NUMERICAL TECHNIQUES IN MATHEMATICAL PROGRAMMING

BY

R. H. BARTELS
G. H. GOLUB
M. A. SAUNDERS

STAN-CS-70-162
MAY 1970

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

NUMERICAL TECHNIQUES IN MATHEMATICAL PROGRAMMING

by

R. H. Bartels*

G. H. Golub**

M. A. Saunders***

* University of Texas, Austin, Texas.

** Stanford University, Stanford, California. The work of this author was in part supported by the U. S. Atomic Energy Commission.

*** Stanford University, Stanford, California. The work of this author was in part supported by the U. S. Atomic Energy Commission and by the Department of Scientific and Industrial Research, Wellington, New Zealand.

The preparation of this report was sponsored by the Office of Naval Research under grant number N0013-67-A-0112-0029, the National Science Foundation under grant number NSF GJ 408 and the Atomic Energy Commission under grant number AT (04-3) 326,PA 30.

NUMERICAL TECHNIQUES IN MATHEMATICAL PROGRAMMING

by

R. H. Bartels, G. H. Golub, M. A. Saunders

Abstract

The application of numerically stable matrix decompositions to minimization problems involving linear constraints is discussed and shown to be feasible without undue loss of efficiency.

Part A describes computation and updating of the product-form of the LU decomposition of a matrix and shows it can be applied to solving linear systems at least as efficiently as standard techniques using the product-form of the inverse.

Part B discusses orthogonalization via Householder transformations, with applications to least squares and quadratic programming algorithms based on the principal pivoting method of Cottle and Dantzig.

Part C applies the singular value decomposition to the nonlinear least squares problem and discusses related eigenvalue problems.

Table of Contents

Introduction

This paper describes the application of numerically stable matrix decompositions to minimization problems involving linear constraints. Algorithms for solving such problems are fundamentally techniques for the solution of selected systems of linear equations, and during the last fifteen years there has been a major improvement in the understanding of these and other linear algebraic problems. We show here that methods which have been analysed by various workers and proven to be numerically stable may be employed in mathematical programming algorithms without undue loss of efficiency.

Part A describes means for computing and updating the product-form of the LU decomposition of a matrix. The solution of systems of equations by this method is shown to be stable and to be at least as efficient as standard techniques which use the product-form of the inverse.

In Part B we discuss orthogonalization via Householder transformations. Applications are given to least squares and quadratic programming algorithms based on the principal pivoting method of Cottle and Dantzig [5]. For further applications of stable methods to least squares and quadratic programming, reference should be made to the recent work of R. J. Hanson [13] and of J. Stoer [26] whose algorithms are based on the gradient projection method of J. B. Rosen [24].

In Part C the application of the singular value decomposition to the nonlinear least squares problem is discussed, along with related eigenvalue problems.

A. THE USE OF LU DECOMPOSITION IN EXCHANGE ALGORITHMS

1. LU Decomposition

If B is an n X n , nonsingular matrix, there exists a permutation matrix π , a lower-triangular matrix L with ones on the diagonal, and an upper-triangular matrix U such that

(1.1) $$\pi B = LU .$$

It is possible to choose π , L , and U so that all elements of L are bounded in magnitude by unity.

A frequently-used algorithm for computing this decomposition is built around Gaussian elimination with row interchanges. It produces the matrices π and L in an implicit form as shown:

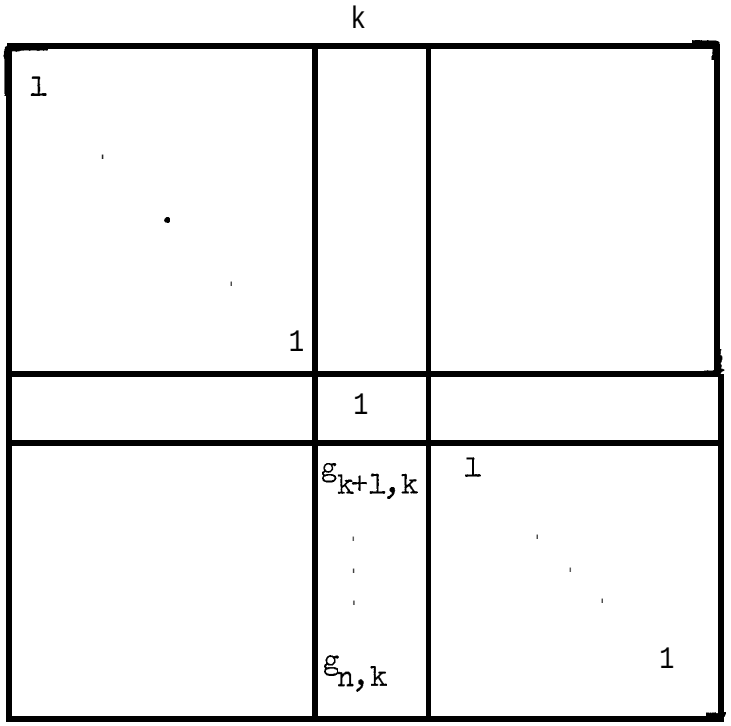
For k = 1,2,...,n-1 in order carry out the following two steps:

(1.2) . Find an element in the k-th column of B , on or below the diagonal, which has maximal magnitude. Interchange the k-th row with the row of the element found.

(1.3) Add an appropriate multiple of the resulting k-th row to each row below the k-th in order to create zeros below the diagonal in the k-th column.

Each execution of the first step (1.2), in matrix notation, amounts to the premultiplication of B by a suitable permutation matrix π_k. The following step (1.3) may be regarded as the premultiplication of B by a matrix Γ_k of the form

(1.4)



,

where $|g_{i,k}| \leq 1$ for each $i = k+1,\ldots,n$.

By repeating the two steps n-1 times, B is transformed into U. And at the same time the matrix $(L^{-1}\pi)$ is collected in product form

$$L^{-1}\pi = \Gamma_{n-1}\pi_{n-1} \cdots \Gamma_1\pi_1 . \tag{1.5}$$

This algorithm requires $n^3/3 + O(n^2)$ multiplication/division operations and again this many addition/subtraction operations. Both U and all of the $g_{i,j}$ can be stored in the space which was originally occupied by B. An additional n locationsare required for the essential information contained in the π_k.

2. Exchange Algorithms

Many algorithms require the solving of a sequence of linear equations

$$B^{(i)}x = v^{(i)} \tag{2.1}$$

for which each $B^{(i)}$ differs from its predecessor in only one column. Examples of such algorithms are: the simplex method, Stiefel's exchange method for finding a Chebyshev solution to an overdetermined linear equation system, and adjacent- path methods for solving the complementary-pivot programming problem.

Given that $B^{(0)}$ has a decomposition of the form

$$B^{(0)} = L^{(0)}U^{(0)} \quad , \tag{2.2}$$

where $U^{(0)}$ is upper-triangular, and given that $L^{(0)^{-1}}$ has been stored as a product

$$L^{(0)^{-1}} = \Gamma_{n-1}^{(0)} \Pi_{n-1}^{(0)} \cdots \Gamma_1^{(0)} \Pi_1^{(0)} \quad , \tag{2.3}$$

the initial system of the sequence is readily solved: Set

$$y = L^{(0)^{-1}} v^{(0)} \quad , \tag{2.4}$$

and then back-solve the triangular system

$$U^{(0)}x = y \quad . \tag{2.5}$$

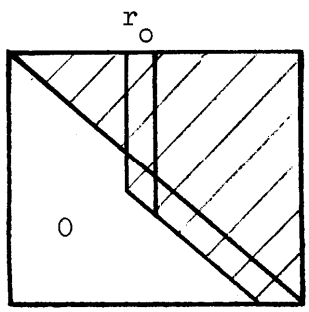
3. Updating the LU Decomposition

Let the column r_o of $B^{(0)}$ be replaced by the column vector $a^{(0)}$. So long as we revise the ordering of the unknowns accordingly, we may insert $a^{(0)}$ into the last column position, shifting columns r_o+1 through n of $B^{(0)}$ one position to the left to make room. We will call the result $B^{(1)}$, and we can easily check that it has the decomposition

$$B^{(1)} = L^{(0)}H^{(1)} \quad , \tag{3.1}$$

where $H^{(1)}$ is a matrix which is upper-Hessenberg in its last $n - r_o + 1$ columns and upper-triangular in its first r_o-1 columns. That is, $H^{(1)}$ has the form

(3.2)



.

The first r_o-1 columns of $H^{(1)}$ are identical with those of $U^{(0)}$. The next $n-r_o$ are identical with the last $n-r_o$ columns of $U^{(0)}$. And the last column of $H^{(1)}$ is the vector $L^{(0)^{-1}}a^{(0)}$.

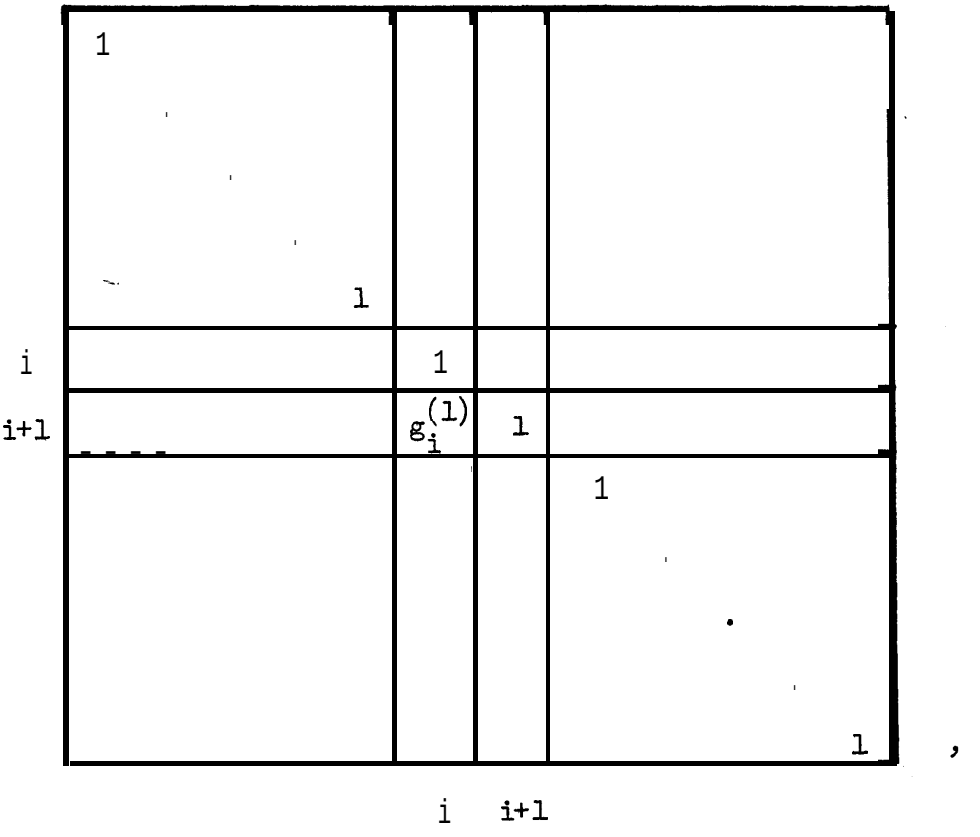
$H^{(1)}$ can be reduced to upper-triangular form by Gaussian elimination with row interchanges. Here, however, we need only concern ourselves with the interchanges of pairs of adjacent rows. Thus $U^{(1)}$ is gotten

from $H^{(1)}$ by applying a sequence of simple transformations:

$$U^{(1)} = \Gamma^{(1)}_{n-1} \pi^{(1)}_{n-1} \cdots \Gamma^{(1)}_{r_o} \pi^{(1)}_{r_o} H^{(1)} \quad , \tag{3.3}$$

where each $\Gamma^{(1)}_i$ has the form

(3.4)



and each $\pi^{(1)}_i$ is either the identity matrix or the identity with the i-th and i+1-st rows exchanged, the choice being made so that $|g_i^{(1)}| \leq 1$.

The essential information in all of these transformations can be stored in $n-r_o$ locations plus an additional n-ro bits (to indicate the interchanges). If we let

$$(3.5) \qquad L^{(1)^{-1}} = \Gamma_{n-1}^{(1)} \pi_{n-1}^{(1)} \cdots \Gamma_{r_o}^{(1)} \pi_{r_o}^{(1)} L^{(0)^{-1}} \quad ,$$

then we have achieved the decomposition

$$(3.6) \qquad B^{(1)} = L^{(1)} U^{(1)}$$

The transition from $B^{(i)}$ to $B^{(i+1)}$ for any i is to be made exactly as was the transition from $B^{(0)}$ to $B^{(1)}$. Any system of linear equations involving the matrix $B^{(i)}$ for any i is to be solved according to the steps given in (2.4) and (2.5).

4. Round-off Considerations

For most standard computing machines the errors in the basic arithmetic operations can be expressed as follows:

$$
\begin{aligned}
fl(a \pm b) &= a(1 + \varepsilon_1) \pm b(1 + \varepsilon_2) \\
fl(a \times b) &= ab(1 + \varepsilon_3) \qquad (4.1) \\
fl(a/b) &= (a/b)(1 + \varepsilon_4) \quad ,
\end{aligned}
$$

where $|\varepsilon_i| < \beta^{1-t}$. Here β stands for the base of the number system in which machine arithmetic is carried out and t is the number of significant figures which the machine retains after each operation. The notation $fl(a \text{ "op" } b)$ stands for the result of the operation "op" upon the two, normal-precision floating-point numbers a and b when standard floating-point arithmetic is used.

The choice of an LU decomposition for each $B^{(i)}$ and the particular way in which this decomposition is updated were motivated by the desire to find a way of solving a sequence of linear equations (2.1) which would retain a maximum of information from one stage to the next in the sequence and which would be as little affected by round-off errors as possible. Under the assumption that machine arithmetic behaves as given in (4.1), the processes described in Sections 2 and 3 are little affectedby round-off errors. The efficiency of the processes will vary from algorithm to algorithm, but we will argue in a subsequent section that the processes should cost roughly as much as those based upon product-form inverses of the $B^{(i)}$.

We will now consider the round-off properties of the basic steps described in Sections 2 and 3.

The computed solution to the triangular system of linear equations

$$U^{(i)}x = y \tag{4.1}$$

can be shown, owing to round-off errors, to satisfy a perturbed system

$$(U^{(i)} + \delta U^{(i)})x = y \quad . \tag{4.3}$$

It is shown in Forsythe and Moler [9] that

$$\frac{\|\delta U^{(i)}\|}{\|U^{(i)}\|} \leq \frac{n(n+1)}{2}\,(1.01)\beta^{1-t} \quad , \tag{4.4}$$

where $\|...\|$ denotes the infinity norm of a matrix, and thus round-off errors in the back-solution of a triangular system of linear equations may be regarded as equivalent to relatively small perturbations in the original system.

Similarly, the computed L and U obtained by Gaussian elimination with row interchanges from an upper-Hessenberg matrix H satisfy the perturbed equation

$$H + \delta H = LU \quad , \tag{4.5}$$

where Forsythe and Moler show that

$$\frac{\|\delta H\|}{\|H\|} \leq n^2 \rho\, \beta^{1-t} \quad , \tag{4.6}$$

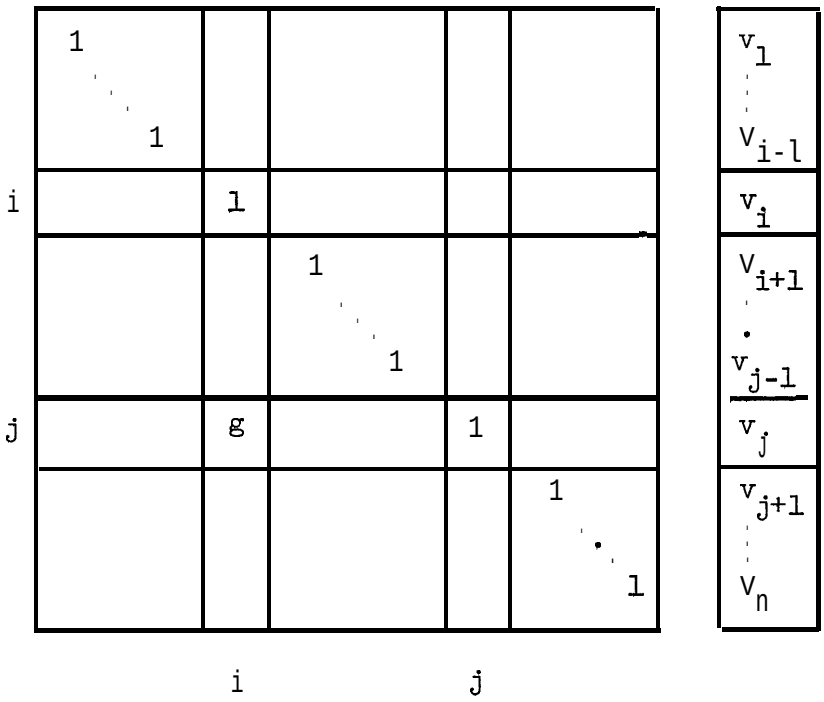
and Wilkinson [28] establishes that $\rho \leq n$. Thus, the computational

process indicated in (3.3) can be regarded as introducing only relatively small perturbations in each of the $h^{(\cdot)}$.

Similar results hold for the initial LU decomposition (2.2) with a different bound for ρ . The reader is referred again to Forsythe and Moler.

The most frequent computational step in the processes which we have described is the application of one Gaussian elimination step Γ to a column vector v :

(4.7) $w = \Gamma v =$



The computed vector w satisfies

(4.8) $w_k = v_k \quad \text{for } k \neq j$

$$w_j = fl(fl(gv_i) + v_j)$$

$$= gv_i(1 + \varepsilon_3)(1 + \varepsilon_1) + v_j(1 + \varepsilon_2)$$

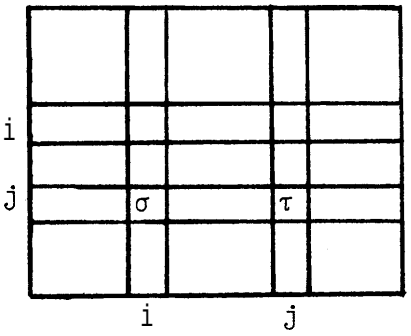
$$= gv_i + v_j + gv_i(\varepsilon_1 + \varepsilon_3 + \varepsilon_1\varepsilon_3) + v_j\varepsilon_2$$

Thus we may regard the computed vector w as the exact result of a perturbed transformation

$$w = (\Gamma + \delta\Gamma)v \quad , \tag{4.9}$$

where

$$\delta\Gamma = \tag{4.10}$$



,

and

$$\sigma = g(\varepsilon_1 + \varepsilon_3 + \varepsilon_1\varepsilon_3) \tag{4.11}$$

$$\tau = \varepsilon_2 \quad .$$

Therefore we have

$$\frac{\|\delta\Gamma\|}{\|\Gamma\|} \leq \frac{|g||\varepsilon_1 + \varepsilon_3 + \varepsilon_1\varepsilon_3| + |\varepsilon_2|}{1 + |g|} \quad , \tag{4.12}$$

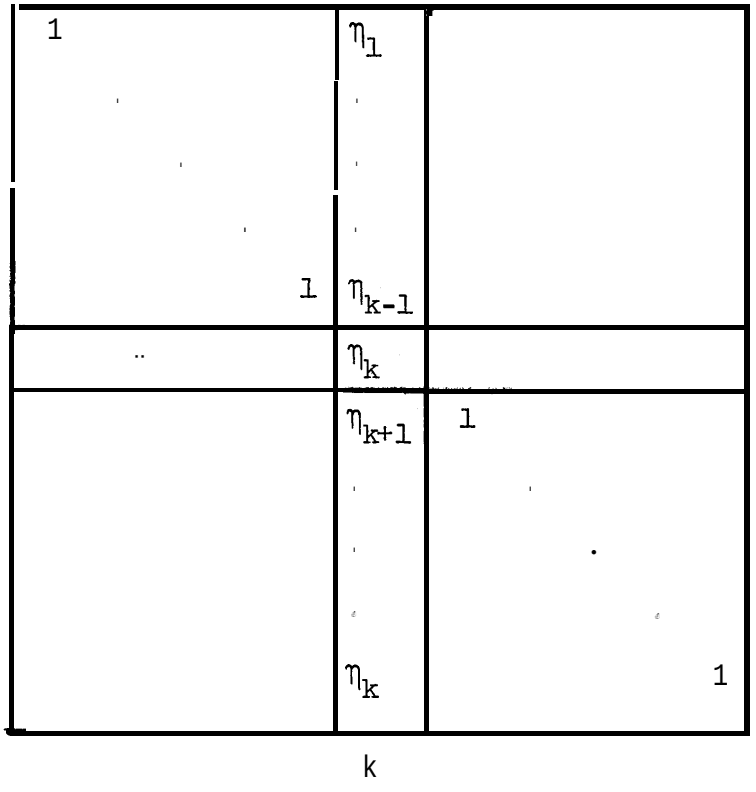
where the right-hand side is bounded, since $|g| \leq 1$, according to

$$\frac{\|\delta\Gamma\|}{\|\Gamma\|} \leq \beta^{1-t}[3 + \beta^{1-t}] < 3.01\beta^{1-t} \quad \text{(say)}. \tag{4.13}$$

Hence, the computations which we perform using transformations (4.7) also introduce relatively small perturbations into the quantities which we manipulate.

It is precisely with regard to such transformations that we feel our method of computation, based upon LU decompositions, is superior to methods based upon the inverses of the matrices $(B)_i$. Such methods use transformations of the form

(4.14)



.

These are applied to each column in $B^{(i-1)^{-1}}$ to produce $B^{(i)^{-1}}$; or alternatively, in product-form methods, they are applied to the vector $v^{(i)}$ to produce the solution to system (2.1). As such, they involve successive computations of the form (4.7). Each such computation may be regarded as satisfying (4.9). But, since the η_j may be unrestricted in magnitude, no bound such as (4.13) can be fixed.

5. Efficiency Considerations

As we have already pointed out, it requires

(5.1) $$n^3/3 + O(n^2)$$

multiplication-type operations to produce an initial LU decomposition (2.2). To produce the product-form inverse of an $n \times n$ matrix, on the other hand, requires

(5.2) $$n^3/2 + O(n^2)$$

operations.

The solution for any system (2.1) must be found according to the LU-decomposition method by computing

(5.3) $$Y = L^{(i)^{-1}} v^{(i)}$$

followed by solving

(5.4) $$U^{(i)} x = y \quad .$$

The application of $L^{(0)^{-1}}$ to $v^{(i)}$ in (5.3) will require

(5.5) $$\frac{n(n-1)}{2}$$

operations. The application of the remaining transformations in $L^{(i)^{-1}}$ will require at most

(5.6) $$i(n-1)$$

operations. Solving (5.4) costs

(5.7) $\frac{n(n+1)}{2}$

operations. Hence, the cost of (5.3) and (5.4) together is not greater than

(5.8) $n^2 + i(n-1)$

operations, and a reasonable expected figure would be $n^2 + \frac{1}{2}(n-1)$.

On the other hand, computing the solution to (2.1) using the usual product form of $B^{(i)^{-1}}$ requires the application of $n+i$ transformations of type (4.14) to $v^{(i)}$ at a cost of

(5.9) $n^2 + in$

operations.

If a vector $a^{(i)}$ replaces column r_1 in $B^{(i)}$, then the updating of $B^{(i)^{-1}}$ requires that the vector

(5.10) $z = B^{(i)^{-1}} a^{(i)}$

be computed. This will cost $n^2 + in$ operations, as shown in (5.9). Then a transformation of form (4.14) must be produced from z , and this will bring the total updating cost to

(5.11) $n^2 + (i+1)n$.

The corresponding cost for updating the LU decomposition will be not more than

(5.12) $\frac{n(n-1)}{2} + i(n-1)$

operations to find $L^{(i)^{-1}} a^{(i)}$, followed by at most

(5.13) $\frac{n(n+1)}{2}$

operations to reduce $H^{(i+1)}$ to $U^{(i+1)}$ and generate the transformations of type (3.4) which effect this reduction. This gives a total of at most

(5.14) $n^2 + i(n-1)$

operations, with an expected figure closer to $n^2+\frac{i}{2}(n-1)$.

Hence, in every case the figures for the LU decomposition: (5.14), (5.8), and (5.1) are smaller than the corresponding figures (5.11), (5.9), and (5.2) for the product-form inverse method.

6. Storage Considerations

All computational steps for the IX-decomposition method may be organized according to the columns of the matrices $B^{(i)}$. For large systems of data this permits a two-level memory to be used, with the high-speed memory reserved for those columns being actively processed.

The organization of Gaussian elimination by columns is well-known, and it is clear how the processes (5.3) may be similarly arranged. Finally, the upper-triangular systems (5.4) can be solved columnwise as indicated below in the 4×4 case:

$$(6.1) \qquad \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} .$$

Bring the y vector and the last column of U into high-speed memory. Set $x_4 = y_4/u_{44}$. Set $y_i' = y_i - u_{i4}x_4$ for $i = 3,2,1$. This leaves us with the following 3×3 system:

$$(6.2) \qquad \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y_1' \\ y_2' \\ y_3' \end{pmatrix}$$

We process it as suggested in the 4×4 case, using now the third column of U to produce x_3. Repeat as often as necessary.

In the event that the matrices $B^{(i)}$ are sparse as well as large, we wish to organize computations additionally in such a way that this sparseness is preserved as much as possible in the decompositions. For the initial decomposition (2.2), for example, we would wish to order the columns of $B^{(0)}$ in such a way that the production of $L^{(0)^{-1}}$ and $U^{(0)}$ introduce as few new nonzero elements as possible. And at subsequent stages, if there is a choice in the vector $a^{(i)}$ which is to be introduced as a new column into the matrix $B^{(i)}$ to produce $B^{(i+1)}$, it may be desirable to make this choice to some extent on sparseness considerations,.

It is not generally practical to demand a minimum growth of nonzero elements over the entire process of computing the initial decomposition. However, one can easily demand that, having processed the first k-l columns according to (1.2) and (1.3), the next column be chosen from those remaining in such a way as to minimize the number of nonzero elements generated in the next execution of steps (1.2) and (1.3). See, for example, Tewarson [27] Choice of the next column may also be made according to various schemes of "merit"; e.g., see Dantzig et al. [6].

The introduction of new nonzero elements during the process of updating the i-th decomposition to the i+1-st depends upon

(6.3) the nonzero elements in $L^{(i)^{-1}} a^{(i)}$ over those in $a^{(i)}$,

and

(6.4) the number r_1 of the column to be removed from $B^{(i)}$.

No freedom is possible in the reduction of $H^{(i+1)}$ to $U^{(i+1)}$ once $a^{(i)}$ has been chosen and the corresponding r_i has been determined.

The growth (6.3) can be determined according to the techniques outlined in Tewarson's paper, at a cost for each value of i , however, which is probably unacceptable. The more important consideration is (6.4). The larger the value of r_i , the fewer elimination steps must be carried out on $H^{(i+1)}$ and the less chance there is for nonzero elements to be generated. Again, however, the determination of the value of r_i corresponding to each possible choice of $a^{(i)}$ may prove for most algorithms to be unreasonably expensive.

7. Accuracy Considerations

During the execution of an exchange algorithm it sometimes becomes necessary to ensure the highest possible accuracy for a solution to one of the systems (2.1). High accuracy is generally required of the last solution in the sequence, and it may be required at other points in the sequence when components of the solution, or numbers computed from them, approach critical values. For example, in the simplex method inner products are taken with the vector of simplex multipliers, obtained by solving a system involving $B^{(i)}$, and each of the non-basic vectors. The computed values are then subtracted from appropriate components of the cost vector, and the results are compared to zero. Those which are of one sign have importance in determining how the matrix $B^{(i+1)}$ is to be obtained from $B^{(i)}$. The value zero, of course, is critical.

The easiest way of ensuring that the computed solution to a system

$$Bx = v \tag{7.1}$$

has high accuracy is by employing the technique of iterative refinement [9, Chapter 13]. According to this technique, if $x^{(0)}$ is any sufficiently good approximation to the solution of (7.1) (for example, a solution produced directly via the W-decomposition of B) then improvements may be made by computing

$$r^{(j)} = v - Bx^{(j)} , \tag{7.2}$$

solving

$$Bz^{(j)} = r^{(j)} , \tag{7.3}$$

and setting

$$x^{(j+1)} = x^{(j)} + z^{(j)} \tag{7.4}$$

for $j = 0,1,2,\ldots$ until $\|z^{(j)}\|$ is sufficiently small. The inner products necessary to form the residuals (7.2) must be computed in double-precision arithmetic. If this rule is observed, however, and if the condition of the system, measured as

$$\mathrm{cond}(B) = \|B\| \, \|B^{-1}\| , \tag{7.5}$$

is not close to β^{t-1}, the refinement process can be counted on to terminate in a few iterations. The final vector $x^{(j)}$ will then be as accurate a solution to (7.1) as the significance of the data in B and v warrant.

Step (7.3) is most economically carried out, of course, via the same I&decomposition which was used to produce $x^{(0)}$. If this is done, each repetition of steps (7.2) through (7.4) will cost only $O(n^2)$ operations. The alternative approach of producing a highly accurate solution to (7.1) by solving the system entirely in double-precision arithmetic is generally more expensive than iterative refinement by a factor of n.

B. THE QR DECOMPOSITION AND QUADRATIC PROGRAMMING

8. Householder Triangularization

Householder transformations have been widely discussed in the literature. In this section we are concerned with their use in reducing a matrix A to upper-triangular form, and in particular we wish to show how to update the decomposition of A when its columns are changed one by one. This will open the way to the implementation of efficient and stable algorithms for solving problems involving linear constraints.

Householder transformations are symmetric orthogonal matrices of the form $P_k = I - \beta_k u_k u_k^T$ where u_k is a vector and $\beta_k = 2/(u_k^T u_k)$. Their utility in this context is due to the fact that for any non-zero vector a it is possible to choose u_k in such a way that the transformed vector $P_k a$ is zero except for its first element. Householder [15] used this property to construct a sequence of transformations to reduce a matrix to upper-triangular form. In [29], Wilkinson describes the process and his error analysis shows it to be very stable.

Thus if $A = (a_1, \ldots, a_n)$ is an $m \times n$ matrix of rank r, then at the k-th stage of the triangularization $(k < r)$ we have

$$A^{(k)} = P_{k-1} P_{k-2} \cdots P_0 A = \begin{pmatrix} R_k & S_k \\ 0 & T_k \end{pmatrix}$$

where R_k is an upper-triangular matrix of order k. The next step is to compute $A^{(k+1)} = P_k A^{(k)}$ where P_k is chosen to reduce the first

column of T_k to zero except for the first component. This component becomes, the last diagonal element of R_{k+1} and since its modulus is equal to the Euclidean length of the first column of T_k it should in general be maximized by a suitable interchange of the columns of $\begin{pmatrix} S_k \\ T_k \end{pmatrix}$. After r steps, T_r will be effectively zero (the length of each of its columns will be smaller than some tolerance) and the process stops.

Hence we conclude that if rank(A) = r then for some permutation matrix Π the Householder decomposition (or "QR decomposition") of A is

$$Q A \Pi = P_{k-1} P_{k-2} \ldots P_0 A = \begin{pmatrix} \overbrace{R}^{r} & \overbrace{S}^{n-r} \\ 0 & 0 \end{pmatrix}$$

where $Q = P_{r1} P_{r2} \ldots P_0$ is an $m \times m$ orthogonal matrix and R is upper-triangular and non-singular.

We are now concerned with the manner in which Q , R , S may be updated if the columns of A are changed. We will suppose that a column a_p is deleted from A and that a column a_q is added. It will be clear what is to be done if only one or the other takes place.

Compact Method:

Since the Householder transformations P_k are defined by the vectors u_k the usual method is to store the u_k's in the area beneath R , with a few extra words of memory being used to store the β_k's and the diagonal

elements of R . The product Qz for some vector z is then easily computed in the form $P_{r\,1}\,P_{r\,2}\,\ldots\,P_0\,z$ where, for example, $P_0 z = (I - \beta_0 u_0 u_0^T)z = z - \beta_0(u_0^T z)u_0$. The updating is best accomplished as follows. The first p-l columns of the new R are the same as before; the other columns p through n are simply overwritten by columns $a_{p+1}, \ldots, a_n, a_q$ and transformed by the product $P_{p-1}\,P_{p-2}\cdots P_0$ to obtain a new $\begin{pmatrix} S_{P-1} \\ T_{P-1} \end{pmatrix}$; then T_{p-1} is triangularized as usual. This method allows Q to be kept in product form always, and there is no accumulation of errors. Of course, if p = 1 the complete decomposition must be re-done and since with $m \geq n$ the work is roughly proportional to $(m-n/3)n^2$ this can mean a lot of work. But if $p \doteq n/2$ on the average, then only about 1/8 of the original work must be repeated each updating.

Explicit Method:

The method just given is probably best when $m \gg n$. Otherwise we propose that Q should be stored explicitly and that the updating be performed as follows:

(1) The initial Q can be computed by transforming the identity matrix thus:

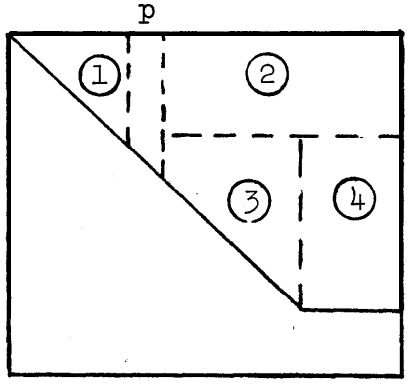
$$P_{r-1}\,P_{r-2}\cdots P_0(A^T \mid I_m) \rightarrow \left(\begin{array}{cc|c} R & S & \\ 0 & 0 & Q \end{array}\right) .$$

(2) If a_q is added to A then compute $s_q = Qa_q$ and add it to the end of $\begin{pmatrix} S \\ O \end{pmatrix}$.

(3) Delete a_P where applicable $(P < r)$. This normally means just updating the permutation vector used to describe π.

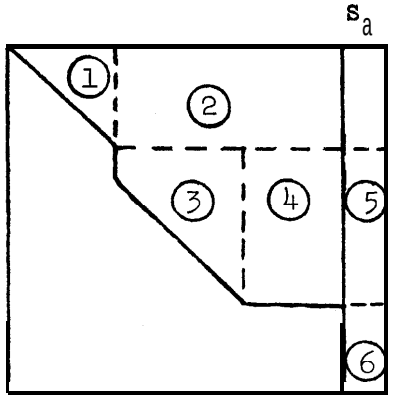
(4) The initial situation

$QA\pi$ =



has thus been changed to

$Q\bar{A}\bar{\pi}$ =



where the areas (1), (2), (3), (4) are the same as before.

This is analogous to the Hessenberg form encountered in updating LU decompositions. We now employ a sequence of (r-p) plane rotations, as used by Givens and analyzed by Wilkinson [30], to reduce the subdiagonal of area θ^3 to zero. This changes areas (3), (4) and (5), and the corresponding rows of Q must also be transformed. Since the plane rotations are elementary orthogonal transformations, the latter step produces a new matrix Q* which is also orthogonal, and the work necessary is approximately proportional to $2mn + n^2$.

(5) Finally, a single Householder transformation P_r is applied to produce $\bar{Q} = P_r Q^*$, where this transformation is the one which reduces area θ^6 to zeros except for the first element. The work involved is proportional to $2(m-n)m$.

Thus the transformation $\bar{Q}$ reduces $\bar{A}\bar{\Pi}$ to a new upper-triangular form, and the original transformations $P_0, \ldots, P_{r-1}$, the plane rotations, and the final Householder transformation may all be discarded since the required information is all stored in $\bar{Q}$. The total work involved is roughly proportional to $(2mn + n^2) + 2(m-n)m = 2m^2 + n^2$ and the stability of the orthogonal transformations is such that accumulation of rounding errors during repeated applications of the updating process should be very slight.

9. Projections

In optimization problems involving linear constraints it is often necessary to compute the projections of some vector either into or orthogonal to the space defined by a subset of the constraints (usually the current "basis"). In this section we show how Householder transformations may be used to compute such projections. As we have shown, it is possible to update the Householder decomposition of a matrix when the number of columns in the matrix is changed, and thus we will have an efficient and stable means of orthogonalizing vectors with respect to basis sets whose component vectors are changing one by one.

Let the basis set of vectors $a_1, a_2, \ldots, a_n$ form the columns of an $m \times n$ matrix A, and let S_r be the sub-space spanned by $\{a_i\}$. We shall assume that the first r vectors are linearly independent and that rank(A) = r . In general, $m \geq n \geq r$, although the following is true even if $m < n$.

Given an arbitrary vector z we wish to compute the projections

$$u = Pz \ , \ v = (I-P)z$$

for some projection matrix P, such that

(a) $z = u + v$

(b) $u^T v = 0$

(c) $u \in S_r$ (i.e., $\exists x$ such that $Ax = u$)

(d) v is orthogonal to S_r (i.e., $A^T v = 0$) .

One method is to write P as AA^+ where A^+ is the $n \times m$ generalized inverse of A , and in [7] Fletcher shows how A^+ may be updated upon changes of basis. In contrast, the method based on Householder transformations does not deal with A^+ explicitly but instead keeps AA^+ in factorized form and simply updates the orthogonal matrix required to produce this form. Apart from being more stable and just as efficient, the method has the added advantage that there are always two orthonormal sets of vectors available, one spanning S_r and the other spanning its complement.

As already shown, we can construct an $m \times n$ orthogonal matrix Q such that

$$QA = \begin{pmatrix} \overbrace{R}^{r} & \overbrace{S}^{n-r} \\ 0 & 0 \end{pmatrix}$$

where R is an $r \times r$ upper-triangular matrix. Let

$$(9.1) \qquad w = Qz = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \begin{matrix} \} r \\ \} m-r \end{matrix}$$

and define

$$(9.2) \qquad u = Q^T \begin{pmatrix} w_1 \\ 0 \end{pmatrix} \quad , \quad v = Q^T \begin{pmatrix} 0 \\ w_2 \end{pmatrix} \quad .$$

Then it is easily verified that u,v are the required projections of z , which is to say they satisfy the above four properties. Also, the x in

(c) is readily shown to be

$$x = \begin{pmatrix} R^{-1}w_1 \\ 0 \end{pmatrix}$$

In effect, we are representing the projection matrices in the form

$$P = Q^T \begin{pmatrix} I_r \\ 0 \end{pmatrix} (I_r \; 0) Q \tag{9.3}$$

and

$$I - P = Q^T \begin{pmatrix} 0 \\ I_{m-r} \end{pmatrix} (0 \; I_{m-r}) Q \tag{9.4}$$

and we are computing $u = Pz$, $v = (I - P)z$ by means of (9.1), (9.2). The first r columns of Q span S_r and the remaining m-r span its complement. Since Q and R may be updated accurately and efficiently if they are canputed using Householder transformations, we have as claimed the means of orthogonalizing vectors with respect to varying bases.

As an example of the use of the projection (9.4), consider the problem of finding the stationary values of x^TAx subject to $x^Tx = 1$ and $C^Tx = o$, where A is a real symmetric matrix of order n and C is an $n \times p$ matrix of rank r , with $r \leq p < n$. It is shown in [12] that if the usual Householder decomposition of C is

$$QC = \begin{pmatrix} \overbrace{R}^{r} & \overbrace{S}^{n-r} \\ 0 & 0 \end{pmatrix}$$

then the problem is equivalent to that of finding the eigenvalues and eigenvectors of the matrix $\hat{P}A$, where

$$\hat{P} = I - P = Q^T \begin{pmatrix} 0 & 0 \\ 0 & I_{n-r} \end{pmatrix} Q$$

is the projection matrix in (9.4). It can then be shown that if

$$Q A Q^T = \begin{pmatrix} G_{11} & G_{12} \\ G_{12}^T & G_{22} \end{pmatrix}$$

where G_{11} is $r \times r$, then the eigenvalues of $\hat{P}A$ are the same as those of G_{22} and so the eigensystem has effectively been deflated by the number of independent linear constraints. Similar transformations can be applied if the quadratic constraint is $x^T Bx = 1$ for some real positive definite matrix B .

10. Orthogonalization with Respect to Positive Definite Forms

Fletcher also shows in [7] how to update projection matrices when it is required to orthogonalize with respect to a given positive definite matrix D . We now show how to compute such projections using Householder transformations, and hence the comments made in the last section concerning changes of basis may also be applied here.

Given an arbitrary vector z it is required to find $u = Pz$, $v = (I - P)z$ for some P , such that

(a) $z = u+v$

(b) $u^T Dv = 0$

(c) $\exists x$ such that $Ax = u$

(d) $(DA)^T v = 0$.

For simplicity we will assume that rank(A) = n . Then, rather than computing P explicitly as Fletcher does according to

$$P = A(A^T DA)^{-1} A^T D ,$$

we obtain the Cholesky decomposition of D thus:

$$D = LL^T$$

where L is lower-triangular and non-singular if D is positive definite. We then compute $B = L^T A$ and obtain the decomposition

$$QB = \begin{pmatrix} R \\ 0 \end{pmatrix} .$$

Defining

$$w = QL^T z = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \begin{matrix} \} n \\ \} m-n \end{matrix}$$

and

$$u = L^{-T}Q^T \begin{pmatrix} w_1 \\ 0 \end{pmatrix}, \quad v = L^{-T}Q^T \begin{matrix} 0 \\ w_2 \end{matrix}$$

it is easily verified that u,v are the required projections, and again the x in (c) is given by $x = R^{-1}w_1$. Since changing a column a_k of A is equivalent to changing the column $L^T a_k$ of B, the matrices Q and R may be updated almost as simply as before.

11. Linear Least Squares and Quadratic Programming

We first consider minimization of quadratic forms subject to linear equality constraints. The solution is given by a single system of equations and the algorithm we describe for solving this system will serve as a basic tool for solving problems with inequality constraints. It will also provide an example of how solutions to even strongly ill-conditioned problems may be obtained accurately if orthogonalization techniques are used.

Let A,G be given matrices of orders $m \times n$, $p \times n$ respectively and let b,h be given vectors of consistent dimension. The least squares problem to be considered here is

$$\text{Problem LS:} \qquad \min \|b - Ax\|_2$$

$$\text{subject to} \quad Gx = h \ .$$

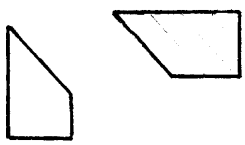
Similarly, let D be a given positive semi-definite matrix and c a given n-dimensional vector. The quadratic programming problem corresponding to the above is

$$\text{Problem QP:} \qquad \min \ \frac{1}{2} x^T D x + c^T x$$

$$\text{subject to} \quad Gx = h \ .$$

Now we can obtain very accurately the following Cholesky decomposition of D :

$$D = A^T A = \quad$$



where we deliberately use A again to represent the triangular factor. If D is semi-definite, a symmetric permutation of rows and columns will generally be required. If D is actually positive definite then A will be a non-singular triangular matrix.

With the above notation, it can be shown that the solutions of both problems satisfy the system

$$\text{(11.1)} \qquad \begin{pmatrix} & & G \\ & I & A \\ G^T & A^T & \end{pmatrix} \begin{pmatrix} z \\ r \\ x \end{pmatrix} = \begin{pmatrix} h \\ b \\ c \end{pmatrix}$$

where

$$c = 0 \ , \quad r = b - Ax \qquad \text{for Problem LS,}$$

$$b = 0 \ , \quad r = -Ax \qquad \text{for Problem QP,}$$

and z is the vector of Lagrange multipliers. In [2], [3] methods for solving such systems have been studied in depth. The method we give here is similar but more suited to our purpose. This method has been worked on independently by Leringe and Wedin [17]. The solution of (11.1) is not unique if the quantity $\operatorname{rank}\begin{pmatrix} G \\ A \end{pmatrix}$ is less than n, but in such cases we shall be content with obtaining one solution rather than many. The important steps follow.

(1) Let Q_1 be the orthogonal matrix which reduces G^T to triangular form, and let Q_1 also be applied to A^T, thus:

$$\text{(11.2)} \qquad Q_1(G^T \mid A^T) = \left(\begin{array}{c|c} R_1 & S \\ 0 & T \end{array}\right)$$

As explained earlier, Q_1 can be constructed as a sequence of Householder transformations, and the columns of G^T should be permuted during the triangularization. This allows any redundant constraints in $Gx = h$ to be detected and discarded.

(2) Let Q_2 be the orthogonal matrix which reduces T^T to triangular form:

$$Q_2 T^T = \begin{pmatrix} R_2 \\ 0 \end{pmatrix} \tag{11.3}$$

Here we assume for simplicity that T is of full rank, which is equivalent to assuming that (11.1) has a unique solution, and again we suppress permutations from the notation.

(3) The combined effect of these decompositions is now best regarded as the application of an orthogonal similarity transformation to system (11.1), since the latter is clearly equivalent to

$$\begin{pmatrix} I & & \\ & Q_2 & \\ & & Q_1 \end{pmatrix} \begin{pmatrix} & & G \\ & I & A \\ G^T & A^T & \end{pmatrix} \begin{pmatrix} I & & \\ & Q_2^T & \\ & & Q_1^I \end{pmatrix} \begin{pmatrix} z \\ Q_2 r \\ Q_1 x \end{pmatrix} = \begin{pmatrix} h \\ Q_2 b \\ Q_1 c \end{pmatrix} .$$

The resulting system consists of various triangular sub-systems involving R_1 , R_2 , S , and can easily be solved.

(4) If desired, the solution thus obtained can be improved upon via the method of iterative refinement [9], since this just involves the solution of system (11.1) with different right-hand sides, and the necessary decompositions are already available.

The algorithm just described has been tested on extremely ill-conditioned systems involving inverse Hilbert matrices of high order and with iterative refinement has given solutions which are accurate to full machine precision.

12. Positive Definite Programming

With the algorithm of the previous section available, we are now prepared to attack the following more general programming problems:

Problem LS: $$\min \|b - Ax\|_2$$

$$\text{subject to } G_1 x = h_1 ,$$

$$G_2 x \geq h_2 .$$

Problem QP: $$\min \tfrac{1}{2} x^T D x + c^T x$$

subject to the same contraints.

Let G_1, G_2 be of orders $p_1 \times n$, $p_2 \times n$ respectively, and again suppose that D has the Cholesky decomposition $A^T A$. In this section we consider problems for which $\operatorname{rank} \begin{pmatrix} A \\ G_1 \end{pmatrix} = n$ (which is most likely to be true with least squares problems, though less likely in QP). In such cases the quadratic form is essentially invertible (but we emphasize that its inverse is not computed) and so x can be eliminated from the problem. With the notation of the preceding section the steps are as follows:

(1) Solve (11.1) with G_1, h_1 to get the solution $x = x_0$, then compute the vector $q = G_2 x_0 - h_2$.

(2) If $q \geq 0$ then x_0 is the solution.

Otherwise, transform the inequality matrix using Q_1 from step (1), so that

$$Q_1 (G_1^T \mid A^T \mid G_2^T) = \left(\begin{array}{c|c|c} R_1 & S & U \\ 0 & T & V \end{array} \right) \begin{matrix} \} p_1 \\ \} n-p_1 \end{matrix} .$$

(3) If $Q_2 T^T = \begin{pmatrix} R_2 \\ 0 \end{pmatrix}$ as before and if $M = R_2^{-T} V^T$ it can be shown that the active constraints are determined by the following linear complementarity problem (LCP):

$$w = q + M^T M z$$

(12.1)

$$w, z > 0 \quad , \quad z^T w = 0 \quad .$$

w, z are respectively the slack variables and Lagrange multipliers associated with the inequality constraints.

(4) The active constraints (for which $w_1 = 0$ in the solution of the LCP) are now added to the equalities $G_1 x = h_1$ and the final solution is obtained from (11.1).

We wish to focus attention on the method by which the LCP (12.1) is solved. Cottle and Dantzig's principal pivoting method [5] could be applied in a straightforward manner if $M^T M$ were computed explicitly, but for numerical reasons and because $M^T M$ $(p_2 \times p_2)$ could be very large, we avoid this. Rather we take advantage of the fact that no more than $n - p_1$ inequalities can be active at any one time and work with a basis M_1 made up of k columns of M , where $1 \leq k \leq n - p_1$. The QR -decomposition

$$QM_1 = \begin{pmatrix} R \\ 0 \end{pmatrix}$$

is maintained for each basis as columns of M are added to or deleted from M_1 and as we know, Q and R can be updated very quickly each

change. Then just as in the LU method for linear programming, the new basic solution is obtained not by updating a simplex tableau but simply by solving the appropriate system of equations using the available decomposition.

As an example we show how canplementary basic solutions may be obtained. Let the basis M_1 contain k columns of M and let M_2 be the remaining (non-basic) columns. The system to be solved is

$$\begin{pmatrix} 0 \\ w_B \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} + \begin{pmatrix} M_1^T M_1 \\ M_2^T M_1 \end{pmatrix} z_B$$

with obvious notation. If we define $y = -M_1 z_B$ this is best written as

$$\begin{pmatrix} I & M_1 \\ M_1^T & \end{pmatrix} \begin{pmatrix} y \\ z_B \end{pmatrix} = \begin{pmatrix} 0 \\ q_1 \end{pmatrix} \tag{12.2}$$

$$w_B = q_2 - M_2^T y \tag{12.3}$$

and the solution of (12.2) is readily obtained from

$$u = R^{-T} q_1 \quad , \quad z_B = -R^{-1} u \quad , \quad y = Q^T \begin{pmatrix} u \\ 0 \end{pmatrix} \begin{matrix} \} k \\ \} n-p_1-k \end{matrix}$$

The blocking variable when a non-basic variable is increased can be found from the solution of the same set of equations with the appropriate right-hand side. It is worth noting that the equations can be simplified

if the basis is square (i.e., if there are as many constraints active as there are free variables). Since it seems very common for the basis to fill up during the iterations (even if the final solution does not have a full set of constraints) it is worth treating a full basis specially.

Almost-complementary solutions can be obtained in similar fashion (with somewhat more work required as the system is then not quite so symmetric). Thus an algorithm such as Cottle and Dantzig's can be implemented using these techniques, and convergence is thereby guaranteed.

Of special interest, however, is the following unpublished and apparently novel idea due to Yonathan Bard, with whose permission we report the results he has obtained. Almost-complementary bases are never allowed to occur; instead, if a basic variable is negative, then it is replaced by its complement <u>regardless of the effect on the other basic variables</u>. Bard'has tried this method (carried to convergence) on hundreds of problems of the form $w = q + Mz$ and cycling has never occurred when the most negative element of q is chosen. In a series of tests on 100 random matrices of orders between 2 and 20, principal pivoting required a total of 537 pivots whereas the Cottle-Dantzig algorithm required 689 .

The present authors' experience with fewer but larger problems confirms the above observation that convergence does actually occur and usually after a small number of iterations. Since the idea eliminates all work other than computation of complementary solutions it is particularly suited to the techniques of this section. At worst it should

be used as a starting procedure to find a close-to-optimal basis quickly, and at best if the conjecture can be proven that it will always converge, then a lot of computer time could be saved in the future.

[It has since been learned that Bard applied the principal-pivoting rule to LCP's of the somewhat special form in which

$$M = P^T P, \quad q = P^T p$$

for some P, p. Problems of this form have been studied by Zoutendijk in [31,32] where several pivot selection rules are discussed. Finiteness is proven for one rule, but simpler methods (such as Bard's) are recommended in practice for efficiency.

The question of finiteness for the more general LCP remains open, and it is likely that somewhat more sophisticated rules (e.g., Cottle and Dantzig) will be required.

13. Semi-#Definite Programming

We now consider the more general problem in which the rank of the quadratic form combined with the equality constraints may be less than n . The method we propose is conceptually as simple as it is stable. It is analogous to the revised simplex method for linear programming in that the essential steps to be implemented are as follows:

(1) Find the current basic solution from a certain system of equations for which a decomposition is available.

(2) Determine according to a certain set of rules what modifications should be made to the system to obtain a new basis.

(3) If necessary, update the decomposition and return to step (1).

Thus, suppose that the current basis contains $G_B x = h_B$ as active constraints. As in (11.1) the corresponding basic solution is then given by

$$\begin{pmatrix} & & G_B \\ & I & A \\ G_B^T & A^T & \end{pmatrix} \begin{pmatrix} z_B \\ r \\ x \end{pmatrix} = \begin{pmatrix} h_B \\ b \\ c \end{pmatrix} \tag{13.1}$$

and

$$w_B = h_B - G_B x \ . \tag{13.2}$$

(Here, $\bar{G}_B x \geq \bar{h}_B$ are the currently inactive constraints, w_B the corresponding slack variables, and z_B the Lagrange multipliers or dual variables associated with the active constraints.) The elements of z_B

corresponding to any equality constraints may be either positive or negative and need never be looked at. Ignoring these, the basic solution above is optimal if and only if

$$z_B \geq 0 \quad \text{and} \quad w_B \geq 0 \ .$$

A "QP algorithm" is now to be regarded as the "certain set of rules" mentioned in step (2) whereby z_B, w_B and possibly other information are used to determine which constraints should be added to or dropped from G_B. The efficiency of the method will depend on the speed with which this decision can be made and on the efficiency with which the decomposition of (13.1) can be updated.

Once again the most promising pivot-selection rule is that of Bard, as discussed in the previous section. The general idea in this context is as follows:

(a) Find $w_\alpha = \min w_i$, $z_\beta = \min z_i$ from those eligible elements of w_B, z_B.

(b) If $w_\alpha < 0$, constraint α could be added.

(c) If $z_\beta < 0$, constraint β could be dropped.

(d) If there are already n constraints active and $w_\alpha < 0$, constraint α could replace constraint β.

We do not consider here the question of convergence, but as already stated, this type of rule has been found to work.

The problem of updating the requisite decompositions is more relevant at present. We discuss this and other points briefly.

(1) The matrices Q_1, R_1 of Equation (11.2) can be updated efficiently using the methods of Section 8.

(2) Q_2, R_2 obtained from the matrix T in Equation (11.3) unfortunately cannot be updated, but the work needed to recompute them might often be very small, for the following reasons:

(a) In Problem LS, a preliminary triangularization of A ($m \times n$) can be applied to obtain an equivalent problem for which $m < n$. The Cholesky factor of D in Problem QP already has this property.

(b) If there are many constraints active (up to n) then T has very few rows.

(c) If the rank of the system is low (relative to n) then T has very few columns.

(3) Hence the method is very efficient if close to n constraints are active each iteration, as should often be the case. It also has the property, along with Beale's algorithm [1], of being most efficient for problems of low rank.

(4) The procedure can be initiated with any specified set of constraints in the first basis, and an initial estimate of x is not required.

(5) Any number of constraints can be handled, in the same way that the revised simplex method can deal with any number of variables.

(6) If $D = 0$ the problem is a linear program and only bases containing n constraints need be considered. The method reduces to something like a self-dual simplex algorithm.

Finally we note that with semi-definite problems it is possible for some basic system (13.1) to be singular. If there are any solutions at all then there are many (this will always be the case with low rank least squares problems) but this does not matter, since z_B is still uniquely determined. However, a low rank quadratic program might be unbounded, and this is manifested by a singular system (13.1) proving to be inconsistent. In general, this just means that there are not yet enough constraints in the basis, so that trouble can usually be avoided by initializing the procedure with a full set of constraints.

C. THE SVD AND NONLINEAR LEAST SQUARES

14. The Singular Value Decomposition

Let A be a real, $m \times n$ matrix (for notational convenience we assume that $m \geq n$) . It is well known (cf. [ ]) that

$$A = U \Sigma V^T \tag{14.1}$$

where U,V are orthogonal matrices and

$$\Sigma = \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_n \\ \hline & 0 & \end{pmatrix} \Big\} (m-n) \times n$$

U consists of the orthonormalized eigenvectors of AA^T, and V consists of the orthonormalized eigenvectors of A^TA . The diagonal elementsof Σ are the non-negative square roots of the eigenvalues of A^TA ; they are called singular values or principal values of A . We assume

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0 \quad .$$

Thus if rank(A) = r , $\sigma_{r+1} = \sigma_{r+2} = \ldots = \sigma_n = 0$. The decomposition (14.1) is called the singular value decomposition (SVD).

An $n \times m$ matrix X is said to be the pseudo-inverse of an $m \times n$ matrix A if X satisfies the following four properties:

(i) $AXA = A$, (ii) $XAX = X$, (iii) $(XA)^T = XA$, (iv) $(AX)^T = AX$.

We denote the pseudo-inverse by A^+ . It can be shown that A^+ can always be determined and is unique (cf. [21]). It is easy to verify that $A^+ = V \Lambda U^T$ where A is the $n \times m$ matrix $A = \text{diag}[\sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_r^{-1}, 0, 0, \ldots, 0]$. There are many applications of the SVD in least squares problems (cf. [11]).

The SVD of an arbitrary matrix is calculated in the following way. First, a sequence of Householder transformations $\{P_k\}_{k=1}^{n}$, $\{Q_k\}_{k=1}^{n-1}$ is constructed [10] that

$$P_n P_{n-1} \cdots P_1 A Q_1 Q_2 \cdots Q_{n-1} \equiv P^T A Q = J$$

and J is an $m \times n$ bi-diagonal matrix of the form

$$J = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & 0 \\ & & \ddots & \ddots & \\ & 0 & & & \beta_{n-1} \\ & & & & \alpha_n \\ \hline & & 0 & & \end{pmatrix}_{3\ (m-n) \times n} .$$

The singular values of J are the same as those of A .

Next the SVD of J is computed by an algorithm given in [11]. The algorithm is based on the highly effective QR algorithm of Francis [10] for computing eigenvalues. If the SVD of $J = X \Sigma Y^T$ then $A = P X \Sigma Y^T Q^T$ so that $U = PX$, $V = QY$.

15. Nonlinear Least Squares

Consider the nonlinear transformation $F(x) = y$ where $x \epsilon E_n$ and $y \epsilon E_m$ with $n \leq m$. We wish to consider the following problem:

$$\min \|b - F(x)\|_2$$

subject to

$$G x = h , \tag{15.1}$$

where G is a $p \times n$ matrix of rank p and $h \epsilon E_p$. A very effective algorithm for solving such problems is a variant of the Levenberg-Marquardt algorithm [18,19]; in this section we consider some of the details of the numerical calculation. Further extensions of the algorithm are given by Shanno [25] and Meyer [20].

Let us assume that we have an approximation $x^{(0)}$ which satisfies the relation $Gx^{(0)} = h$. Then at each stage of the iteration we determine $\delta^{(k)}$ so that

$$x^{(k+1)} = x^{(k)} + \delta^{(k)} \tag{15.2}$$

and

$$G\delta^{(k)} = 0 . \tag{15.3}$$

Again as in Section 11, we write $Q_1 G^T = \begin{matrix} R \\ 0 \end{matrix}$ where Q_1 is the product of p Householder transformations and R is an upper triangular matrix. Let

$$Q_1 \delta^{(k)} = \begin{pmatrix} \xi^{(k)} \\ \eta^{(k)} \end{pmatrix} \begin{matrix} \} \ p \\ \} \ n-p \end{matrix} \tag{15.4}$$

Then from (15.3), we see that $\xi^{(k)} = 0$.

For notational convenience, let us drop the superscript k ; we write $x^{(k)}$ as x_0 and $x^{(k+1)}$ as x_1 .

In the Levenberg-Marquardt algorithm one determines the vector δ so that

$$\|r - J\delta\|_2^2 + \lambda \|\delta\|^2 = \min. \tag{15.5}$$

where

$$r = b - F(x_0) ,$$

J is the Jacobian evaluated at x_0 , and λ is an arbitrary non-negative parameter. From (15.4), we see that (15.5) is equivalent to determining η so that

$$\begin{cases} \|r - JQ_1^T \begin{pmatrix} \xi \\ \eta \end{pmatrix}\|_2^2 + \lambda(\|\xi\|_2^2 + \|\eta\|_2^2) = \min. \\ \text{subject to} \quad \xi = 0 \ . \end{cases} \tag{15.6}$$

Now let us write $JQ_1^T = [M,N]$ where N consists of the last n-p columns of JQ_1^T . Then (15.6) is equivalent to finding η so that

$$\Phi(\eta) \equiv \|r - N\eta\|_2^2 + \lambda \|\eta\|_2^2 = \min.$$

Consider the SVD of N ; namely

$$N = U \Sigma V^T \ .$$

Then

$$(15.7) \qquad \Phi(\eta) = \|U^T r - \Sigma V^T \eta\|_2^2 + \lambda \|V^T \eta\|_2^2$$

$$= \|s - \Sigma \zeta\|_2^2 + \lambda \|\zeta\|_2^2$$

where

$$s = U^T r \quad , \quad \zeta = V^T \eta \quad .$$

Writing out (15.7) explicitly, we have

$$\Phi(\zeta) \quad \sum_{j=1}^{\rho} (s_j - \sigma_j \zeta_j)^2 \quad \lambda \sum_{j=1}^{n-p} (\zeta_j)^2$$

where ρ is the rank of N . (Note ρ may change from iteration to iteration.) Then

$$\Phi(\hat{\zeta}) = \min$$

when

$$\hat{\zeta}_j = \frac{s_j \sigma_j}{\lambda \quad \sigma_j^2} \qquad \text{for } j = 1,2,\ldots,\rho,$$

$$= 0 \qquad \text{for } j > \rho$$

and hence

$$\eta = \sum_{j=1}^{\rho} \frac{s_j \sigma_j}{\lambda + \sigma_j^2} v_j$$

where v_j is the j-th column of V . Thus

$$\delta = Q_1^T \begin{pmatrix} 0 \\ \eta \end{pmatrix} \quad .$$

Note it is an easy matter to compute η (and hence δ) for various values of λ . The algorithm for computing the SVD can easily be organized so that s is computed directly ([11]).

There are several possible strategies for determining λ . One possibility is to choose $\hat{\lambda}$ so that

$$\|b - F(x_1(\hat{\lambda}))\|_2 \leq \|b - F(x_1(\lambda))\|_2 \quad .$$

This requires, of course, the evaluation of $F(x)$ at a great many points.

Another possibility is to choose δ such that

$$(15.8) \qquad \begin{cases} \|r - J\delta\|_2 = \min. \\ \\ \text{subject to} \quad \|\delta\|_2 \leq \alpha \quad . \end{cases}$$

This is equivalent to determining λ such that

$$\|\eta\|_2^2 = \sum_{j=1}^{\rho} \left(\frac{s_j \sigma_j}{\lambda + \sigma_j^2} \right)^2 \leq \alpha^2 \quad .$$

When $\lambda = 0$, we have the solution to the unconstrained problem and

$$\eta_0 = \sum_{j=1}^{\rho} \frac{s_j}{\sigma_j} v_j \quad .$$

Let $\|\eta_0\|_2 = \beta$. If $\beta \leq \alpha$, then we have the solution to (15.8). Otherwise, we must determine λ so that

$$(15.9) \qquad \sum_{j=1}^{\rho} \left(\frac{s_j \sigma_j}{\lambda + \sigma_j^2} \right)^2 = \alpha^2 \quad .$$

Let

$$u = \alpha^{-1} \begin{pmatrix} \sigma_1 s_1 \\ \sigma_2 s_2 \\ \vdots \\ \sigma_\rho s_\rho \end{pmatrix} \quad , \quad \Omega = \mathrm{diag}(\sigma_1^2, \sigma_2^2, \ldots \quad \text{yOF}) \quad ;$$

we assume $s_j \neq 0$ for $j = 1,2,\ldots,\rho$. By repeated use of the relationship

$$\det \begin{pmatrix} X & Y \\ Z & W \end{pmatrix} = \det(X) \det(W - ZX^{-1}Y) \quad \text{if} \quad \det(X) \neq 0$$

we can show that (15.9) is equivalent to

$$(15.10) \qquad \det((\Omega + \lambda I)^2 - uu^T) = 0$$

which has 2ρ roots; it can be shown that we need the largest real root, which we denote by λ^* ([8]). Let

$$\Gamma(\lambda) = \sum_{j=1}^{\rho} \left(\frac{s_j \sigma_j}{\lambda + \sigma_j^2} \right)^2 - \alpha^2$$

and assume that $\sigma_1^2 \geq \sigma_2^2 \geq \ldots \geq \sigma_\rho^2 > 0$. Note $\Gamma'(0) = \beta^2 - \alpha^2 > 0$, and $\Gamma(\lambda) \to -\alpha^2$ as $\lambda \to \infty$, so that $0 \leq \lambda^* < \infty$ and it is the only root in that interval. We seek a more precise upper bound for λ^*.

From (15.10) we see, using a Rayleigh quotient argument, that

$$A^* \leq \max_{\|y\|_2=1} [-y^T \Omega y + \sqrt{(y^T \Omega y)^2 - y^T(\Omega^2 - uu^T)y}\ ] .$$

A short manipulation then shows that

$$0 < \lambda^* \leq \sqrt{\sigma_1^4 - \sigma_\rho^4 + u^T u} - \sigma_\rho^2 \quad . \tag{15.11}$$

Thus, we wish to find a root of (15.10) which lies in the interval given by (15.11). Note that the determinantal equation (15.10) involves a diagonal matrix plus a matrix of rank one. In the next section we shall describe an algorithm for solving such problems.

16. Modified Eigensystems

As was pointed out in Section 15, it is sometimes desirable to determine some eigenvalues of a diagonal matrix which is modified by a matrix of rank one. Also, Powell [23] has recently proposed a minimization algorithm which requires the eigensystem of a matrix after a rank one modification. In this section, we give an algorithm for determining in $O(n^2)$ numerical operations some or all of the eigenvalues and eigenvectors of $D+\sigma uu^T$ where $D = \mathrm{diag}(d_i)$ is a diagonal matrix of order n and $u \epsilon E_n$.

Let $C = D+\sigma uu^T$; we denote the eigenvaluesof C by $\lambda_1,\lambda_2,\ldots,\lambda_n$ and we assume $\lambda_i > \lambda_{i+1}$ and $d_i \geq d_{i+1}$. It can be shown (cf. [30]) that

(1) If $\sigma \geq 0$, $d_1 + \sigma u^T u \geq \lambda_1 \geq d_1$, $d_i \geq \lambda_i \geq d_i$ $(i = 2,\ldots,n)$,

(2) If $\sigma < 0$, $d_i \geq \lambda_i \geq d_{i-1}$ $(i = 1,2,\ldots,n-1)$, $d_n \geq \lambda_n \geq d_n + \sigma u^T u$.

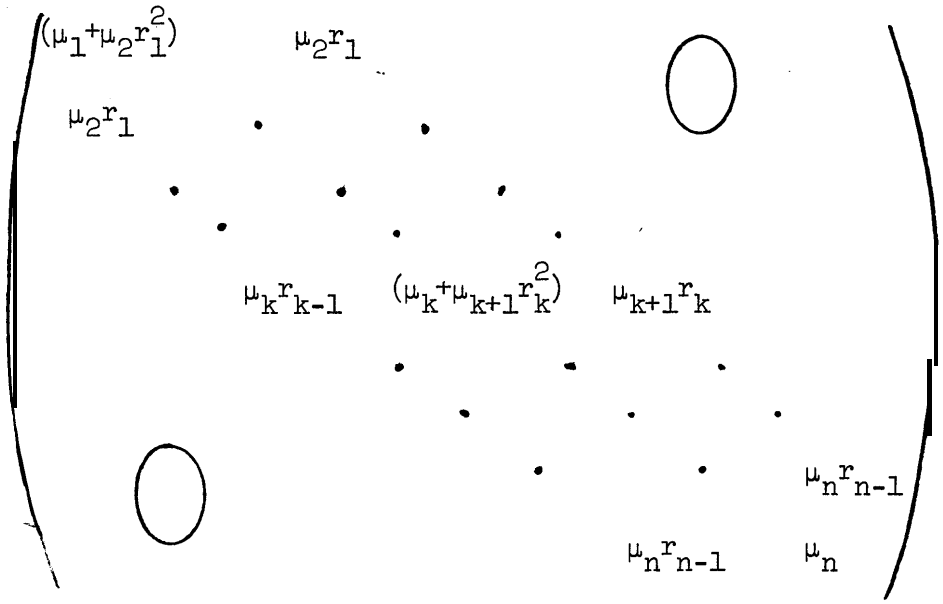
Thus we have precise bounds on each of the eigenvalues of the modified matrix.

Let K be a bi-diagonal matrix of the form

$$K = \begin{pmatrix} 1 & r_1 & & & 0 \\ & 1 & \ddots & & \\ & & \ddots & \ddots & \\ & & & \ddots & r_{n-1} \\ 0 & & & & 1 \end{pmatrix}$$

and let $M = \mathrm{diag}(\mu_i)$. Then

(16.1) $KMK^T =$



is a symmetric, tri-diagonal matrix.

Consider the matrix equation

(16.2) $$(D + \sigma uu^T)x = \lambda x \ .$$

Multiplying (16.2) on the left by K , we have

$$K(D + \sigma uu^T)K^T K^{-T} x = \lambda \, K K^T K^{-T} x$$

or

(16.3) $$(KD\,K^T + \sigma K u u^T K^T)y = \lambda \, K K^T y$$

where $x = K^T y$. Let us assume that we have re-ordered the elements of u so that

$$u_1 = u_2 = \cdots = u_{p\,1} = 0 \text{ and } 0 < |u_p| \le |u_{p+1}| \le \cdots \le |u_n| \ .$$

Now it is possible to determine the elements of K so that

$$(16.4) \qquad Ku = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ u_n \end{pmatrix} .$$

Specifically,

$$r_1 = 0 \qquad (i = 1,2,\ldots,p-1) ,$$

$$r_1 = -u_i/u_{i+1} \qquad (i = p,p+1,\ldots,n) ,$$

and we note that $|r_i| \leq 1$. (This device of using a bi-diagonal matrix for annihilating n-1 elements of a vector has been used by Björck and Pereyra [4] for inverting Vandermonde matrices.) Therefore, if Ku satisfies (16.4), we see from (16.1) that $KDK^T + \sigma Kuu^T K^T$ is a tri-diagonal matrix and similarly KK^T is a tri-diagonal matrix. Thus we have a problem of the form

$$Ay = \lambda By$$

where A and B are symmetric, tri-diagonal matrices and B is positive definite.

In [22], Peters and Wilkinson show how linear interpolation may be used effectively for computing the eigenvalues for such matrices when the eigenvalues are isolated. The algorithm makes use of the value of $\det(A - \lambda B)$. When A and B are tri-diagonal, it is very simple

to evaluate $\det(A - \lambda B)$ for arbitrary λ . Once the eigenvalues are computed it is easy to compute the eigenvectors by inverse iteration.

In Section 15, we showed it was necessary to compute a parameter λ^* which satisfied the equation

$$\det((\Omega + \lambda I)^2 - uu^T) = 0 \quad . \tag{16.5}$$

Again we can determine K so that Ku satisfies (16.4) and hence (16.5) is equivalent to

$$\det(K(\Omega + \lambda I)^2 K^T - Kuu^T K^T) = 0 \quad . \tag{16.6}$$

The matrix $G(\lambda) = K(\Omega + \lambda I)^2 K^T - Kuu^T K^T$ is tri-diagonal so that it is easy to evaluate $G(\lambda)$ and $\det G(\lambda)$. Since we have an upper and lower bound on λ^* , it is possible to use linear interpolation to find λ^* , even though $G(\lambda)$ is quadratic-in λ . Numerical experiments have indicated it is best to compute $G(\lambda) = K(\Omega + \lambda I)^2 K^T - Kuu^T K^T$ for each approximate value of λ^* rather than computing $G(\lambda) = (K\Omega^2 K^T - Kuu^T K^T) + 2\lambda K\Omega K^T + \lambda^2 K K^T$.

The device of changing modified eigensystems to tri-diagonal matrices and then using linear interpolation for finding the roots can be extended to matrices of the form

$$C = \left(\begin{array}{c|c} D & u \\ \hline u^T & \sigma \end{array}\right)$$

Again we choose K so that Ku satisfies (16.4) and thus obtain the eigenvalue problem $Ay = \lambda By$ where

$$A = \left(\begin{array}{c|c} KDK^T & Ku \\ \hline u^TK^T & \sigma \end{array}\right), \qquad B = \left(\begin{array}{c|c} KK^T & 0 \\ \hline 0 & 1 \end{array}\right)$$

so that A and B are both tri-diagonal and B is positive definite. Bounds for the eigenvalues of C can easily be established in terms of the eigenvalues of D and hence the linear interpolation algorithm may be used for determining the eigenvalues of C .

Bibliography

[1] Beale, E.M.L., "Numerical Methods," in Nonlinear Programming, J. Abadie (ed.). John Wiley, New York, 1967, pp. 133-205.

[2] Björck, Å., "Iterative Refinement of Linear Least Squares Solutions II," BIT **8** (1968), pp. 8-30.

[3] _____ and G. H. Golub, "Iterative Refinement of Linear Least Squares Solutions by Householder Transformations," BIT 7 (1967), pp. 322-37.

[4] _____ and V. Pereyra, "Solution of Vandermonde Systems of Equations," Publicación 70-02, Universidad Central de Venezuela, Caracas, Venezuela, 1970.

[5] Cottle, R. W. and G. B. Dantzig, "Complementary Pivot Theory of Mathematical Programming," Mathematics of the Decision Sciences, Part 1, G. B. Dantzig and A. F. Veinott (eds.), American Mathematical Society (1968), pp. 115-136.

[6] Dantzig, G. B., R. P. Harvey, R. D. McKnight, and S. S. Smith, 'Sparse Matrix Techniques in Two Mathematical Programming Codes," Proceedings of the Symposium on Sparse Matrices and Their Applications, T. J. Watson Research Publication RAl, no. 11707, 1969.

[7] Fletcher, R., "A Technique for Orthogonalization," J. Inst. Maths. Applics. 5 (1969), pp. **162-66.**

[8] Forsythe, G.E.,and G. H. Golub, "On the Stationary Values of a Second-Degree Polynomial on the Unit Sphere," J. SIAM, 13 (1965), pp. 1050-68.

[9] _____ and C. B. Moler, Computer Solution of Linear Algebraic Systems, Prentice-Hall, Englewood Cliffs, New Jersey, 1967.

[10] Francis, J., "The QR Transformation. A Unitary Analogue to the LR Transformation," Comput. J. 4 (1961-62), pp. 265-71.

[11] Golub, G. H., and C. Reinsch, "Singular Value Decomposition and Least Squares Solutions," Numer. Math., 14 (1970), pp. 403-20.

[12] _____, and R. Underwood, "Stationary Values of the Ratio of Quadratic Forms Subject to Linear Constraints," Technical Report No. CS 142, Computer Science Department, Stanford University, 1969.

[13] Hanson, R. J., "Computing Quadratic Programming Problems: Linear Inequality and Equality Constraints," Technical Memorandum No. 240, Jet Propulsion Laboratory, Pasadena, California, 1970.

[14] _____, and C. L. Lawson, "Extensions and Applications of the Householder Algorithm for Solving Linear Least Squares Problems," Math. Comp., 23 (1969), pp. 787-812.

[15] Householder, A.S., "Unitary Triangularization of a Nonsymmetric Matrix," J. Assoc. Comp. Mach., 5 (1958), pp. 339-42.

[16] Lanczos, c., Linear Differential Operators. Van Nostrand, London, 1961. Chapter 3.

[17] Leringe, Ö., and P. Wedin, "A Comparison Between Different Methods to Compute a Vector x Which Minimizes $\|Ax - b\|_2$ When Gx = h," . Technical Report, Department of Computer Sciences, Lund University, Sweden.

[18] Levenberg, K., "A Method for the Solution of Certain Non-Linear Problems in Least Squares," Quart. Appl. Math., 2 (1944), pp. 164-68.

[19] Marquardt, D. W., "An Algorithm for Least-Squares Estimation of Non-linear Parameters," J. SIAM, 11 (1963), pp. 431-41.

[20] Meyer, R. R., "Theoretical and Computational Aspects of Nonlinear Regression," P-1819, Shell Development Company, Emeryville, California.

[21] Penrose, R., "A Generalized Inverse for Matrices," Proceedings of the Cambridge Philosophical Society, 51 (1955), pp. 406-13.

[22] Peters, G., and J. H. Wilkinson, "Eigenvalues of $Ax = \lambda Bx$ with Band Symmetric A and B," Comput. J., 12 (1969), pp. 398-404.

[23] Powell, M.J.D., "Rank One Methods for Unconstrained Optimization," T.P. 372, Atomic Energy Research Establishment, Harwell, England, (1969).

[24] Rosen, J. B., "Gradient Projection Method for Non-linear Programming. Part I. Linear Constraints," J. SIAM, 8 (1960), pp. 181-217.

[25] Shanno, D. C., "Parameter Selection for Modified Newton Methods for Function Minimization," J. SIAM, Numer. Anal., Ser. B,7 (1970).

[26] Stoer, J., "On the Numerical Solution of Constrained Least Squares Problems" (private communication), 1970.

[27] Tewarson, R. P., "The Gaussian Elimination and Sparse Systems," Proceedings of the Symposium on Sparse Matrices and Their Applications, T. J. Watson Research Publication RA1, no. 11707, 1969.

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

[29] _____, "Error Analysis of Transformations Based on the Use of Matrices of the Form $I - 2ww^H$," in Error in Digital Computation, Vol. ii, L. B. Rall (ed.), John Wiley and Sons, Inc., New York, 1965, pp. 77- 101.

[30] _____, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

[31] Zoutendijk, G., Methods of Feasible Directions, Elsevier Publishing Company, Amsterdam (1960), pp. 80-90.

[32] _____, "Nonlinear Programming, Computational Methods," in Nonlinear and Integer Programming, J. Abadie (ed.), North-Holland Publ. Co., 1970, pp. 37-86.

Key words

Linear programming

Quadratic programming

Householder transformations

Matrix decompositions

Least squares

Non-linear least squares

Complementary pivoting