Part II

NUMERICAL MATHEMATICS

BIT 24 (1984), 206–224

SOLVING THE MINIMAL LEAST SQUARES PROBLEM SUBJECT TO BOUNDS ON THE VARIABLES

PER LÖTSTEDT [1])

Department of Numerical Analysis and Computing Science, The Royal Institute of Technology, S-100 44 Stockholm, Sweden

Abstract.

A computational procedure is developed for determining the solution of minimal length to a linear least squares problem subject to bounds on the variables. In the first stage, a solution to the least squares problem is computed and then in the second stage, the solution of minimal length is determined. The objective function in each step is minimized by an active set method adapted to the special structure of the problem.

The systems of linear equations satisfied by the descent direction and the Lagrange multipliers in the minimization algorithm are solved by direct methods based on QR decompositions or iterative preconditioned conjugate gradient methods. The direct and the iterative methods are compared in numerical experiments, where the solutions are sought to a sequence of related, minimal least squares problems subject to bounds on the variables. The application of the iterative methods to large, sparse problems is discussed briefly.

1. Introduction.

It is well-known that the solution to the linear least squares problem subject to bounds on the variables

$$\min \|Ax+b\|_2, \tag{1.1a}$$

$$c_i \leqq x_i \leqq d_i, \qquad i = 1, 2, \ldots, k, \tag{1.1b}$$

$$x_i \geqq c_i, \qquad i = k+1, \ldots, l,$$

$A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$, $l \leqq n$, is unique if A has full column rank, see e.g. p. 79 in [7]. If the columns of A are linearly dependent, i.e. $r = \operatorname{rank}(A) < n$, then there is in general a manifold M of solutions to (1.1). In order to determine a unique solution x to (1.1) even if A is rank-deficient, the value of x of minimal length in M satisfying

$$\min \|x\|_2, \tag{1.2}$$

$$x \in M = \{y | y \text{ solves } (1.1)\},$$

Received May 1983. Revised November 1983.

[1]) Present address: TKLUB2, Aerospace Division, SAAB-Scania, S-581 88 Linköping, Sweden.

This work was supported by The National Swedish Board for Technical Development under contract dnr 80-3341.

is chosen here. In a problem (1.1) without constraints, $l = 0$, the solution x to (1.1) and (1.2) is the Moore–Penrose pseudoinverse solution to the least squares problem (1.1) [1]. In this paper we shall develop numerical methods for solving (1.1) and (1.2), particularly suitable when a sequence of problems with a slowly changing matrix A^j, $j = 1, 2, \ldots$, is to be solved or for large, sparse problems. The first type of problem appears e.g. in the time-dependent simulation of contact problems for mechanical systems [16].

The problem (1.1) is a special case of the constrained least squares problem for which several algorithms have been proposed [6, 13, 15, 17, 19, 20]. We intend to exploit the structure of the constraints in (1.1) and do not need the full generality of these methods. The solution of (1.1) is also the solution of a quadratic programming (QP) problem with the objective function

$$\tfrac{1}{2}x^T G x + x^T h, \tag{1.3}$$

where $G = A^T A$ and $h = A^T b$, with the constraints (1.1b). Methods for the QP problem defined by (1.3) and (1.1b) are described in [7] and [10]. Iterative procedures for solving systems of linear equations have been adapted to the minimization of (1.3) over (1.1b) with a symmetric, positive definite G in [4] and [5] (the successive over-relaxation method) and in [18] (a preconditioned conjugate gradient (cg) method). See also the recent books by Fletcher [7] and Gill, Murray and Wright [12]. In most of the above mentioned papers the assumption is that A in (1.1a) has full column rank or that G in (1.3) is nonsingular. Here we allow $r = \text{rank}\,(A)$ to satisfy $r < n$ and determine the solution of minimal length.

This paper is organized as follows. In the next section the two algorithms for solving (1.1) and (1.2) are described. The systems of linear equations satisfied by the descent directions and the Lagrange multipliers are solved by direct methods or by the cg methods developed by Björck and Elfving [1]. A particular choice of preconditioning matrices for a sequence of related, dense problems (1.1) and (1.2) is justified in section 3. The numerical results are presented in the final section. The iterative methods are compared with the direct methods and conclusions are drawn from the experiments.

In the sequel $\|\cdot\|$ denotes the Euclidean vector norm or the subordinate spectral matrix norm. I is the identity matrix of appropriate dimension. The notation for the range and the nullspace of a matrix B is $R(B)$ and $N(B)$, respectively. The orthogonal projector on the space Y is denoted by P_Y. For simplicity, we assume that $c_i < d_i$, $i = 1, 2, \ldots, k$.

2. The minimization algorithms.

The solution to (1.1) and (1.2) is computed in two separate stages. First, a solution to (1.1) is determined. Second, the minimum value of x in (1.2) is

calculated. The second stage is not necessary if rank $(A) = n$. The active set strategy by Gill and Murray [11] for linearly constrained QP problems is modified for efficient handling of the least squares problem (1.1) and the least distance problem (1.2) taking their special properties into account. In a problem without constraints, $l = 0$, our method to solve (1.1) and (1.2) is identical with algorithm 6.2 in [1].

2.1. *The least squares problem.*

Assume that x is feasible, i.e. x fulfils (1.1b). Let the indices of x belong to one of the sets F, P or X. If $j \in X$ then x_j is *fixed* on its lower or upper bound, c_j or d_j. The constraint $x_j \geqq c_j$ or $x_j \leqq d_j$ is active. If $j \in F$ then x_j is a *free* variable. The remaining variables x_j, $j \in P = \{1, 2, \ldots, n\} \backslash \{F \cup X\}$, are termed *passive*. Let n_F be the number of free variables and let $E_F \in \mathbb{R}^{n \times n_F}$ be such that the vector $x_F = E_F^T x$ contains all the free variables. A_F consists of the columns a_j of A such that $j \in F$, i.e. $A_F = AE_F$. In the optimization algorithm we attempt to minimize

$$\|A_F x_F + b'\|, \qquad b' = b + \sum_{j \in X \cup P} a_j x_j,$$

while maintaining the feasibility of x. The free set is altered until certain optimality conditions are met. The algorithm for solving (1.1) is:

Algorithm 1.

Initialization.

1. Let x be feasible, let $X = \emptyset$ and choose the initial F and A_F.
2. Compute the residual $r = Ax + b$.

Main iteration loop.

3. If $F \neq \emptyset$ then compute a new descent direction p by solving

$$A_F^T A_F p + A_F^T r = 0. \tag{2.1}$$

Otherwise go to 7.

4. Let $\hat{\theta} p$, $\hat{\theta} \geqq 0$, be the maximal step possible in the direction p in the space of free variables without violating any constraints and let $\theta = \min(\hat{\theta}, 1)$. The step length θ satisfies

$$\begin{cases} \tau_i^- = (c_i - x_i)/p_i, & i \in F_- = \{j \mid j \leqq l, j \in F, p_j < 0\}, \\ \tau_i^+ = (d_i - x_i)/p_i, & i \in F_+ = \{j \mid j \leqq k, j \in F, p_j > 0\}. \\ \theta = \min\left(\min_{i \in F_-} \tau_i^-, \min_{i \in F_+} \tau_i^+, 1\right). \end{cases} \tag{2.2}$$

Note that p_i is not necessarily the ith component of the vector p.

5. Update x and r

$$\begin{cases} x_i := x_i + \theta p_i, & i \in F, \\ r := r + \theta A_F p. \end{cases} \tag{2.3}$$

6. If $\tau_i^- > 1$, $i \in F_-$, and $\tau_i^+ > 1$, $i \in F_+$, in step 4 then go to 7. For each x_j attaining its lower or upper bound in step 5, transfer j from F to X and remove the corresponding column in A_F. Go to 3.

7. Compute the gradient Δ of the objective function

$$\Delta = A^T r = A^T A x + A^T b. \tag{2.4}$$

8. Let

$$\delta_i = \begin{cases} \Delta_i, & x_i = d_i \\ -\Delta_i, & x_i = c_i \end{cases} \qquad i \in X.$$

Find γ defined by

$$\gamma = \max \left(\max_{i \in X} \delta_i, \max_{i \in P} |\Delta_i| \right),$$

and a j such that $\gamma = \delta_j$, $j \in X$, or $\gamma = |\Delta_j|$, $j \in P$. If $\gamma > \varepsilon$, a non-negative error tolerance, then the objective function can be decreased further if x_j becomes a free variable. Transfer j from X or P to F and add the corresponding column to A_F. Go to 3. If $\gamma \leqq \varepsilon$, then an approximate optimum has been found.

9. End of algorithm.

We shall outline a proof of convergence of the algorithm after a finite number of steps to an optimal solution of (1.1). Assume that $\varepsilon = 0$. The main iteration loop consists of two loops, an inner loop: steps 3–6, and an outer loop: steps 3–8. To reach step 7 we have $F = \emptyset$ or $\theta = 1$. If $\theta = 1$ then the actual step taken in the space of free variables satisfies (2.1). Hence, by (2.3) and (2.4) $E_F^T \Delta = 0$ and after termination of the algorithm the following relations are valid for x_i and Δ_i:

$$\begin{cases} i \in F\text{: } \Delta_i = 0, \text{ if } i \leqq l \text{ then } x_i \text{ satisfies the constraints in (1.1b)}, \\ i \in X\text{: if } x_i = c_i \text{ then } \Delta_i \geqq 0, \\ \qquad \text{if } x_i = d_i \text{ then } \Delta_i \leqq 0, \\ i \in P\text{: } \Delta_i = 0, \ x_i \text{ has not been changed by the algorithm.} \end{cases} \tag{2.5}$$

These are the necessary and sufficient Kuhn-Tucker conditions for a minimum of (1.1), see p. 159 in [15]. The finiteness of a similar algorithm is proved on pp.

160–164 in [15]. The termination of the inner and outer loops in algorithm 1 after a finite number of iterations can be proved along the same lines. At step 7 we have $A_F^T r = 0$. If we return from step 8 to step 3 then the column a_j added to A_F at step 8 is linearly independent of the previous columns in A_F since $\Delta_j = a_j^T r \neq 0$. Thus, if F is selected such that A_F has full column rank initially, then this property is preserved.

The system of linear equations (2.1) can be solved either by a direct method based on Householder transformations [2] or by the preconditioned conjugate gradient method CGPCNE devised by Björck and Elfving [1]. In the direct solution procedure suitable for an isolated, dense problem (1.1), a QR decomposition of A_F, $A_F = Q_F R_F$, is updated when F is changed and the descent direction p in (2.1) satisfies

$$R_F p + Q_F^T r = 0, \tag{2.6}$$

see pp. 180–181 in [12]. Methods for modifying the QR decomposition of a matrix when columns are added to or removed from the matrix are described in [9]. In step 1, F is chosen such that rank (A) = rank (A_F). The criterion for determining the rank of A is based on [14].

A preconditioning matrix C_F is introduced in [1] to improve the convergence rate when applying the cg method to

$$C_F^{-T} A_F^T A_F C_F^{-1} v + C_F^{-T} A_F^T r = 0. \tag{2.7}$$

If we take $p = C_F^{-1} v$ we have a solution to (2.1). The reason why we wish to solve (2.1) by a preconditioned cg method is that when a sequence of problems (1.1) is to be solved, where the variation of A is small from problem to problem, the same preconditioning matrix can be used for several consecutive problems. Numerical results are presented in section 4 with the direct method and with the iterative method and a preconditioning matrix C_F discussed in section 3.1. The results indicate that the iterative method is more economical than the direct method for a sequence of problems.

Fletcher and Jackson [8] have proposed a more efficient criterion for reintroducing a fixed variable into the set of free variables in step 8. Since the gains reported in [8] with their improved strategies are small in comparison with the simple criterion in step 8 for the QP problem with a positive definite G in (1.3), we have not attempted to include their ideas in algorithm 1. The algorithm by O'Leary [18] is designed for a symmetric, positive definite G in (1.3) and differs from algorithm 1 in the strategy for releasing variables from their bounds and in that the cg method is directly involved in the search for a new feasible point.

2.2. *A least distance problem.*

Split the solution x of (1.1) into two parts, $x = x_R + x_N$, $x_R \in R(A^T)$, $x_N \in N(A)$. According to prop. 3 in [16] x_R is uniquely determined, but any x_N such that x remains feasible is admissible. Since $\|x\|^2 = \|x_R\|^2 + \|x_N\|^2$, the problem (1.2) is equivalent to

$$\begin{cases} \min \|x_N\|, & x_N \in N(A), \\ c_i \leqq x_{Ri} + x_{Ni} \leqq d_i, & i = 1, 2, \ldots, k, \\ x_{Ri} + x_{Ni} \geqq c_i & i = k+1, \ldots, l. \end{cases} \tag{2.8}$$

Let (U_R, U) be the orthogonal matrix in the QR decomposition of A^T,

$$A^T = (U_R, U)S. \tag{2.9}$$

$N(A)$ is spanned by $U \in \mathbb{R}^{n \times (n-r)}$ and $R(A^T)$ by $U_R \in \mathbb{R}^{n \times r}$. Then there is a $z = U^T x \in \mathbb{R}^{n-r}$ such that $x_N = Uz$. The problem (2.8) can be rewritten as

$$\begin{cases} \min \|z\|, \\ c_i \leqq u_i z + x_{Ri} \leqq d_i, & i = 1, 2, \ldots, k, \\ u_i z + x_{Ri} \geqq c_i, & i = k+1, \ldots, l, \end{cases} \tag{2.10}$$

where u_i is the ith row of U. This is a least distance problem, see [7]. The algorithm for solving (2.10) is based on the QP algorithm by Gill and Murray [11]. A different procedure for least distance programming is given in [18].

Introduce the notation T for the *active* set of constraints,

$$T = \{i | x_i = c_i \text{ or } x_i = d_i\}.$$

The set of free variables is F, cf. algorithm 1. The input x to this algorithm is the output of algorithm 1.

Algorithm 2.

Initialization.

1. Choose F and let $T = \{1, 2, \ldots, n\} \backslash F$.

Compute $z = U^T x$ and $x_N = P_{N(A)} x$ and let $q = n - r - t$, where t is the number of elements in T.

Main iteration loop.

2. If $q > 0$, then compute a new descent direction $p = UZ_T s$, where Z_T and s satisfy

$$Z_T \in \mathbb{R}^{(n-r) \times q}, \quad Z_T^T Z_T = I, \quad u_i Z_T = 0, \quad i \in T, \quad s = -Z_T^T z. \tag{2.11}$$

Otherwise go to 6.

3. Let $\hat{\theta}p$, $\hat{\theta} \geqq 0$, be the maximal step possible in the direction p without violating any constraints and let $\theta = \min(\hat{\theta}, 1)$. The step length θ satisfies

$$\text{(2.12)} \quad \begin{cases} \tau_i^- = (c_i - x_i)/p_i, & i \in F_- = \{j | j \leqq l, j \in F, p_j < 0\}, \\ \tau_i^+ = (d_i - x_i)/p_i, & i \in F_+ = \{j | j \leqq k, j \in F, p_j > 0\}, \\ \theta = \min(\min_{i \in F_-} \tau_i^-, \min_{i \in F_+} \tau_i^+, 1). \end{cases}$$

4. Update z, x_N and x

$$\text{(2.13)} \quad \begin{cases} z := z + \theta Z_T s, \\ x_N := x_N + \theta p, \\ x := x + \theta p. \end{cases}$$

5. If $\tau_i^- > 1$, $i \in F_-$, and $\tau_i^+ > 1$, $i \in F_+$, in step 3 then go to 6. If an x_j attains its lower or upper bound in step 4, transfer j from F to T and update q, $q := q - 1$. Go to 2.

6. Compute the Lagrange multipliers λ_i in

$$\text{(2.14)} \quad \begin{cases} H = (u_1^T, u_2^T, \ldots, u_l^T), \\ H\lambda = z, \ \lambda \in \mathbb{R}^l, \\ \lambda_i = 0, \ i \in \{1, 2, \ldots, l\} \backslash T. \end{cases}$$

7. Let $T_- = \{j | j \in T, x_j = c_j\}$ and $T_+ = \{j | j \in T, x_j = d_j\}$.
Find γ defined by

$$\gamma = \max(-\min_{i \in T_-} \lambda_i, \max_{i \in T_+} \lambda_i)$$

and a j such that $\gamma = -\lambda_j$, $j \in T_-$, or $\gamma = \lambda_j$, $j \in T_+$. If $\gamma > \varepsilon$, a non-negative error tolerance, then $\|x_N\| = \|z\|$ can be decreased further if x_j becomes a free variable. Transfer j from T to F and increase q, $q := q + 1$. Go to 2. If $\gamma \leqq \varepsilon$, then an approximate optimum has been found.

8. End of algorithm.

A full step in the descent direction p, $\theta = 1$, minimizes $\frac{1}{2} z^T z$ subject to the constraints $u_i z + x_{Ri} = 0$, $i \in T$. The relations in (2.14) and

$$\text{(2.15)} \quad \begin{cases} \lambda_i \geqq 0, & i \in T_-, \\ \lambda_i \leqq 0, & i \in T_+, \end{cases}$$

are the necessary and sufficient Kuhn-Tucker conditions for the optimal solution of (2.10). These conditions are fulfilled in step 7 if $\gamma \leqq 0$. If only one new constraint j is included in T at step 5, then u_j is linearly independent of the

previously active constraint vectors u_i, $i \in T_{\text{old}}$, since $p_j = u_j Z_T s \neq 0$ but $p_i = u_i Z_T s = 0$, $i \in T_{\text{old}}$, cf. p. 201 in [12]. Let H_T consist of the columns u_i^T, $i \in T$. If T in step 1 is such that H_T has linearly independent columns initially then H_T has full column rank throughout the iteration. If $q = 0$ in step 2, then there is no descent direction and Z_T is null. In algorithm 1, we have assumed that $c_i < d_i$ and at any point x the binding constraints cannot be linearly dependent. Contrary to the case in algorithm 1, the constraints in algorithm 2 satisfied as equalities may be linearly dependent. According to p. 201 in [12] there is a risk of cycling, i.e. we return to the same set F repeatedly without decreasing $\|z\|$. However, this phenomenon has not been observed in the numerical experiments in section 4.

The question now arises how to compute the descent direction in step 2 and the Lagrange multipliers in step 6. We shall discuss two possibilities: one direct method and one iterative method.

In the direct method, the QR decomposition of A^T is first determined by Householder transformations [2]. The first l rows of U in (2.9) define the constraints in (2.10) and $P_{N(A)} = UU^T$. The decomposition of H_T is

$$H_T = V \begin{pmatrix} R \\ 0 \end{pmatrix} = V_1 R, \qquad V = (V_1, V_2), \qquad V_2^T H_T = 0, \tag{2.16}$$

where V is orthogonal and R upper triangular and nonsingular. Since V_2 spans $N(H_T^T)$ we choose $Z_T = V_2$ in (2.11). The vector of components of λ in (2.14) corresponding to the columns of H_T is denoted by λ_T. By (2.16) λ_T satisfies

$$\lambda_T = R^{-1} V_1^T z. \tag{2.17}$$

The factorization (2.16) is updated by the methods in [9] when a constraint is added to T in step 5 or dropped from T in step 7. Initially, we take $T = \emptyset$.

The iterative method is based on the algorithm CGPCMN by Björck and Elfving [1]. CGPCMN is a preconditioned cg algorithm for solution of the consistent system

$$Bu = BB^T v = f, \qquad f \in R(B). \tag{2.18}$$

With a proper choice of preconditioning matrix D, the convergence rate of the cg iterations is faster when

$$D^{-T} BB^T D^{-1} w = D^{-T} f, \qquad v = D^{-1} w, \tag{2.19}$$

is solved instead of (2.18).

In the algorithm we search for a minimum in the subspace $N(A_F)$. The

resulting p in step 3 is

$$p = -UZ_T Z_T^T U^T x. \tag{2.20}$$

The matrix $W = UZ_T$ is orthogonal and WW^T is an orthogonal projector. Define $U_F = E_F^T U$ and $p_F = E_F^T p$. By (2.11) $W_F = U_F Z_T$ is also orthogonal and $A_F W_F = AW = 0$. The dimension of $N(A_F)$ is equal to the number of degrees of freedom of z, which is q according to (2.11). Thus, W_F spans $N(A_F)$. Use CGPCMN to solve (2.18) with $B = A_F$, $f = 0$ and the initial approximation $u = -x_F$,

$$A_F u = 0. \tag{2.21}$$

It follows from p. 158 in [1] that after termination the solution is $u = -P_{N(A_F)} x_F$. We draw the conclusion from (2.20) that $u = -W_F W_F^T x_F = p_F$. The descent direction p in step 2 is easily constructed from p_F and the fact that $p_i = 0$, $i \in T$.

After a full step in the descent direction, $\theta = 1$, we infer from (2.13) and (2.11) that

$$P_{R(Z_T)} z = Z_T Z_T^T z = 0. \tag{2.22}$$

Augment λ by the components $\lambda_i = 0$, $i = l+1, \ldots, n$, so that $\lambda \in \mathbb{R}^n$. Then by (2.14), z and λ satisfy

$$U^T \lambda = z \tag{2.23}$$

at step 6.

It follows from (2.11) that

$$P_{R(Z_T)} u_i^T = \begin{cases} 0, & i \in T, \\ Z_T Z_T^T u_i^T, & i \in F. \end{cases} \tag{2.24}$$

Combine (2.22), (2.23) and (2.24) and let $\lambda_F = E_F^T \lambda$ to obtain the identity

$$0 = P_{R(Z_T)} z = Z_T W_F^T \lambda_F.$$

Since the columns of Z_T are linearly independent we have

$$P_{N(A_F)} \lambda_F = 0 \tag{2.25}$$

at step 6. A λ satisfying (2.23) can be written

$$\lambda = Uz + A^T y = x_N + A^T y, \tag{2.26}$$

where y is arbitrary. According to (2.14) we have $\lambda_F = 0$ in step 6. Taking (2.25) into account, an additional condition on the components in λ_F is $P_{R(A_F^T)}\lambda_F = 0$ which is equivalent to $A_F\lambda_F = 0$. Therefore, by (2.26) y solves

$$A_F A_F^T y + A_F x_{NF} = 0, \qquad x_{NF} = E_F^T x_N. \tag{2.27}$$

This equation (2.27) is solved by CGPCMN with $B = A_F$ and $f = -A_F x_{NF}$ in (2.18). The resulting value of y is inserted into (2.26) to obtain λ_i, $i \in T$, used in the tests in step 7. The dimension q of $N(A_F)$ is reduced by one when a column a_j is removed in step 5 and is increased by one when A_F is augmented by a column a_j in step 7. Hence, the rank of A_F is constant and there is an h_j such that $a_j = A_F h_j$, $j \in T$. Suppose that y_1 and y_2 satisfy (2.27), $A_F A_F^T(y_1 - y_2) = 0$. Since $N(A_F A_F^T) = N(A_F^T)$, we find $A_F^T(y_1 - y_2) = 0$. For $j \in T$ the same result is valid, $a_j^T(y_1 - y_2) = h_j^T A_F^T(y_1 - y_2) = 0$, and consequently, $A^T(y_1 - y_2) = 0$. The computed λ in (2.26) is unique. The computation of z in the steps 1 and 4 is not necessary in the iterative method. For a certain application, a preconditioning matrix D_F for (2.21) and (2.27) is proposed in section 3.2. Numerical test runs are reported in section 4.

The sequence of approximations to the optimal value of x is the same in both the direct and the iterative method. The advantage of the iterative method is that we do not have to determine U in (2.9) and V_2 in (2.16) explicitly.

The value of Δ in (2.4) is not altered by algorithm 2. The Kuhn-Tucker conditions (2.5) are fulfilled also at step 8 of algorithm 2. If $\Delta_i \neq 0$ in the end of algorithm 1, then we conclude that $x_i = c_i$ or $x_i = d_i$ for all steps of algorithm 2 and one would expect that $X \subset T$. Unfortunately, the columns of H in (2.14) may be linearly dependent if we take $T = X$ and we do not know which constraints are binding in problem (2.10) with Lagrange multipliers satisfying (2.14) and (2.15). Consider the following illustrative example.

$$\begin{aligned} &\min \left\| \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} -2 \\ 1 \end{pmatrix} \right\| \\ &0 \leqq x_1 \leqq 1, \\ &x_2 \geqq 0. \end{aligned} \tag{2.28}$$

The solution is $(1, 0)$ and $\Delta^T = (-1, 1)$. It is easily shown that the problem corresponding to (1.2) and (2.10) is

$$\begin{cases} \min z^2, \\ 0 \leqq z/\sqrt{2} + 0.5 \leqq 1, \\ z/\sqrt{2} - 0.5 \geqq 0. \end{cases} \tag{2.29}$$

The binding constraints from (2.28) are $x_1 = 1$ and $x_2 = 0$. Let $T = X$ in (2.29).

Then $H_T = (1, 1)/\sqrt{2}$. The columns of H_T are linearly dependent and λ in (2.14) is not uniquely determined. The correct solution is $\lambda^T = (0, 1)$.

We conclude this section by a few comments on the solution of a problem with a sparse matrix A. An equation of the form (2.1) is solved in [1] by the cg method in order to take advantage of the sparsity of A_F. Note that if A is sparse then this property is normally inherited by A_F. Algorithm 1 in combination with CGPCNE and a preconditioning matrix C_F in (2.7), which is simple to update when F is changed, is a possible alternative for sparse problems (1.1). Similarly, by solving the two systems (2.21) and (2.27) in the algorithm for the least distance problem (2.10) with CGPCMN we can utilize the sparsity in order to reduce the storage requirements. The preconditioning matrix D in (2.19) should also be chosen such that it is updated conveniently when F and A_F are altered.

3. The preconditioning matrices.

In this section transformations of the systems of linear equations (2.1), (2.21) and (2.27) are suggested to improve the convergence rate of the conjugate gradient method. The preconditioning in [1] is well suited for sparse problems. We focus here on preconditioning matrices with the purpose of increasing the efficiency when solving a sequence of related problems (1.1) and (1.2) defined by (A^j, b^j, c^j, d^j), $j = 1, 2, \ldots$, with dense matrices A^j. An application from engineering physics, where such a sequence of problems is to be solved, is the simulation of time-dependent contact and friction problems in rigid body mechanics. Approximations of the positions and velocities of the bodies are computed at discrete time points $t_1, t_2, \ldots$. The contact and friction forces in the system of bodies at t_j are the solution of a time-dependent least squares problem. If $t_j = t_{j-1} + h$, then $\|A^j - A^{j-1}\| = O(h)$. The columns of A^j are often linearly dependent in this application. For further details see [16].

3.1. *The least squares problem.*

The normal equations for the restricted least squares problem, $\min \|A_F^j p + r\|$, are solved in step 3 of algorithm 1. A possible choice of C_F in (2.7) is to let $C_F = R_F$, where R_F is upper triangular and Q_F is orthogonal in an approximate QR decomposition $Q_F R_F$ of A_F^j. Let F be a maximal set of linearly independent columns in A^1 at step 1 for $j = 1$. Compute the decomposition $A_F^1 = Q_F R_F$. Taking R_F from the exact QR decomposition of A_F^1 as preconditioning matrix C_F, the cg method CGPCNE [1] converges immediately in the first iteration. As the free set F in step 1 for $j = N+1$, F_1^{N+1}, take F in step 9 for $j = N$, F_9^N. The initial preconditioning matrix for $j = N+1$ is the same as the last matrix for $j = N$. A fresh QR decomposition of A_F^{N+1} is determined at step 1 only if the rate of convergence was too slow for $j = N$. When A_F^j is altered in step 6, the corresponding column is removed from R_F using the formulas in [9]. If a column a_i^j is added to A_F^j in step 8, then R_F is updated with a_i^j.

If the difference in the data (A^j, b^j, c^j, d^j) is small between $j = N$ and $j = N+1$, then the differences in the solutions x^j and the sets X^j and F^j at x^j, $j = N, N+1$, are usually small.

The rank of A_F^{N+1} at step 1 defined by F_9^N may be less than rank(R_F). The cg process will still converge to a solution of (2.1) and (2.7), see [1]. It is possible to extend the convergence proof of algorithm 1 to the case where the columns of A_F are linearly dependent by slightly modifying the ideas in sect. 2.1 and ch. 23 in [15].

3.2. *Computation of the minimal solution.*

Two systems of the form (2.18) are solved in the iterative version of algorithm 2. The solution to (2.21) defines the descent direction in step 2. The Lagrange multipliers in step 6 are determined by the solution to (2.27). We transform the two systems (2.21) and (2.27) by the same preconditioning matrix $D = D_F$ in (2.19).

After permutation of the rows, every B in (2.18) can be written as

$$B^T = (B_1^T, B_1^T B_2), \tag{3.1}$$

where B_1 has full row rank and rank (B) = rank (B_1). Since $f \in R(B)$ there is a g with the property $f = Bg$. By the definition (3.1), the solutions u_1 and v_1 to

$$B_1 u_1 = B_1 B_1^T v_1 = B_1 g = f_1$$

are also solutions to (2.18) with $u = u_1$ and $v = (v_1^T, 0)^T$. Let $A_f \in \mathbb{R}^{r \times n_F}$, where r = rank (A_F) and $n_F = r+q$, denote a submatrix of A_F corresponding to B_1 in (3.1). The solution u of

$$A_f u = 0 \tag{3.2}$$

satisfies (2.21) and $y = (y_1^T, 0)^T$, where y_1 solves

$$A_f A_f^T y_1 + A_f x_{NF} = 0, \tag{3.3}$$

satisfies (2.27). Compute the QR decomposition of A_F^T by Householder transformations

$$A_F^T = Q_F \left(\begin{array}{c|c} R_f & R' \\ \hline 0 & 0 \end{array} \right), \tag{3.4}$$

and determine the rank of A_F simultaneously, see [14]. $R_f \in \mathbb{R}^{r \times r}$ is nonsingular and upper triangular and $Q_F \in \mathbb{R}^{n_F \times n_F}$ is orthogonal. Let $A_f = (R_f^T, 0) Q_F^T$ and apply CGPCMN in [1] to (3.2) and (3.3) with the preconditioning matrix

$D_F = R_f$. The factorization (3.4) is updated when rows are added to and deleted from A_F^T by the methods on p. 529 in [9] and p. 227 in [15].

When we wish to solve a series of problems, we proceed as in the least squares problem. In the first problem of the sequence, $j = 1$, we take $T = \emptyset$ and compute the factorization (3.4) to obtain the preconditioning matrix R_f. When the number of columns of A_F^j is changed then R_f is updated. A new decomposition of A_F^j is determined only if motivated by efficiency. Usually, we choose the set F_1^{N+1} identical to F_9^N as in section 3.1, but if there is a $j \in F_9^N$ such that the result x_j from algorithm 1 is strictly inside the feasible region, then j is included in F_1^{N+1} and A_F and R_f are modified accordingly. Since we need the projection $x_N = P_{N(A)}x$ initially in step 1 and later in (2.13) and (2.27), the QR decomposition of $(A^1)^T$ is saved for use in future computations of x_N by CGPCMN, cf. (2.21). The rows of A_F^{N+1} to be included in A_f^{N+1} are determined by R_f from the previous problem. In exceptional cases, we may have rank $(A_F^{N+1}) <$ rank (R_f). The algorithm CGPCMN still converges to a solution of (3.2) or (3.3) according to [1]. A more serious event is when rank(A_F^j) increases between $j = N$ and $j = N+1$. Then there is a row a_* of A_F^{N+1}, linearly independent of the rows in A_f^{N+1} defined by R_f, such that $((A_f^{N+1})^T, a_*^T)$ is the matrix corresponding to B_1^T in (3.1). Hence, (2.21) and (2.27) will in general not be satisfied by the solutions to (3.2) and (3.3). A remedy is to compute the residual of (2.21) or (2.27) the first time they are solved in algorithm 2. If this residual is not sufficiently small, then we need a new factorization (3.4).

4. Numerical results and discussion.

Numerical experiments with an implementation of the preconditioned cg method for solution of (1.1) and (1.2) developed in sections 2 and 3 are reported in this section. The direct methods are compared with the iterative methods applied to sequences of problems with dense matrices A^j and small differences between the data (A^j, b^j, c^j, d^j) for consecutive problems.

We generate A^j and b^j, $j = 1, 2, \ldots$, in the following way:

$$
\text{(4.1)}\quad \begin{cases} A^j = K_1^j K_2^j, \quad K_1^j \in \mathbb{R}^{m \times R}, \quad K_2^j \in \mathbb{R}^{R \times n}, \quad 0 < R \leqq \min(m, n), \\ K_{1pq}^1 = \text{uniform } [-2, 2], \text{ a uniformly distributed random number in } [-2, 2], \\ K_{2pq}^1 = \text{uniform } [-2, 2], \\ b_p^1 = \text{uniform } [-n, n], \\ K_{ipq}^{j+1} = K_{ipq}^j + \text{uniform } [-0.02, 0.02], \quad i = 1, 2, \\ b_p^{j+1} = b_p^j + \text{uniform } [-0.02, 0.02]. \end{cases}
$$

The lower and upper bounds in (1.1b) are $c_i = -1$ and $d_j = 1$ for $i \leqq k$ and $c_i = 0$ for $k < i \leqq l$, i.e. $c^j = c$, $d^j = d$, for all j. The very first x in each sequence

of problems is also generated by random numbers consistent with the constraints.

The direct algorithm for solving (1.1) is implemented according to section 2.1 in the routine QR1. The routine CG1 is coded following the description in sections 2.1 and 3.1 of the iterative method for (1.1) with one exception. When an exact QR decomposition of A_F is available, then (2.6) is solved instead of (2.7). Hence, QR1 and CG1 are identical for the first problem in a sequence.

We interrupt the cg procedure when $\|A_F^T A_F p_* + A_F^T r\| < \varepsilon/\|R_F^{-1}\|$ and accept p_* as solution to (2.1). It is shown in [16] that the error δp in p_* satisfies $\|A_F \delta p\| < \varepsilon$. An estimate of $\|R_F^{-1}\|$ is obtained by the method in [3].

The routine, where the direct method in section 2.2 for solving (1.2) is implemented, is denoted by QR2. The routine for the iterative method in sections 2.2 and 3.2 is written in two versions, CG2a and CG2b. We take $T = \emptyset$ in step 1 of algorithm 2 in CG2a and save only the QR decomposition of A^T for the next problem in the sequence. In exact arithmetic, the routines QR2 and CG2a follow the same path $x_0, x_1, x_2, \ldots$ in their search for an optimal x. The routine CG2b is an implementation of the iterative method precisely as described in sections 2.2 and 3.2. The cg iterations are terminated in both CG2a and CG2b when the residual r of (3.2) or (3.3) fulfils $\|r\| < \varepsilon$. If convergence is not achieved in 20 iterations, a new factorization (3.4) is determined.

In the examples, the performance of the routines is measured by the number of flops required for convergence to an optimal value. A flop is here defined as one addition or subtraction and one multiplication or division in floating point arithmetic. We use the same sequence of random numbers $u_0, u_1, u_2, \ldots$ to generate the data for each set of test examples. Moreover, A^j is a square matrix, $m = n$ in (4.1). By constructing A^j as in (4.1) we have $r = \text{rank}\,(A^j) \leqq R$ and almost always $r = R$. The error tolerance ε is 10^{-4}. The programs were run in single precision on a DEC-10 computer with a machine unit of $0.7 \cdot 10^{-8}$. A routine in one of the two groups QR1, CG1 and QR2, CG2a, CG2b is initialized with the same x as the other member(s) of the same group. Let the difference between the results produced by two members of a group be δ. Then δ satisfies $\|\delta\| < 10^{-5}$ in almost all examples below.

In the first sequence of problems $m = n = 20$ and $R = 10$ in (4.1) and $k = 7$ and $l = 14$ in (1.1b). Table 1 displays the number of flops required when QR1 or CG1 solves (1.1) and QR2, CG2a or CG2b solves (1.2) with the data (A^j, b^j, c, d), $j = 1, 2, \ldots, 10$. The number of variables on a bound on termination of algorithm 2 is denoted by *nvb*. The average number of flops per problem is also shown. The graphs of the quotients

$$(4.2)\quad \begin{cases} Q_{\text{I}} = (\text{CG1 flops})/(\text{QR1 flops}), \\ Q_{\text{II}} = (\text{CG2a flops})/(\text{QR2 flops}), \\ Q_{\text{III}} = (\text{CG2b flops})/(\text{QR2 flops}), \end{cases}$$

Table 1. *The number of flops required by the routines QR1, CG1, and QR2, CG2a, CG2b for solving a sequence of problems (1.1) and (1.2).*

j	QR1	CG1	QR2	CG2a	CG2b	nvb
1	29108	29108	15018	28335	29125	4
2	17816	11495	13921	22075	7753	4
3	14141	4514	13921	25374	9382	4
4	11922	4514	13920	25378	9382	4
5	18131	4513	13340	21166	21903	3
6	18401	5084	13919	25890	14365	4
7	16077	5084	13920	28162	10496	4
8	14418	5085	13919	28674	11009	4
9	18089	5085	13920	27557	9893	4
10	16483	5085	13921	26938	10409	4
average	17459	7957	13972	25955	13372	4

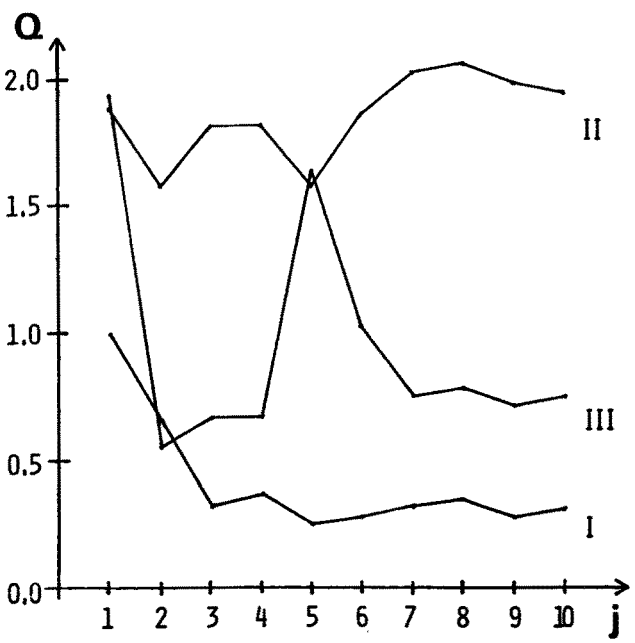

Fig. 1. A sequence of minimal least squares problems is solved, $j = 1, 2, \ldots 10$, $m = n = 20$, $R = 10$. The quotients (4.2) are plotted for each value of j.

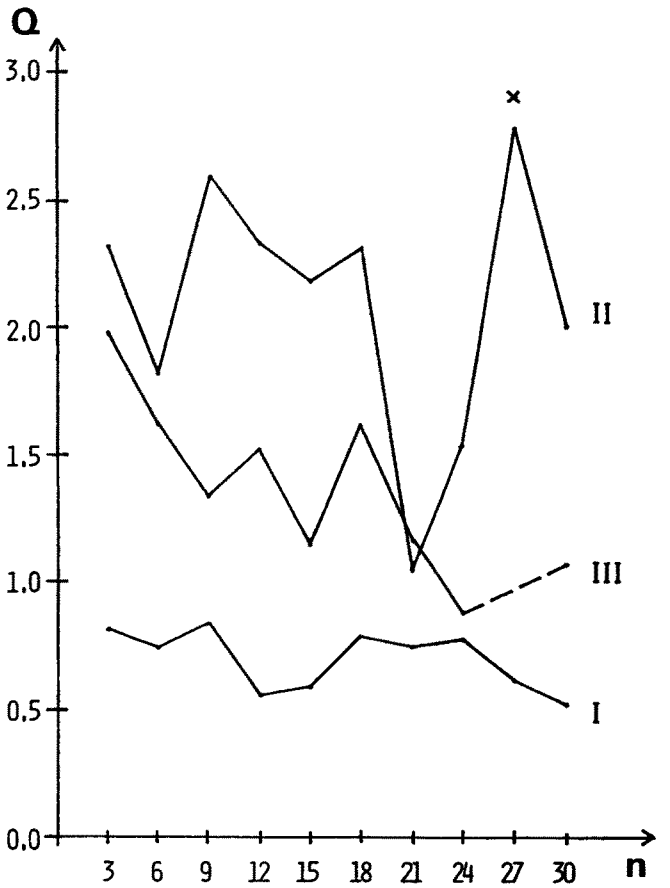

Fig. 2. The number of variables in the least squares problem is n, $m = n$. The performance of the methods is compared for a sequence of three consecutive problems for each n.

are drawn in fig. 1. Evidently, CG1 is superior to QR1 for a sequence of related problems. The performance of CG2b is more irregular in comparison with QR2. In the fifth problem the number of binding constraints in the solution is changed, which increases the amount of computation in CG2b. The routine CG2a is always slower than QR2.

In the second set of examples, we take $R = \lfloor (n+1)/2 \rfloor$, $k = n/3$ and $l \doteq 2k$. The quotients in (4.2) for the sums of flops over a sequence of three problems, $j = 1, 2, 3$ for each n, are shown in fig. 2. The routine CG1 is better than QR1, $Q_{\mathrm{I}} < 1$, for all n. Recall from table 1 and fig. 1 that this advantage is often more pronounced for longer sequences of problems. The trend in Q_{III} is that its value decreases and approaches 1 as n grows. For larger problems and longer sequences, CG2b may be the best choice. The *nvb*-vector consisting of the average number of variables on their bounds is $(1, 1, 4, 6, 5, 7, 1, 2, 11, 8)$ for $n = 3, 6, \ldots, 30$. The value of Q_{III} for $n = 27$ is marked by a cross. An extra QR

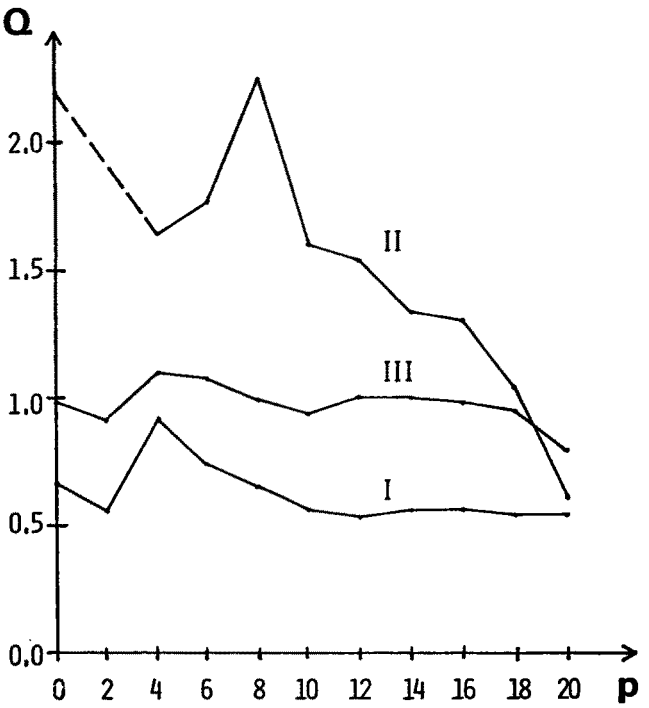

Fig. 3. The number of unconstrained variables is p and $m = n = 20$, $R = 10$. The quotients (4.2) are displayed for each p for a sequence of three problems.

decomposition of A_F was necessary in CG2b for $n = 27$ due to slow convergence in the cg process. With a different random number sequence for $n = 27$ we obtained $Q_{\mathrm{I}} = 0.705$, $Q_{\mathrm{II}} = 1.39$, $Q_{\mathrm{III}} = 1.08$ and the average *nvb* was 3. The performance of CG2a is worse than that of QR2 for all n.

The number of bounds on the variables is varied in fig. 3 and $m = n = 20$ and $R = 10$. Let p be the number of unconstrained variables, i.e. $l = 20 - p$, and let $k = l/2$. Like the previous figure, fig. 3 displays the quotients in (4.2) for a sequence of three problems. Also in this case CG1 is better than QR1 and CG2b is as efficient as QR2. The improvement in the performance of CG2a is significant as p grows. For $p = 2$, $Q_{\mathrm{II}} = 4.06$ because of slow convergence and an extra factorization (3.4). When $p = 20$ we actually compute the Moore-Penrose pseudoinverse solution to an unconstrained least squares problem. The *nvb*-vector is (8, 13, 4, 4, 8, 4, 3, 2, 2, 1, 0).

In the last set of examples, $m = n = 20$, $k = 7$ and $l = 14$. The graphs of the quotients in (4.2) for sequences of three problems with different values of R are shown in fig. 4. We find that CG1 is more efficient than QR1. For small values of R, QR2 is inferior to CG2b and even CG2a. The broken lines in fig. 4 indicate that values of Q have been omitted in the plots. Q_{III} for $R = 12$ is marked by a cross and $Q_{\mathrm{II}} = 7.55$ for $R = 14$. These large values are caused by

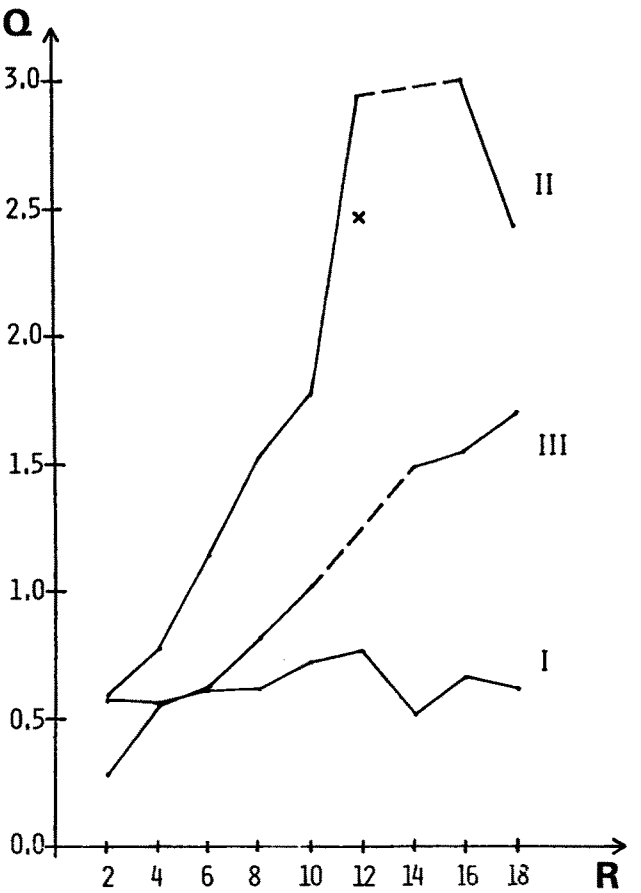

Fig. 4. The rank of the matrix in the objective function is varied for $m = n = 20$. The performance of the methods is measured by computing the quotients (4.2) for a sequence of three problems for each R.

slow convergence of the cg method and extra QR decompositions. The *nvb*-vector is (6, 2, 5, 5, 4, 7, 8, 5, 7).

A conclusion we can draw from our investigation is that CG1 is more effective than QR1 in solving sequences of relatively small, dense, least squares problems (1.1). The situation is less clear for the problem (1.2). For moderately long sequences, we can expect CG2b to be better on the average than QR2, see table 1 and fig. 1. However, the behavior of CG2b seems to be less predictable than that of QR2. For an isolated problem or a short sequence, the high initial cost of CG2b makes it less attractive and for long sequences, the efficiency of the original preconditioning matrix deteriorates gradually. An iterative method seems to be the best choice if rank (A) is low, see fig. 4. This is the only case where the performance of CG2a is better than of QR2 for constrained problems. The value of Q_{III} is probably lower than in fig. 3 for longer sequences with $j > 3$.

The initial cost of determining the QR decomposition of A^T in (2.9) in QR2 is

high, but the cost per iteration in algorithm 2 is low. Conversely, there is no such initial cost in CG2b for $j > 1$ but a high cost per iteration: solving (2.21) and (2.27) by the cg method and the updating of the decomposition (3.4). Therefore, if the number of variables on their bounds in the solution is almost constant so that the number of iterations in algorithm 2 is small, then CG2b is more competitive in comparison with QR2.

In [16] we have decided to use CG1 in combination with QR2, mainly because QR2 appears to be more robust and exhibits more regular behavior than CG2b. No experiments have been performed with the iterative methods and other preconditioning matrices applied to large and sparse problems.

REFERENCES

1. Å. Björck and T. Elfving, *Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations.* BIT 19 (1979), 145–163.
2. P. Businger and G. H. Golub, *Linear least squares solutions by Householder transformations.* Num. Math. 7 (1965), 269–276.
3. A. K. Cline, C. B. Moler, G. W. Stewart and J. H. Wilkinson, *An estimate for the condition number of a matrix.* SIAM J. Numer. Anal. 16 (1979), 368–375.
4. C. Cryer, *The solution of a quadratic programming problem using systematic overrelaxation.* SIAM J. Control 9 (1971), 385–392.
5. U. Eckhardt, *Quadratic programming by successive overrelaxation.* Report Jül-1064-MA, Kernforschungsanlage, Jülich (1974).
6. L. Eldén, *Numerical analysis of regularization and constrained least squares methods.* Ph.D. thesis, Dept. of Mathematics, Linköping University, Linköping (1977).
7. R. Fletcher, *Practical Methods of Optimization, vol. 2, Constrained Optimization.* Wiley, Chichester-New York (1981).
8. R. Fletcher and M. P. Jackson, *Minimization of a quadratic function of many variables subject only to lower and upper bounds.* J. Inst. Maths Applics 14 (1974), 159–174.
9. P. E. Gill, G. H. Golub, W. Murray and M. A. Saunders, *Methods for modifying matrix factorizations.* Math. Comp. 28 (1974), 505–535.
10. P. E. Gill and W. Murray, *Minimization subject to bounds on the variables.* NPL report NAC 72, National Physical Laboratory, Teddington (1976).
11. P. E. Gill and W. Murray, *Numerically stable methods for quadratic programming.* Math. Prog. 14 (1978), 349–372.
12. P. E. Gill, W. Murray and M. H. Wright, *Practical Optimization.* Academic Press, London-New York (1981).
13. K. H. Haskell, and R. J. Hanson, *An algorithm for linear least squares problems with equality and nonnegativity constraints.* Math. Prog. 21 (1981), 98–118.
14. I. Karasalo, *A criterion for truncation of the QR-decomposition algorithm for the singular linear least squares problem.* BIT 14 (1974), 156–166.
15. C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems.* Prentice-Hall, Englewood Cliffs, NJ (1974).
16. P. Lötstedt, *Numerical simulation of time-dependent contact and friction problems in rigid body mechanics.* Report TRITA-NA-8214, Dept. of Num. Anal. and Comp. Science, Royal Institute of Technology, Stockholm (1982) (to appear in SIAM J. Sci. Stat. Comput 5 (1984)).
17. R. Mifflin, *A stable method for solving certain constrained least squares problems.* Math. Prog. 16 (1979), 141–158.
18. D. P. O'Leary, *A generalized conjugate gradient algorithm for solving a class of quadratic programming problems.* Lin. Alg. Appl. 34 (1980), 371–399.
19. K. Schittkowski and J. Stoer. *A factorization method for the solution of constrained linear least squares problems allowing subsequent data changes.* Num. Math. 31 (1979), 431–463.
20. J. Stoer, *On the numerical solution of constrained least-squares problems.* SIAM J. Num. Anal. 8 (1971), 382–411.