3600000000D+03	-8.8200000000D+04	5.6448000000D+05	-1.4112000000D+06	1.5120000000D+06	9.7020000000D+04
-7.5600000000D+03	2.1168000000D+05	-1.4112000000D+06	3.6288000000D+06	-3.9690000000D+06	-2.5872000000D+05
7.5600000000D+03	-2.2050000000D+05	1.5120000000D+06	-3.9690000000D+06	4.4100000000D+06	2.9106000000D+05
-2.7720000000D+03	8.3160000000D+04	-5.8212000000D+05	1.5523200000D+06	-1.7463600000D+06	-1.1642400000D+05

PIVOT

5	4	3	2	1	
		R			
-6.3706872199D+06	5.7760446368D+06	-2.2224495814D+06	3.2875491420D+05	-1.1522407952D+04	
	-6.7135626821D+04	6.0308471363D+04	-1.6102552917D+04	9.4910780925D+02	
		-1.4425503500D+03	9.4707042643D+02	-1.0982644636D+02	
			4.6175131643D+01	-1.4949038075D+01	
				2.0095559062D+00	

ITERATION

1						
	X	9.9999912362D-01	4.9999972493D-01	3.3333322021D-01	2.4999995198D-01	1.9999998332D-01	
	R	1.4528632164D-06	-4.7683715820D-07	0.0000000000D+00	0.0000000000D+00	0.0000000000D+00	0.0000000000D+00
	E	5.8284221383D-07	1.7292551586D-07	6.9381181279D-08	2.9087566395D-08	1.0038052093D-08	
2							
	X	9.9999970646D-01	4.9999989786D-01	3.3333328959D-01	2.4999998107D-01	1.9999999336D-01	
	R	3.2037496567D-07	4.7683715820D-07	3.8146972656D-06	-3.8146972656D-06	3.8146972656D-06	0.0000000000D+00
	E	-3.7819874906D-07	-1.1487774268D-07	-4.6569725113D-08	-1.9660767363D-08	-6.8227426601D-09	

TABLE II

		PIVOT 5	4	3	2	1	
				R			
		-6.3706872199@+06	5.7760446368@+06	-2.2224495814@+06	3.2875491420@+05	-1.1522407952@+04	
			-6.7135626821@+04	6.0308471363@+04	-1.6102552917@+04	9.4910780925@+02	
				-1.4425503500@+03	9.4707042643@+02	-1.0982644637@+02	
					4.6175131642@+01	-1.4949038077@+01	
						2.0095559061@+00	
				ITERATION			
1							
	X	9.9999912347@-01	4.9999972435@-01	3.3333321985@-01	2.4999995180@-01	1.9999998325@-01	
	R	1.4480865502@-06	-8.8621163741@-08	9.3205017038@-08	-2.6189081836@-06	2.1314917831@-07	3.2270327211@-07
	E	8.7653116638@-07	2.7564789022@-07	1.1348434536@-07	4.8201409291@-08	1.6752167940@-08	
2							
	X	1.0000000000@+00	5.0000000000@-01	3.3333333333@-01	2.5000000000@-01	2.0000000000@-01	
	R	-7.5378920883@-09	2.1391315386@-07	-1.4423858374@-06	3.7434801925@-06	-4.1254679672@-06	1.6236008378@-06
	E	9.4773720382@-18	2.6923536909@-18	-6.0632875784@-13	4.2556290006@-19	-7.2759561775@-13	
3							
	X	1.0000000000@+00	5.0000000000@-01	3.3333333333@-01	2.5000000000@-01	2.0000000000@-01	
	R	-7.5378920883@-09	2.1391315386@-07	-1.4423858374@-06	3.7434801925@-06	-4.1254679672@-06	1.6236008378@-06
	E	9.4773720382@-18	2.6923536909@-18	-6.0632875784@-13	4.2556290006@-19	-7.2759561775@-13	

5. An Iterative Scheme.

For many problems, even with the use of orthogonal transformations it may be impossible to obtain an accurate solution. Or, the rank of A may truly be less than n. In this section, we give an algorithm for finding the least squares solution even if A^TA is singular.

In [7], Riley suggested the following algorithm for solving linear least squares problems for $r = n$. Let $\underline{x}^{(0)}$ be an arbitrary vector, then solve

$$(A^TA + \alpha I)\, \underline{x}^{(q+1)} = A^T \underline{b} + \alpha\, \underline{x}^{(q)} \tag{5.1}$$

The sequence $\underline{x}^{(q)}$ converges to $\hat{\underline{x}}$ if $\alpha > 0$ since the spectral radius of $\alpha(A^TA + \alpha I)^{-1}$ is less than 1. Again we may implement this algorithm more effectively by the use of orthogonal transformations.

First, let us note that (5.1) is equivalent to the following:

$$\underline{r}^{(q)} = \underline{b} - A\, \underline{x}^{(q)} \tag{5.2a}$$

$$(A^TA + \alpha I)\, \underline{e}^{(q)} = A^T\, \underline{r}^{(q)} \tag{5.2b}$$

$$\underline{x}^{(q+1)} = \underline{x}^{(q)} + \underline{e}^{(q)} \tag{5.2c}$$

The vector $\underline{e}^{(q)}$ is itself the solution of a linear least squares problem since $\underline{e}^{(q)}$ minimize $\| \underline{d}^{(q)} - C\, \underline{e}^{(q)} \|$

where

$$C = \begin{pmatrix} A \\ \cdots\cdots \\ \sqrt{\alpha}\, I \end{pmatrix}, \quad \underline{d}^{(q)} = \begin{pmatrix} \underline{r}^{(q)} \\ \cdots\cdots \\ \underline{0} \end{pmatrix}.$$

Thus the numerical procedure should go as follows. Decompose C by the methods described in Section 2 so that

$$PC = S = \begin{pmatrix} \tilde{S} \\ \cdots\cdots \\ \bigcirc \end{pmatrix}$$

where $P^TP = I$ and $\tilde{S}$ is an upper triangular matrix. Then let $\underline{x}^{(0)} = \underline{0}$,

$$\tilde{S}\,\underline{e}^{(q)} = \tilde{\underline{f}}^{(q)}$$

$$\underline{x}^{(q+1)} = \underline{x}^{(q)} + \underline{e}^{(q)}$$

and $\tilde{\underline{f}}^{(q)}$ is the vector whose components are the first n components of $P\,\underline{d}^{(q)}$. We choose $\underline{x}^{(0)} = \underline{0}$ since otherwise there is no assurance that $\underline{x}^{(q)}$ will converge to $\hat{\underline{x}}$.

Now going back to the original process (5.1),

$$\underline{x}^{(q+1)} = G\,\underline{x}^{(q)} + \underline{h} \tag{5.3}$$

where $G = \alpha(A^TA + \alpha I)^{-1}$ and $\underline{h} = (A^TA + \alpha I)^{-1} A^T \underline{b}$.

Thus

$$\underline{x}^{(q+1)} = (G^q + G^{q-1} + \ldots + I)\,\underline{h} \tag{5.4}$$

It is well known (cf. [6]) that A may be written as

$$A = U \Sigma V^T$$

where Σ is an $m \times n$ matrix with the singular values σ_j on the diagonal and zeros elsewhere, and U and V are the matrices of eigenvectors of AA^T and A^TA, respectively. Then

$$A^T \underline{b} = V \Sigma^T U^T \underline{b} = \beta_1 \sigma_1 \underline{v}_1 + \beta_2 \sigma_2 \underline{v}_2 + \cdots + \beta_r \sigma_r \underline{v}_r$$

where $\underline{\beta} = U^T \underline{b}$, and r is the rank of A. Then from (5.4) we see that

$$\underline{x}^{(q)} = \gamma_1^{(q)} \underline{v}_1 + \cdots + \gamma_r^{(q)} \underline{v}_r$$

where $$\gamma_j^{(q)} = \left[1 - \left(\frac{\alpha}{\alpha+\sigma_j^2}\right)^q\right] \frac{\beta_j}{\sigma_j} \qquad (j = 1, 2, \ldots, r)$$

Thus as $q \to \infty$

$$\underline{x}^{(q)} \to \frac{\beta_1}{\sigma_1} \underline{v}_1 + \cdots + \frac{\beta_r}{\sigma_r} \underline{v}_r = \hat{\underline{x}} \quad .$$

The choice of α will greatly affect the rate of convergence of the iterative method, and thus one must choose α with great care. If α is too small then the equations will remain ill-conditioned. If δ is a lower

bound of the smallest non-zero singular value, then α should be chosen so that

$$\frac{\alpha}{\alpha + \delta^2} < 0.1, \quad \text{say} .$$

This means at each stage, there will be at least one more place of accuracy in the solution. There are a number of methods for accelerating the convergence of (5.1) (cf. [1]).

It is easy to see that

$$\underline{e}^{(q+1)} = G\,\underline{e}^{(q)} = \alpha(A^TA + \alpha I)^{-1}\,\underline{e}^{(q)} .$$

Since $\underline{e}^{(q)}$ lies in the space spanned by $\underline{v}_1, \ldots, \underline{v}_r$, it follows immediately that

$$\| \underline{e}^{(q+1)} \| \leq \frac{\alpha}{\alpha + \sigma_r^2} \| \underline{e}^{(q)} \| < \| \underline{e}^{(q)} \| .$$

Thus a good termination procedure is to stop iterating as soon as $\| \underline{e}^{(q)} \|$ increases.

6. Statistical Calculations.

In many statistical calculations, it is necessary to compute certain auxilliary information associated with A^TA. These can readily be obtained from the orthogonal decomposition. Thus

$$\det(A^TA) = (r_{11} \times r_{22} \times \cdots \times r_{nn})^2 \quad .$$

Since $A^TA = \tilde{R}^T \tilde{R}$, $(A^TA)^{-1} = \tilde{R}^{-1} \tilde{R}^{-T}$.

The inverse of $\tilde{R}$ can be readily obtained since $\tilde{R}$ is an upper triangular matrix. Waugh and Dwyer [8] have noted that it is possible to calculate $(A^TA)^{-1}$ directly from $\tilde{R}$ by the relation

$$\tilde{R}\,(AA^T)^{-1} = \tilde{R}^{-T} \quad .$$

No operations are saved over the first method but it may be somewhat more accurate.

In some statistical applications, the original set of observations are augmented by an additional set of observations. In this case, it is not necessary to begin the calculation from the beginning again if the method of orthogonalization is used. Let $\tilde{R}_1$, $\tilde{\underline{c}}_1$ correspond to the original data after it has been reduced by orthogonal transformations and let $A_2, \underline{b}_2$ correspond to the additional observations. Then the up-dated least squares solution can be obtained directly from

$$A = \begin{pmatrix} \tilde{R}_1 \\ \cdots\cdots \\ A_2 \end{pmatrix} \quad , \quad \underline{b} = \begin{pmatrix} \tilde{\underline{c}}_1 \\ \cdots\cdots \\ \underline{b}_2 \end{pmatrix} \quad .$$

The above observation has another implication. One of the arguments frequently advanced for using normal equations is that only $n(n+1)/2$ memory locations are required. By partitioning the matrix A by rows, however, then similarly only $n(n+1)/2$ locations are needed when the method of orthogonalization is used.

7. Least Squares Problems with Constraints.

Frequently, one wishes to determine $\hat{\underline{x}}$ so that $\| \underline{b} - A \hat{\underline{x}} \|$ is minimized subject to the condition that $H \hat{\underline{x}} = g$ where H is a $p \times n$ matrix of rank p. One can, of course, eliminate p of the columns of A by Gaussian elimination after a $p \times p$ submatrix of H has been determined and then solve the resulting normal equations. This, unfortunately, would not be a numerically stable scheme since no row interchanges between A and H would be permitted.

If one uses Lagrange multipliers, then one must solve the $(n + p) \times (n + p)$ system of equations

$$\begin{pmatrix} A^TA & H^T \\ H & \bigcirc \end{pmatrix} \begin{pmatrix} \hat{\underline{x}} \\ \cdots \\ \underline{\lambda} \end{pmatrix} = \begin{pmatrix} A^T \underline{b} \\ \cdots\cdots \\ \underline{g} \end{pmatrix} .$$

where $\underline{\lambda}$ is the vector of Lagrange multipliers. Since $\hat{\underline{x}} = (A^TA)^{-1} A^T \underline{b} - (A^TA)^{-1} H^T \underline{\lambda}$,

$$H(A^TA)^{-1} H^T \underline{\lambda} = H \underline{z} - g$$

where

$$\underline{z} = (A^TA)^{-1} A^T \underline{b} .$$

Note $\underline{z}$ is the least squares solution of the original problem without constraints and one would frequently wish to compare this vector with the final solution $\hat{\underline{x}}$. The vector $\underline{z}$, of course, should be computed by the orthogonalization procedures discussed earlier.

Since $A^TA = \tilde{R}^T\tilde{R}$, $H(A^TA)^{-1}H^T = W^TW$ where $W = \tilde{R}^{-T}H^T$. After W is computed, it should be reduced to a $p \times p$ upper triangular matrix K by orthogonalization which is the Choleski decomposition of W^TW. The matrix equation

$$K^TK\,\underline{\lambda} = H\,\underline{z} - g$$

should be solved by the obvious method. Finally, one finds

$$\underline{\hat{x}} = \underline{z} - (A^TA)^{-1}H\,\underline{\lambda}$$

where $(A^TA)^{-1}H\,\underline{\lambda}$ can be easily computed by using $\tilde{R}^{-1}$.

Acknowledgments

The author is very pleased to acknowledge the programming efforts of Mr. Peter Businger and Mr. Alan Merten. He also wishes to thank Professor Thomas Robertson for his critical remarks.

REFERENCES

[1] G. H. Golub and R. S. Varga, "Chebyshev Semi-Iterative Methods, Successive Over-Relaxation Iterative Methods, and Second Order Richardson Iterative Method," Numer. Math., Vol. 3 (1961) pp. 147-168.

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

[3] P. Läuchli, "Jordan-Elimination und Ausgleichung mach kleinsten quadraten," Numer. Math., Vol. 3 (1961) pp. 226-240.

[4] Y. Linnik, Method of Least Squares and Principles of the Theory of Observations, translated from Russian by R. C. Elandt, Pergamon Press, New York, 1961.

[5] W. M. McKeeman, "Crout with Equilibration and Iteration." Algorithm 135 Comm. Assoc. Comput. Mach., Vol. 5 (1962) pp. 553-555.

[6] E. E. Osborne, "On Least Squares Solutions of Linear Equations," J. Assoc. Comput. Mach., Vol. 8 (1961) pp. 628-636.

[7] J. D. Riley, "Solving Systems of Linear Equations with a Positive Definite, Symmetric, but Possibly Ill-Conditioned Matrix," Math. Tables Aids Comput., Vol. 9 (1956) pp. 96-101.

[8] F. V. Waugh and P. S. Dwyer, "Compact Computation of the Inverse of a Matrix," Ann. Math. Stat., Vol. 16 (1945) pp. 259-271.

[9] J. H. Wilkinson, "Householder's Method for the Solution of the Algebraic Eigenproblem," Comput. J., Vol. 3 (1960) pp. 23-27

[10] J. H. Wilkinson, "Error Analysis of Direct Methods of Matrix Inversion," J. Assoc. Comput. Mach., Vol. 8 (1961) pp. 281-330.

AN ALGOL PROCEDURE FOR FINDING LINEAR LEAST SQUARES SOLUTIONS

by

Peter Businger*/

*/ Present Address: Computing Center, University of Texas, Austin, Texas

```
procedure least squares solution (a, x, b, m, n, p, singular) ;
value m, n, p ;
array a, x, b ; integer m, n, p ; label singular ;
comment The array a[1:m,1:n] contains the given matrix of an
overdetermined system of m linear equations in n unknowns (m ≥ n).
For the p right hand sides given as the columns of the array
b[1:m,1:p], the least squares solutions are computed and stored
as the columns of the array x[1:n,1:p]. If rank(a) < n then the
problem is left unsolved and the emergency exit singular is used.
In either case a and b are left intact ;
begin
   real procedure inner product (i, m, n, a, b, c) ;
   value m, n, c ;
   real a, b, c ; integer i, m, n ;
   comment The body of this inner product routine should pre-
   ferably be replaced by its double precision equivalent in
   machine code ;
   begin
      for i := m step 1 until n do c := c + a×b ;
      inner product := c
   end inner product ;
   procedure decompose (m, n, qr, alpha, pivot, singular) ;
   value m, n ;
   integer m, n ; array qr, alpha ; integer array pivot ;
   label singular ; comment nonlocal real procedure inner product ;
   comment decompose reduces the matrix given in the array
   qr[1:m,1:n] (m ≥ n) to upper right triangular form by means
```

of n elementary orthogonal transformations (I-2ww') = (I-beta uu'). The diagonal elements of the reduced matrix are stored in the array alpha[1:n], the off diagonal elements in the upper right triangular part of qr. The non-zero components of the vectors u are stored on and below the leading diagonal of qr. Pivoting is done by choosing at each step the column with the largest sum of squares to be reduced next. These interchanges are recorded in the array pivot[1:n]. If at any stage of the reduction the sum of squares of the column to be reduced next is exactly equal to zero then the emergency exit singular is used ;

```
begin
   integer i, j, jbar, k ; real beta, sigma, alphak, qrkk ;
   array y, sum[1:n] ;
   for j := 1 step 1 until n do
   begin
      pivot[j] := j ;
      sum[j] := inner product (i, 1, m, qr[i,j], qr[i,j], 0)
   end j ;
   for k := 1 step 1 until n do
   begin
      sigma := sum[k] ; jbar := k ;
      for j := k+1 step 1 until n do if sigma < sum[j] then
      begin
         sigma := sum[j] ; jbar := j
      end ;
      if jbar≠k then
```

```
begin
   i := pivot[k] ;
   pivot[k] := pivot[jbar] ;
   pivot[jbar] := i ;
   sum[jbar] := sum[k] ;
   sum[k] := sigma ;
   for i := 1 step 1 until m do
   begin
      sigma := qr[i,k] ;
      qr[i,k] := qr[i,jbar] ;
      qr[i,jbar] := sigma
   end i
end ;
sigma := inner product (i,k,m, qr[i,k], qr[i,k], 0) ;
if sigma=0 then goto singular ;
qrkk := qr[k,k] ;
alphak := alpha[k] := if qrkk < 0 then sqrt(sigma)
                                  else -sqrt(sigma) ;
qr[k,k] := qrkk-alphak ;
beta := 1/(sigma-qrkk×alphak) ;
for j := k+1 step 1 until n do
y[j] := beta × inner product (i, k, m, qr[i,k], qr[i,j], 0) ;
for j := k+1 step 1 until n do
begin
   for i := k step 1 until m do
   qr[i,j] := qr[i,j] -qr[i,k]×y[j] ;
   sum[j] = sum[j]-qr[k,j]↑2
```

```
      end j
   end k
end decompose ;
procedure solve (m, n, qr, alpha, pivot, r, y) ;
value m, n ;
integer m, n ; array qr, alpha, r, y ; integer array pivot ;
comment nonlocal real procedure inner product ;
comment Using the vectors u whose nonzero components are
stored on and below the main diagonal of qr[1:m,1:n] solve
applies the n elementary orthogonal transformations (I-2ww')
to the right hand side r[1:m]. From the reduced matrix given
in alpha[1:n] and the upper right triangular part of qr,
solve then computes by backsubstitution an approximate solution
to the linear system. The components of the solution vector
are stored in y[1:n] in the order prescribed by pivot[1:n] ;
begin
   integer i,j ; real gamma ; array z[1:n] ;
   for j := 1 step 1 until n do
   begin
      gamma := inner product (i, j, m, qr[i,j], r[i], 0)
               /(alpha[j]×qr[j,j]) ;
      for i := j step 1 until m do r[i] := r[i]+gamma×qr[i,j]
   end j ;
   z[n] := r[n]/alpha[n] ;
   for i := n-1 step -1 until 1 do
   z[i] := -inner product (j, i+1, n, qr[i,j], z[j], -r[i])
           /alpha[i] ;
```

```
    for i := 1 step 1 until n do y[pivot[i]] := z[i]
  end solve ;
  integer i, j, k ; real norm0, norm1 ;
  array qr[1:m,1:n], alpha, e, y[1:n], r[1:m] ;
  integer array pivot[1:n] ;
  for j := 1 step 1 until n do for i := 1 step 1 until m do
  qr[i,j] := a[i,j] ;
  decompose (m, n, qr, alpha, pivot, singular) ;
  for k := 1 step 1 until p do
  begin
    for i := 1 step 1 until m do r[i] := b[i,k] ;
    solve (m, n, qr, alpha, pivot, r, y) ;
    for i := 1 step 1 until m do
    r[i] = -inner product (j, 1, n, a[i,j], y[j], -b[i,k]) ;
    solve (m, n, qr, alpha, pivot, r, e) ;
    norm0 := norm1 := 0 ;
    for i := 1 step 1 until n do
    begin
      norm0 = norm0+y[i]↑2 ; norm1 = norm1+e[i]↑2
    end i ;
    if norm1 > 0.0625×norm0 then goto singular ; comment No
    attempt at obtaining the solution is made unless the norm
    of the first correction is significantly smaller than the
    norm of the initial approximation ;
iterate:
    for i := 1 step 1 until n do y[i] := y[i]+e[i] ;
    for i := 1 step 1 until m do
```

```
        r[i] := -inner product (j, 1, n, a[i,j], y[j], -b[i,k]) ;
        solve (m, n, qr, alpha, pivot, r, e) ;
        norm0 := norm1 ; norm1 := 0 ;
        for i := 1 step 1 until n do norm1 := norm1+e[i]↑2 ;
        if norm1≤0.0625×norm0 then goto iterate ; comment iterative
        improvement of the solution is terminated as soon as the
        norm of a correction is not significantly smaller than the
        norm of the previous correction ;
        for i := 1 step 1 until n do x[i,k] = y[i]
    end k
end least squares solution
```