BIT 19 (1979), 145–163

ACCELERATED PROJECTION METHODS FOR COMPUTING PSEUDOINVERSE SOLUTIONS OF SYSTEMS OF LINEAR EQUATIONS

ÅKE BJÖRCK and TOMMY ELFVING

Abstract.

Iterative methods are developed for computing the Moore–Penrose pseudoinverse solution of a linear system $Ax=b$, where A is an $m\times n$ sparse matrix. The methods do not require the explicit formation of A^TA or AA^T and therefore are advantageous to use when these matrices are much less sparse than A itself. The methods are based on solving the two related systems (i) $x=A^Ty$, $AA^Ty=b$, and (ii) $A^TAx=A^Tb$. First it is shown how the *SOR*- and *SSOR*-methods for these two systems can be implemented efficiently. Further, the acceleration of the *SSOR*-method by Chebyshev semi-iteration and the conjugate gradient method is discussed. In particular it is shown that the *SSOR-cg* method for (i) and (ii) can be implemented in such a way that each step requires only two sweeps through successive rows and columns of A respectively. In the general rank deficient and inconsistent case it is shown how the pseudoinverse solution can be computed by a two step procedure. Some possible applications are mentioned and numerical results are given for some problems from picture reconstruction.

1. Introduction.

Let A be a given $m\times n$ sparse matrix, b a given m-vector and $x=A^+b$ the Moore–Penrose pseudoinverse solution of the linear system of equations

$$Ax = b\,. \tag{1.1}$$

We denote the range and nullspace of a matrix A by $R(A)$ and $N(A)$ respectively. Convenient characterizations of the pseudoinverse solution are given in the following two lemmas.

LEMMA 1.1. *$x=A^+b$ is the unique solution of the problem: minimize $\|x\|_2$ when $x\in\{x;\ \|b-Ax\|_2=\text{minimum}\}$.*

LEMMA 1.2. *$x=A^+b$ is the unique vector which satisfies $x\in R(A^T)$ and $(b-Ax)\perp R(A)$, or equivalently $x\perp N(A)$ and $(b-Ax)\in N(A^T)$.*

These lemmas are easily proved by using the singular value decomposition of A and the resulting expression for A^+ (see Stewart [32], pp. 317–326).

Received June 14, 1978. Revised March 6, 1979.

In two important special cases the pseudoinverse solution can be obtained as the solution of one of the two Gaussian transformations of the original system (1.1).

THEOREM 1.1. *Assume that the system* (1.1) *is consistent, i.e.* $b \in R(A)$. *Then the solution* x *of*

$$x = A^T y, \quad AA^T y = b \tag{1.2}$$

is the pseudoinverse solution $x = A^+ b$.

PROOF. By construction we have $x \in R(A^T)$. Since $R(AA^T) = R(A)$ the system $AA^T y = b$ is also consistent and it follows that $0 = (b - AA^T y) = (b - Ax) \perp R(A)$. Thus, by lemma 1.2, x is the pseudoinverse solution. ∎

THEOREM 1.2. *Let* x *be a solution of*

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

and assume that $x \in R(A^T)$. *Then* x *is the pseudoinverse solution* $A^+ b$.

PROOF. From (1.3) it follows directly that $(b - Ax) \perp R(A)$. Thus, by lemma 1.2, x is the pseudoinverse solution. ∎

Note that the assumption in theorem 1.1 is trivially satisfied if rank $(A) = m$, and similarly that in theorem 1.2 if rank $(A) = n$. In the general case the pseudoinverse solution can be computed by a two-step procedure. First an arbitrary solution of (1.3) is computed and then the pseudoinverse solution is obtained by solving (1.2) with a modified right hand side. This procedure is further discussed in section 6.

The two systems of equations (1.2) and (1.3) are symmetric and positive semidefinite. The aim of this paper is to develop iterative methods for solving these systems, which *do not require the explicit computation of* AA^T *or* $A^T A$. We remark that (1.2) is the natural dual problem to (1.3) and therefore it is advantageous to study these two systems together. It will be seen that algorithms and convergence properties carry over naturally.

There are two reasons for making the above restriction on the iterative methods. First, as has been much emphasized for direct methods, small perturbations in the elements of AA^T or $A^T A$ may perturb the solution to (1.2) or (1.3) much more than perturbations of similar size in A. Perhaps more important, in some applications the matrices AA^T and $A^T A$ will be much less sparse than A itself. Notice that this is in contrast to the dense case, where e.g. $A^T A$ will always have fewer non-zero elements than A if $m \geqq n$ and symmetry is taken into account, and hence the computation of $A^T A$ can be viewed as a data reduction.

The fill-in when forming AA^T or $A^T A$ is in general difficult to predict. The case when A is a rectangular band matrix is a favourable as the dense case, see Björck

[5]. On the other hand, consider the case of a randomly sparse matrix A such that $a_{ij} \neq 0$ with probability $p \leqq 1$. Then, ignoring cancellations, $(AA^T)_{ij} \neq 0$ with probability $q = 1 - (1-p^2)^n$. Thus AA^T will be almost dense already when $p \approx n^{-\frac{1}{2}}$, i.e. when the average number of non-zeroes in a row is $\approx n^{\frac{1}{2}}$. It is interesting to observe that systems where the matrix A has approximately $n^{\frac{1}{2}}$ non-zeroes in each row occur naturally, e.g. in some methods for the reconstruction of two-dimensional images from projections [16].

In this paper we will primarily study the implementation and computational complexity of different methods. Convergence proofs and generalizations to group-iterative methods are given in Elfving [11].

In section 2 the general concept of preconditioning of the system (1.1) with respect to the minimum norm problem (1.2) and the least squares problem (1.3) is introduced and discussed. In section 3 we derive a relation between the Gauss–Seidel method for (1.2) and (1.3) and two classes of projection methods introduced by Householder and Bauer. This provides efficient implementations of the *SOR*-methods, which only require the original matrix A. In section 4 the same idea is used to implement the *SSOR* semi-iterative methods. Section 5 shows how the generalized conjugate gradient method based on *SSOR*-preconditioning can be implemented. Section 6 discusses briefly two step algorithms for the general rank deficient case. Finally, in section 7 some applications and numerical results are given.

2. Preconditioning.

In this section we briefly discuss the use of preconditioning to improve the rate of convergence of iterative methods for (1.2) and (1.3). Let the *non-zero* singular values of A be $\sigma_1(A) \geqq \sigma_2(A) \geqq \ldots \geqq \sigma_r(A)$, $r \leqq \min(m, n)$, and define the condition number of A to be

$$\varkappa(A) = \sigma_1(A)/\sigma_r(A) .$$

The asymptotic rate of convergence of iterative methods is in general proportional to $\varkappa^{-p}(A)$ for some $p > 0$. By preconditioning we mean a transformation of the original system (1.1), which decreases the relevant condition number. Obviously (1.1) can be transformed in two ways:

$$AC^{-1}\bar{x} = b, \quad x = C^{-1}\bar{x} , \tag{2.1}$$

$$C^{-1}Ax = C^{-1}b . \tag{2.2}$$

Here (2.1) corresponds to a transformation of the unknowns and (2.2) to a transformation of the residuals. This is equivalent to a preconditioning of A from the right or from the left. The matrix C should be chosen so that $\varkappa(AC^{-1})$ and $\varkappa(C^{-1}A)$ become smaller than $\varkappa(A)$.

Note that for the transformation (2.1) we have in general $\|x\|_2 \neq \|\bar{x}\|_2$, and we cannot expect an iterative method applied to (2.1) to converge to a minimum

norm solution of (1.1). Similarly, when solving the transformed system (2.2), we will not in general get a least squares solution of (1.1). This is why a two step procedure may be preferable for computing A^+b in the general case.

The simplest case of preconditioning is when C is chosen to be a diagonal matrix D. Then the transformations in (2.1) and (2.2) obviously correspond to a scaling of the columns respectively the rows of A. It has been shown by van der Sluis [34] that if A^TA has at most q non-zero elements in any row, then if the columns of A are scaled to have equal Euclidean norm $\varkappa(AD^{-1})$ is not more than a factor $q^{\frac{1}{2}}$ larger than the minimum under all column scalings of A. If this theorem is applied to AA^T a similar conclusion follows for row scalings of A. Therefore it is in general recommended to equilibrate the rows or columns before applying an iterative method to solve (1.2) or (1.3) respectively.

Some possible choices of preconditioning matrices for the least squares problem are given in [5]. These require in general some initial transformation of A which usually partly destroys sparsity. We also remark that many of the preconditioning techniques for symmetric positive definite problems rely for their success on some special structure of the matrix. The *SSOR*-preconditioning discussed later in section 5 can be efficiently implemented for general sparse matrices A.

3. Projection methods and *SOR*.

Householder and Bauer [19] have studied two classes of projection methods for square and non-singular systems. In this section we formally generalize these methods to rectangular systems and show that as special cases we get the Gauss–Seidel methods for (1.2) and (1.3). The *SOR*-methods then follow easily.

First let $q_i \notin N(A^T)$, $i=1,2,\ldots$ be a sequence of non-zero m-vectors and compute a sequence of approximations from

$$x_{i+1} = x_i+\alpha_iA^Tq_i, \quad \alpha_i = q_i^T(b-Ax_i)/\|A^Tq_i\|_2^2 . \tag{3.1}$$

If the system $Ax=b$ is consistent then $AA^+b=b$ and it follows that $d_{i+1}^TA^Tq_i=0$, where $d_i=A^+b-x_i$. Therefore

$$\|d_{i+1}\|_2^2 = \|d_i\|_2^2-|\alpha_i|^2\|A^Tq_i\|_2^2 \leqq \|d_i\|_2^2 ,$$

i.e. for consistent systems the class of methods (3.1) is error reducing.

For the second class let $p_j \notin N(A)$, $j=1,2,\ldots$ be a sequence of non-zero n-vectors and compute a sequence of approximations from

$$x_{j+1} = x_j+\alpha_jp_j, \quad \alpha_j = p_j^TA^T(b-Ax_j)/\|Ap_j\|_2^2 . \tag{3.2}$$

It follows immediately that $r_{j+1}^TAp_j=0$, where $r_j=b-Ax_j$. Therefore

$$\|r_{j+1}\|_2^2 = \|r_j\|_2^2-|\alpha_j|^2\|Ap_j\|_2^2 \leqq \|r_j\|_2^2 ,$$

i.e. the class of methods (3.2) is residual reducing.

Now assume that A has non-zero rows and let q_i in (3.1) equal the unit vectors e_i taken in cyclic order. We then have

$$A^T q_i = A^T e_i = a_{i.} ,$$

where $a_{i.}^T$ is the ith row vector of A. Then (3.1) becomes

$$x_{i+1} = x_i + a_{i.}(b_i - a_{i.}^T x_i)/\|a_{i.}\|_2^2, \quad i = 1,2,\ldots,m . \tag{3.3}$$

This method was originally devised by Kaczmarz [20], who considered only the case of a square matrix A. Its convergence properties have recently been studied for general $m \times n$ matrices by Tanabe [33]. Now consider Gauss–Seidel's method for (1.2). There, in the ith minor step, y_i is updated by

$$\Delta y_i = e_i(b_i - a_{i.}^T A^T y_i)/a_{i.}^T a_{i.} ,$$

and with $x_i = A^T y_i$ and $x_{i+1} = x_i + A^T \Delta y_i$ we recover (3.3). Therefore, if we take $x_0 = A^T y_0$ for an arbitrary y_0, then (3.3) is equivalent to Gauss–Seidel's method for (1.2). It now follows immediately that one step of the *SOR*-method applied to (1.2) can be written

$$x_{i+1} = x_i + \omega a_{i.}(b_i - a_{i.}^T x_i)/\|a_{i.}\|_2^2, \quad i = 1,2,\ldots,m \tag{3.4}$$

i.e. by introducing the acceleration parameter ω, $0 < \omega < 2$, in Kaczmarz' method. We remark that the method (3.4) is invariant under row-scalings of (A, b) and permutation of the unknowns. However, the rate of convergence of (3.4) does depend on the ordering of the equations in $Ax = b$.

We next assume that A has non-zero columns and let p_j in (3.2) equal the unit vectors e_j taken in cyclic order. Then

$$Ap_j = Ae_j = a_{.j} ,$$

where $a_{.j}$ is the jth column of the matrix A and (3.2) becomes

$$x_{j+1} = x_j + e_j a_{.j}^T (b - Ax_j)/\|a_{.j}\|_2^2, \quad j = 1,2,\ldots,n . \tag{3.5}$$

Note that in the jth step only the jth component of x_j is changed. For a square matrix A this method was developed by de la Garza [14]. However, (3.5) is easily shown to be equivalent to the Gauss–Seidel method applied to (1.3). As before, by including an acceleration parameter ω we get the *SOR*-method for (1.3)

$$x_{j+1} = x_j + \omega e_j a_{.j}^T (b - Ax_j)/\|a_{.j}\|_2^2, \quad j = 1,2,\ldots,n . \tag{3.6}$$

The iteration method (3.6) is invariant under column scalings of A and permutation of the equations. Its rate of convergence depends on the ordering of the unknowns.

We stress that (3.4) and (3.6) are very efficient ways to implement the *SOR*-methods for (1.2) and (1.3) when A is a sparse matrix. Let $nz(A)$ be the number of non-zero elements of A and consider (3.4). There we first have to compute the ith component of the residual vector $(b_i - a_{i.}^T x_i)$. This only requires the scalar product

of x_i with the sparse row vector $a_{i.}$. Next x_i is updated by a multiple of the same vector $a_{i.}$. Thus, a complete cycle, $i=1,2,\ldots,m$, requires only about $2nz(A)$ multiplications (the row norms $\|a_{i.}\|_2^2$ can be computed once and for all). Only one sweep sequentially through the *rows* of A from top to bottom is needed. The vector b is also row-wise accessed and only the components of x are updated in non-sequential order. No extra storage is required.

Similarly, for (3.6) only one sweep through the *columns* of A from left to right is needed. Rewriting (3.6) as

$$x_{j+1} = x_j+\omega e_j\delta_j, \quad r_{j+1} = r_j-\omega a_{.j}\delta_j \quad \delta_j = a_{.j}^T r_j/\|a_{.j}\|_2^2 , \tag{3.7}$$

we see that the residual vector r_j can be computed recursively. Then again about $2nz(A)$ multiplications are needed for a complete cycle $j=1,2,\ldots,n$ and only the components of r need to be accessed in non-sequential order.

We finally remark that Kaczmarz's method (3.3) has been rediscovered and used successfully in picture reconstruction [16]. In this context the method is known as the unconstrained *ART*-algorithm (*A*lgebraic *R*econstruction *T*echnique).

4. *SSOR*-semi-iterative methods.

The *SOR*-methods described in section 3 have the advantage of simplicity and small storage requirements. However, when A^TA and AA^T do not have Young's "property A" the rate of convergence may be very slow for any choice of ω. In this section we therefore consider semi-iterative methods based on the *SSOR*-method.

If a sweep forward in (3.4) is followed by a sweep backward i.e.

$$x_{i+1} = x_i+\omega a_{j.}(b_i-a_{j.}^T x_i)/\|a_{j.}\|_2^2, \quad i=1,2,\ldots,2m, \; j=\min(i,2m+1-i) , \tag{4.1}$$

then we obtain the *SSOR*-method for (1.2). In a complete double sweep the unknowns are then transformed

$$x^{(k+1)} = Q_{SSOR}x^{(k)}+R_{SSOR}b , \tag{4.2}$$

where

$$Q_{SSOR} = Q_1Q_2\cdots Q_mQ_m\cdots Q_2Q_1, \quad Q_i = I-\omega a_{i.}a_{i.}^T/\|a_{i.}\|_2^2 . \tag{4.3}$$

The matrices Q_i are symmetric and it follows that the iteration matrix Q_{SSOR} in (4.2) is symmetric.

Similarly the *SSOR*-method for (1.3) can be written

$$x_{j+1} = x_j+\omega e_i a_{.i}^T(b-Ax_j)/\|a_{.i}\|_2^2, \quad j=1,2,\ldots,2n, \; i=\min(j,2n+1-j) . \tag{4.4}$$

In a double sweep $x^{(k)}$ and the corresponding residual vector $r^{(k)}$ are now transformed

$$x^{(k+1)} = \bar{Q}_{SSOR}x^{(k)}+\bar{R}_{SSOR}b, \quad r^{(k+1)} = \bar{P}_{SSOR}r^{(k)} . \tag{4.5}$$

Here

$$\bar{P}_{SSOR} = P_1P_2 \dots P_nP_n \dots P_2P_1, \quad P_j = I - \omega a_{\cdot j}a_{\cdot j}^T/\|a_{\cdot j}\|_2^2 , \tag{4.6}$$

and it follows that $\bar{P}_{SSOR}$ is a symmetric matrix. There is no simple expression for $\bar{Q}_{SSOR}$ in terms of the P_j.

Although $\bar{Q}_{SSOR}$ is not symmetric it follows from the general *SSOR*-theory, Young [37] p. 461, that $\bar{Q}_{SSOR}$ is similar to a symmetric matrix. Chebyshev semi-iteration can therefore be used to accelerate both (4.2) and (4.5).

The *SSOR* semi-iterative method for the minimum norm problem (1.2), where $b \in R(A)$, is obtained by applying Chebyshev semi-iteration to the linear system related to (4.2)

$$(I - Q_{SSOR})x = R_{SSOR}b . \tag{4.7}$$

We note from (4.3) that for any vector $x \in N(A)$ we have $(I - Q_{SSOR})x = 0$. Therefore, if $n > m$, then $(I - Q_{SSOR})$ has at least $n - m$ zero eigenvalues and is only positive semidefinite. However, from the general *SSOR*-theory it is known that (4.7) is a consistent system and as remarked in [36] Chebyshev semi-iteration applies also in this case. If we denote the accelerated sequence of approximations for (4.2) by $(\tilde{x}_k)$, then we have

$$\tilde{x}^{(k+1)} = \tilde{x}^{(k)} + \{p_{k-1}(\tilde{x}^{(k)} - \tilde{x}^{(k-1)}) + (z^{(k)} - \tilde{x}^{(k)})\}/q_k , \tag{4.8}$$

$$z^{(k)} = Q_{SSOR}\tilde{x}^{(k)} + R_{SSOR}b . \tag{4.9}$$

Formulas for computing p_k and q_k from lower and upper bounds a and b for the non-zero eigenvalues of $(I - Q_{SSOR})$ are given in [36]. In (4.9) $z^{(k)}$ is the result of performing one *SSOR*-step from the approximation $\tilde{x}^{(k)}$. Thus, as for *SSOR*, only two sweeps through the matrix A are required for each step of (4.9), but there is a storage overhead of two n-vectors $\tilde{x}^{(k)}$ and $(\tilde{x}^{(k)} - \tilde{x}^{(k-1)})$.

The *SSOR* semi-iterative method for the least squares problem (1.3) can be developed analogously by applying Chebyshev semi-iteration to the linear system related to (4.5)

$$(I - \bar{Q}_{SSOR})x = \bar{R}_{SSOR}b . \tag{4.10}$$

There is one complication, since in (4.4) the residuals $(b - Ax_j)$ should be computed recursively as in (3.7). To start this recursion we have to compute $(b - A\tilde{x}^{(k)})$ at the beginning of each *SSOR*-step, which requires an additional sweep through the matrix A and $nz(A)$ additional multiplications. An alternative way of implementing this semi-iterative method, which avoids this overhead, will be described in section 5.

A drawback with Chebyshev semi-iteration is that often it is very difficult to find a suitable lower bound for the eigenvalues of the iteration matrix. We remark that it is possible to apply this method with any choice of $a > 0$. Taking $a > \lambda_{\min}$ will in general suppress that part of the solution corresponding to eigenvalues

smaller than a. This kind of regularization of the solution may be desirable in some problems.

5. *SSOR*-preconditioned conjugate gradient methods.

Instead of Chebyshev semi-iteration it is possible to use the conjugate gradient method (cg-method) as an acceleration technique. The cg-method has a number of attractive properties when used for this purpose, see e.g. Concus et al. [8]. Perhaps most important is that it does not require an estimation of parameters and that it takes advantage of the distribution of the eigenvalues of the iteration operator.

In our applications the cg-method will be used to compute the minimum norm solution $x=B^+c$ of a consistent system $Bx=c$, where B is symmetric and semidefinite. We state below some important properties of the cg-method applied to such systems.

LEMMA 5.1. *Let $Bx=c$, where B is symmetric and positive semidefinite and where $c \in R(B)$. Let $x_0=x_0'+x_0''$ with $x_0' \in R(B)$ and $x_0'' \in N(B)$, $p_0=s_0=c-Bx_0$ and compute for $k=0,1,2,\ldots$*

$$
\begin{aligned}
q_k &= Bp_k, \quad \alpha_k = \|s_k\|_2^2/p_k^T q_k , \\
x_{k+1} &= x_k+\alpha_k p_k, \quad s_{k+1} = s_k-\alpha_k q_k , \\
\beta_k &= \|s_{k+1}\|_2^2/\|s_k\|_2^2, \quad p_{k+1} = s_{k+1}+\beta_k p_k .
\end{aligned} \tag{5.1}
$$

Then the sequence of approximations $\{x_k\}$ has the following properties:

(i) *The vector x_k minimizes the quadratic form*

$$Q(x) = (B^+c-x)^T B(B^+c-x)$$

among all vectors of the form $x=x_0+w$, where w lies in the space

$$S_k = \{s_0, Bs_0, \ldots, B^{k-1}s_0\} .$$

(ii) *The algorithm terminates with $x_k=B^+c+x_0''$ after $t \leqq \operatorname{rank}(B)$ steps, and for $k \leqq t$ we have*

$$Q(x_k) < Q(x_{k-1}), \quad \|B^+c-x_k\|_2 < \|B^+c-x_{k-1}\|_2 .$$

PROOF. The above properties are well known in case B is positive definite (see e.g. Hestenes and Stiefel [18]). In the semidefinite case $c \in R(B)$ implies that also p_0 and s_0 lie in $R(B)$. But then it follows from the formulas (5.1) by recursion that x_k-x_0'', p_k and s_k lie in $R(B)$ for all k. Therefore $\{x_k-x_0''\}$ is the same sequence as that generated by the cg-method for the positive definite system $\tilde{B}x=c$, where $\tilde{B}$ is the restriction of B to $R(B)$, and the lemma follows. (For a slightly different proof see Elfving [10]). ∎

To accelerate the *SSOR*-method for the minimum norm problem (1.2) we apply the cg-method to the symmetric, consistent linear system (4.7). Using $R(I-Q_{SSOR})=R(A^T)$ which follows from (4.1) we obtain:

Algorithm CGMN: Take

$$x_0 \in R(A^T),\ s_0 = p_0 = (Q_{SSOR}x_0 + R_{SSOR}b) - x_0 ,$$

and for $k=0,1,2,\ldots$ compute x_{k+1}, s_{k+1} and p_{k+1} from (5.1), where

$$q_k = p_k - Q_{SSOR}p_k . \tag{5.2}$$

To compute p_0 in (5.2) we perform one *SSOR*-step (4.1), and to compute q_k one step (4.1) is made with $x_1 = p_k$ and now with $b=0$. As in *SSOR* only two sweeps through the rows of A are needed for one iteration step. If A is only moderately sparse the computational overhead of cg-acceleration is small, but we now need storage for three n-vectors, s, p and q in addition to x. However, only the components of q are accessed in non-sequential order and x, s and p may be held in backing storage. For a possible way to save storage, see [1].

We note that since the system (4.10) is not symmetric, no dual method to *CGMN* exists for the least squares problem (1.3). The natural dual method would be an iterative method for the residual vector $r=b-Ax$ of (1.3), cf. (4.5)–(4.6).

If $m<n$, then as remarked in section 4, the matrix $(I-Q_{SSOR})$ is always rank deficient. We now derive a different *SSOR*-cg method for the minimum norm problem, which is more stable and requires slightly less storage. This method will also have a natural dual for the least squares problem.

We start by giving an explicit expression for the iteration matrix in (4.2) using classical *SSOR*-theory. First let $AA^T = L+D+L^T$, where L is strictly lower triangular, D diagonal and

$$(L)_{ij} = a_{i.}^T a_{j.},\quad i > j,\quad d_{ii} = \|a_{i.}\|_2^2 . \tag{5.3}$$

Then it is well known (Young [37], p. 462) that the iteration matrix for the *SSOR*-method applied to $AA^Ty=b$ is

$$Q_\omega = I - \omega(2-\omega)(D+\omega L)^{-T}D(D+\omega L)^{-1}AA^T . \tag{5.4}$$

Thus, after dividing out the scaling constant $\omega(2-\omega)$, the related linear system can be written

$$C_\omega^{-T}C_\omega^{-1}AA^Ty = C_\omega^{-T}C_\omega^{-1}b,\quad C_\omega = (D+\omega L)D^{-\frac{1}{2}} . \tag{5.5}$$

Thus $z=C_\omega^T y$ satisfies the symmetric system

$$C_\omega^{-1}AA^TC_\omega^{-T}z = C_\omega^{-1}b,\quad x = A^TC_\omega^{-T}z . \tag{5.6}$$

We remark that (5.6) corresponds to a preconditioning of A from the left with the matrix C_ω^{-1}, cf. Axelsson [3]. We now apply the cg-method to solve (5.6). Transforming back to the original variables x, the algorithm can be written as follows:

Algorithm CGPCMN: Take

$$x_0 \in R(A^T), \quad p_0 = r_0 = C_\omega^{-1}(b - Ax_0),$$

and for $k = 0, 1, 2, \ldots$ compute

$$\begin{aligned} q_k &= A^T C_\omega^{-T} p_k, \quad \alpha_k = \|r_k\|_2^2 / \|q_k\|_2^2, \\ x_{k+1} &= x_k + \alpha_k q_k, \quad r_{k+1} = r_k - \alpha_k C_\omega^{-1} A q_k, \\ \beta_k &= \|r_{k+1}\|_2^2 / \|r_k\|_2^2, \quad p_{k+1} = r_{k+1} + \beta_k p_k. \end{aligned} \tag{5.7}$$

This algorithm needs storage for two n-vectors x and q and two m-vectors p and r. As in *CGMN* only the components of q need to be accessed in non-sequential order.

Since the elements in L defined by (5.3) are not explicitly known, it remains to be shown that the vectors $A^T C_\omega^{-T} p_k$ and $C_\omega^{-1} A q_k$ in (5.7) can be efficiently computed. Dropping the index k we put

$$q = A^T C_\omega^{-T} p = A^T (D + \omega L^T)^{-1} D^{\frac{1}{2}} p = A^T s.$$

It follows that $Ds = D^{\frac{1}{2}} p - \omega L^T s$, or component-wise

$$d_i s_i = d_i^{\frac{1}{2}} p_i - \omega a_{i.}^T h_i, \qquad h_i = \sum_{j=i+1}^{m} a_{j.} s_j .$$

Thus we can compute $q = A^T C_\omega^{-T} p = h_0$ by the recursion:

Put $h_m = 0$, and for $i = m, m-1, \ldots, 1$ compute

$$s_i = d_i^{-\frac{1}{2}} p_i - \omega d_i^{-1} a_{i.}^T h_i, \qquad h_{i-1} = h_i + a_{i.} s_i . \tag{5.8}$$

Further, from $t = C_\omega^{-1} A q = D^{\frac{1}{2}} (D + \omega L)^{-1} A q$ we get $D^{\frac{1}{2}} t = Aq - \omega L D^{-\frac{1}{2}} t$ or component-wise

$$d_i^{\frac{1}{2}} t_i = a_{i.}^T \left(q - \omega \sum_{j=1}^{i-1} t_j d_j^{-\frac{1}{2}} a_{j.} \right), \qquad i = 1, 2, \ldots, m .$$

Thus, we can compute $t = C_\omega^{-1} A q$ by the recursion:

Put $g_1 = q$, and for $i = 1, 2, \ldots, m$ compute

$$t_i = d_i^{-\frac{1}{2}} a_{i.}^T g_i, \qquad g_{i+1} = g_i - (\omega d_i^{-\frac{1}{2}} t_i) a_{i.} . \tag{5.9}$$

Using (5.8) and (5.9) we can implement a cg-step in (5.7) again with only two sweeps through the rows of A. Note that since we need not store the vectors s and t in (5.8) and (5.9) no extra storage is involved.

As shown in the following theorem the two algorithms *CGMN* and *CGPCMN* are closely related but not equivalent.

THEOREM 5.1. *The algorithms CGMN and CGPCMN both generate approximations* $x_k \in R(A^T)$ *of the form* $x_0 + w_k$, *where* $w_k \in S_k$,

$S_k = \{s_0, Bs_0, \ldots, B^{k-1}s_0\}$ *and*

$$s_0 = A^T C_\omega^{-T} C_\omega^{-1}(b - Ax_0), \quad B = A^T C_\omega^{-T} C_\omega^{-1} A .$$

Furthermore, if $b \in R(A)$, *then the algorithms minimize the quadratic forms*

$Q_{CGMN}(x) = \|C_\omega^{-1}(b - Ax)\|_2^2$ *and* $Q_{CGPCMN}(x) = \|A^+ b - x\|_2^2$, *respectively*.

Also for CGMN the norm $\|A^+ b - x_k\|_2$ *decreases monotonically.*

PROOF. It follows from (5.4) that, after division by $\omega(2-\omega)$, the system (4.7) can be written $Bx = c$, where

$$B = A^T C_\omega^{-T} C_\omega^{-1} A, \quad c = A^T C_\omega^{-T} C_\omega^{-1} b . \tag{5.10}$$

Thus, from lemma 5.1 *CGMN* generates approximations $x_k = x_0 + w_k$, where $w_k \in S_k$ and $s_0 = c - Bx_0 = A^T C_\omega^{-T} C_\omega^{-1}(b - Ax_0)$. Moreover, from lemma 5.1, *CGMN* minimizes the quadratic form $\|C_\omega^{-1} A(B^+ c - x)\|_2^2$. But when $b \in R(A)$ the two systems $C_\omega^{-1} Ax = C_\omega^{-1} b$ and $Ax = b$ have the same set of solutions. Therefore also $B^+ c = A^+ b$ and the properties for *CGMN* now follow from the relation $AA^+ b = b$. For *CGPCMN* we have $x = A^T C_\omega^{-T} z$, where from (5.6) z satisfies $Ez = f$ with $E = C_\omega^{-1} AA^T C_\omega^{-T}$, $f = C_\omega^{-1} b$. Thus from lemma 5.1, $z_k = z_0 + u_k$, $u_k \in \{t_0, Et_0, \ldots, E^{k-1} t_0\}$, where $t_0 = C_\omega^{-1}(b - AA^T C_\omega^{-T} z_0) = C_\omega^{-1}(b - Ax_0)$. But for $p \geqq 0$, $A^T C_\omega^{-T} E^p t_0 = B^p s_0$, so $x_k - x_0 \in S_k$. Again from lemma 5.1 it follows that *CGPCMN* minimizes $\|A^T C_\omega^{-T}(E^+ f - z)\|_2^2 = \|A^T C_\omega^{-T} E^+ f - x\|_2^2$. But $b \in R(A)$ and thus $AA^T y = b$ and $C_\omega^{-1} AA^T y = C_\omega^{-1} b$ have the same set of solutions. Therefore $C_\omega^{-T} E^+ f = (AA^T)^+ b$ and it follows that $A^T C_\omega^{-T} E^+ f = A^T (AA^T)^+ b = A^+ b$, which proves the minimizing property of *CGPCMN*. ∎

We remark that from theorem 1.2 and (5.10) it follows that in the general case of an inconsistent system $Ax = b$ the algorithm *CGMN* solves the problem

$$\min \|x\|_2, \; x \in \{x \; ; \; \|C_\omega^{-1}(b - Ax)\|_2 = \text{minimum}\} .$$

The non-zero eigenvalues of the matrices occurring in (5.6) and (5.10) are identical. Hence we can expect similar convergence behaviour for *CGPCMN* and *CGMN*. This is also confirmed by computational experience, cf. fig. 1. We recommend the use of *CGPCMN* mainly for the following reasons. It requires less storage, if $m < n$, than *CGMN*. It is more stable, cf. the discussion in section 7, and finally it minimizes a more natural error-norm than does *CGMN*.

We now consider the acceleration of the *SSOR*-method for the least squares problem (1.3). We let $A^T A = \bar{L} + \bar{D} + \bar{L}^T$, where $\bar{L}$ is strictly lower triangular, $\bar{D}$ diagonal and

$$(\bar{L})_{ij} = a_{.i}^T a_{.j}, \quad i > j, \quad \bar{d}_{jj} = \|a_{.j}\|_2^2 . \tag{5.11}$$

Then for the matrix in (4.10) we have the expression

$$\bar{Q}_{SSOR} = I - \omega(2-\omega)(\bar{D} + \omega\bar{L})^{-T} \bar{D} (\bar{D} + \omega\bar{L})^{-1} A^T A .$$

We again drop the scaling constant $\omega(2-\omega)$ and make the change of variables

$$z = \bar{C}_\omega^T x, \quad \bar{C}_\omega = (\bar{D}+\omega\bar{L})\bar{D}^{-\frac{1}{2}}. \tag{5.12}$$

Then the system (4.11) can be written

$$\bar{C}_\omega^{-1}A^TA\bar{C}_\omega^{-T}z = \bar{C}_\omega^{-1}A^Tb, \quad x = \bar{C}_\omega^{-T}z. \tag{5.13}$$

If we now apply the cg-method to (5.13) we get the dual algorithm to *CGPCMN*. The resulting algorithm can be written in the following form:

Algorithm CGPCNE: Take

$$x_0 \in R(A^T), \quad r_0 = b-Ax_0, \quad p_0 = s_0 = \bar{C}_\omega^{-1}A^Tr_0,$$

and for $k=0,1,2,\ldots$ compute

$$\begin{aligned} q_k &= A\bar{C}_\omega^{-T}p_k, \quad \alpha_k = \|s_k\|_2^2/\|q_k\|_2^2 \\ x_{k+1} &= x_k+\alpha_k\bar{C}_\omega^{-T}p_k, \quad r_{k+1} = r_k-\alpha_kq_k, \\ s_{k+1} &= \bar{C}_\omega^{-1}A^Tr_{k+1}, \quad \beta_k = \|s_{k+1}\|_2^2/\|s_k\|_2^2, \quad p_{k+1} = s_{k+1}+\beta_kp_k. \end{aligned} \tag{5.14}$$

The quantities $\bar{C}_\omega^{-T}p_k$, $A\bar{C}_\omega^{-T}p_k$ and $\bar{C}_\omega^{-1}A^Tr_k$ in (5.14) can be computed efficiently with only two sweeps through the columns of the matrix A. The algorithms for this are derived exactly analogous to (5.8) and (5.9). Dropping the index k we let $t=\bar{C}_\omega^{-T}p$ and using (5.12) get the recursion:

Put $h_n=0$, and for $j=n,n-1,\ldots,1$ compute

$$t_j = \bar{d}_j^{-\frac{1}{2}}p_j-\omega\bar{d}_j^{-1}a_{\cdot j}^Th_j, \quad h_{j-1} = h_j+a_{\cdot j}t_j. \tag{5.15}$$

Note that $q=A\bar{C}_\omega^{-T}p=h_0$.

Further the vector $s=\bar{C}_\omega^{-1}A^Tr$ is computed by the recursion:

Put $h_1=r$ and for $j=1,2,\ldots,n$ compute

$$s_j = \bar{d}_j^{-\frac{1}{2}}a_{\cdot j}^Th_j, \quad h_{j+1} = h_j-(\omega\bar{d}_j^{-\frac{1}{2}}s_j)a_{\cdot j}. \tag{5.16}$$

The algorithm requires storage for two m-vectors r, q and three n-vectors x, p and s. Only one m-vector needs to be accessed in non-sequential order.

We first remark that from theorem 1.2 and (5.13) it follows that the algorithm *CGPCNE*, provided $x_0=\bar{C}_\omega^{-T}z_0$ where $z_0 \in R(\bar{C}_\omega^{-1}A^T)$, solves the problem

$$\min \|\bar{C}_\omega^Tx\|_2, \quad x \in \{x\ ;\ \|b-Ax\|_2=\text{minimum}\}.$$

We state some further properties of this algorithm in the following theorem.

THEOREM 5.2. *The algorithm CGPCNE generates approximates* $x_k=x_0+C_\omega^{-T}w_k$, *where* $x_0 \in R(A^T)$, $w_k \in \{s_0, Bs_0, \ldots, B^{k-1}s_0\}$ *and*

$$s_0 = \bar{C}_\omega^{-1}A^T(b-Ax_0), \quad B = \bar{C}_\omega^{-1}A^TA\bar{C}_\omega^{-T}.$$

Furthermore, CGPCNE minimizes in each step the quadratic form

$$Q_{CGPCNE}(x) = \|b-Ax\|_2^2 .$$

PROOF. The first part of the theorem is an immediate consequence of lemma 5.1. From the same lemma follows that with $c=\bar{C}_\omega^{-1}A^Tb$, *CGPCNE* minimizes

$$\|A\bar{C}_\omega^{-T}(z-B^+c)\|_2^2 = \|A(x-x_{LS})\|_2^2 ,$$

where $x_{LS}=\bar{C}_\omega^{-T}B^+c$ is a least squares solution of $Ax=b$. Since $b-Ax_{LS}\in N(A^T)$ we have $\|b-Ax\|_2^2=\|b-Ax_{LS}\|_2^2+\|A(x-x_{LS})\|_2^2$, and therefore also $\|b-Ax\|_2$ is minimized. ∎

Notice that since we divided out the factor $\omega(2-\omega)$, it is possible to let ω take any value in *CGPCMN* and *CGPCNE*. In particular, if we take $\omega=0$, then it is easily verified that the resulting algorithms are the cg-method applied to the systems (1.2) and (1.3) after row- and column-equilibration respectively of A. In this case approximately half the number of operations in the recursions (5.8), (5.9) respectively (5.15), (5.16) can be saved. However, we still need to access the rows/columns of A twice per iteration. If A is so big that it cannot be kept in internal store it obviously becomes important to restrict the number of accesses. In [10] several mathematically equivalent versions of the *CG*-method for (1.3) and one for (1.2) are given which only access the elements of A once per iteration step.

We end this section by pointing out some generalizations of the derived algorithms. In [11] Elfving has shown that by partitioning A into blocks of rows or columns one can derive block-*SOR* (and *SSOR*) methods for problems (1.2) and (1.3). These block-iterative methods can also be accelerated by Chebyshev semi-iteration or the cg-method. In particular we point out that the algorithms *CGPCMN* and *CGPCNE* given in this section have been formulated so that they easily generalize to block methods.

Using the derived recursions (5.8)–(5.9) and (5.15)–(5.16) it is straightforward to apply Chebyshev semi-iteration to the systems (5.6) and (5.13). We also remark that in *CGPCMN* and *CGPCNE* it is possible to use other precondition matrices than C_ω and $\bar{C}_\omega$. The conclusions in theorems 5.1 and 5.2 still hold if these other precondition matrices are substituted.

6. General pseudoinverse algorithms.

If rank $(A)<\min(m,n)$ and b is not in $R(A)$, then neither of the algorithms developed above will converge to the pseudoinverse solution. However, a two-step procedure based on the given algorithms can easily be derived for the general case. First let

$$b = b'+b'', \quad b'\in R(A), \quad b''\in N(A^T) . \tag{6.1}$$

Then, for any least squares solution x_{LS} of $Ax=b$, we have

$$Ax_{LS} = b', \quad r_{LS} = b - Ax_{LS} = b'' .$$

The pseudoinverse solution A^+b therefore equals the minimum norm solution of the consistent system $Ax=b'$. This corresponds to a splitting of x_{LS},

$$x_{LS} = x' + x'', \quad x' = A^+b \in R(A^T), \quad x'' \in N(A) . \tag{6.2}$$

Thus, the following algorithm can be used to compute A^+b (cf. Plemmons [27]).

Algorithm 6.1.

(i) Compute a least squares solution x_{LS} of $Ax=b$ and the corresponding residual r_{LS}, using e.g. *CGPCNE*. Put $b'=b-r_{LS}$.

(ii) Compute the minimum norm solution of the system $Ax=b'$, using e.g. *CGPCMN*. This solution equals A^+b.

We remark that in step (i) it is not necessary to compute x_{LS}, since only the residual r_{LS} is needed to start step (ii). Note that this means that in the *SOR*-method (3.7), we do not need to iterate for x. A similar simplification applies if the *SSOR*-method is used for step (i). Also, it is possible to accelerate the *SSOR*-method (4.5) $r^{(k+1)} = \bar{P}_{SSOR} r^{(k)}$, for r, by Chebyshev semi-iteration or the cg-method, since, as remarked in section 4, $\bar{P}_{SSOR}$ is a symmetric matrix.

If we actually do compute a solution x_{LS} in step (i), then another possible algorithm is the following.

Algorithm 6.2.

(i) Compute a least squares solution x_{LS} of $Ax=b$ using e.g. *CGPCNE*.

(ii) Use one of the given minimum norm algorithms, e.g. *CGPCMN*, to solve the consistent system $Ax=0$, using the starting approximation $x_0=x_{LS}$. The computed solution is then $x'' \in N(A)$, where x'' is defined by (6.2). We therefore have $A^+b=x_{LS}-x''$.

Algorithm 6.2 is based on the fact that if a starting approximation $x_0=x'+x''$, $x' \in R(A^T)$, $x'' \in N(A)$, is used in one of the given minimum norm algorithms, then the component x'' is left unchanged and does not affect the iterations. This is easily seen to be true for the *SOR*-method (3.4) and the *SSOR*-method (4.1). Indeed, if $x_i=x_i'+x_i''$, then we have $a_i^T x_i = a_i^T x_i'$. The same observation for Kaczmarz's method has been made by Tanabe in [33]. It easily follows from theorem 5.1 that the conclusion also holds for the accelerated versions of the *SSOR*-method.

7. Applications and computational experience.

Areas where applications for our methods may arise include datafitting, surface approximations, geodesy, photogrammetry, image reconstruction and convex programming.

The solution of geodetic normal equations by the *SOR*-method has been considered by Ashkenazi [2], and the solution of similar problems by the cg-method is discussed in [30] and [7]. Whitney and Meany [35] discuss applications of Kaczmarz's method in circuit theory. As remarked earlier this method is also used for reconstruction of two-dimensional pictures from their one-dimensional projections [16]. Consistent systems of similar structure as (1.2) and (1.3) also arise when solving the maximum entropy problem with linear constraints [12].

We now discuss results from some numerical experiments with the methods described in section 5. These were carried out on a DEC-10 with a relative precision of $\varepsilon = 0.4 \cdot 10^{-8}$. For *CGPCMN* and *CGPCNE* the Fortran subroutines given in Report LiTH-MAT-R-1978-5, Linköping University, were used. These implementations differ slightly from the formulas given in the text. To avoid the square roots in (5.15) and (5.16) $\bar{D}^{-\frac{1}{2}}s$ and $\bar{D}^{-\frac{1}{2}}p$ are computed instead of s and p. The square roots occuring in *CGPCMN* are eliminated in a similar way. Further, the residuals in *CGPCMN* are computed using the formula $\tilde{r}_k = C_\omega^{-1}(b - Ax_k)$ instead of the recursion (5.7). Note that this does not increase the number of matrix by vector products needed per iteration.

Our first test matrices come from a modification of the unweighted picture reconstruction model [16], which for the case of three equally spaced projections compensates for the lack of weights. We consider the modified model with 600 meshpoints and projection data corresponding to 60 equations. In this model there are always two redundant equations. A full rank matrix can be obtained by deleting the first and the last row.

For the given matrix A we have generated a random vector y, then computed $x = A^T y$ and finally $b = Ax$. The starting vector $x_0 = 0$ was used. In fig. 1 the norm of

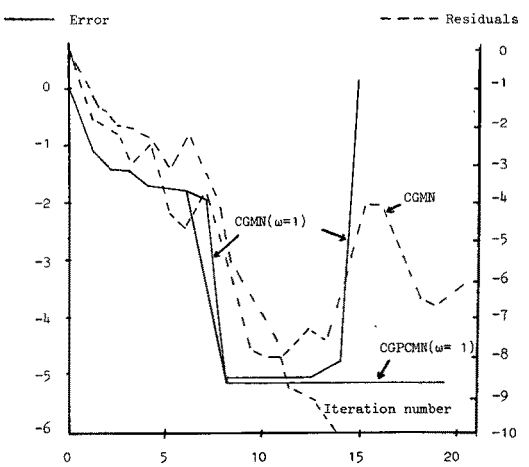


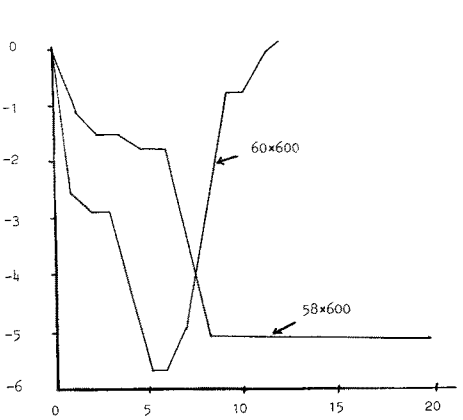
Fig. 1. Logarithms of relative error, $\|x_k - A^+b\|_2/\|A^+b\|_2$ and residuals, $\|r_k\|_2/\|r_0\|_2$ (*CGPCMN*) and $\|s_k\|_2/\|s_0\|_2$ (*CGMN*) when solving a full rank problem of order 58×600.

the error and residual vectors are given for *CGMN* and *CGPCMN* and $\omega = 1$. The figure illustrates a typical instability of *CGMN* if too many iterations are carried out when rank $(A) < m$. Except for this the two methods perform roughly similarly, as was to be expected.

In fig. 2 we compare the behaviour of the relative error for *CGPCMN* with $\omega = 1$, for the singular and non-singular problem. The figure illustrates that when AA^T is rank-deficient this algorithm exhibits the same kind of instability as observed for *CGMN*. It is also seen that the rate of convergence may be worsened by deleting redundant rows, cf. the remark in [28], page 74.

A set of different values of ω was tested in *CGPCMN* when solving the same two problems as above. Fig. 3 shows the necessary number of iterations required to satisfy the stop-criterion

$$\|r_k\|_2 \leqq 10^3 \cdot \varepsilon \|r_0\|_2, \qquad r_k = C_\omega^{-1}(b - Ax_0) .$$



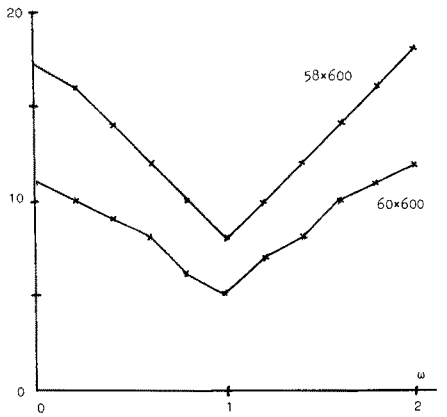


Fig. 2. Behaviour of $\log \|x_k - A^+ b\|_2 / \|A^+ b\|_2$ for method *CGPCMN*, using $\omega = 1$, when solving (i) a full rank problem of order 58×600, (ii) a rank deficient problem of order 60×600.

Fig. 3. Necessary number of iterations, for method *CGPCMN*, versus the iteration parameter, ω, when solving (i) problem of order 58×600, (ii) problem of order 60×600.

Note that the achieved gain in rate of convergence for $\omega = 1$ is outweighed by the fact that roughly half the number of operations per iteration can be saved when $\omega = 0$.

To test the algorithm *CGPCNE* we used the problem of least squares surface approximations by bicubic splines. The function

$$g(x, y) = \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} B_i(x) B_j(y) ,$$

where $B_i(x)$ and $B_j(y)$ are B-splines with knots on a uniform grid over the unit square, was fitted to data points (x_i, y_i, z_i), $i = 1, 2, \ldots, m$. The coordinates x_i and y_i

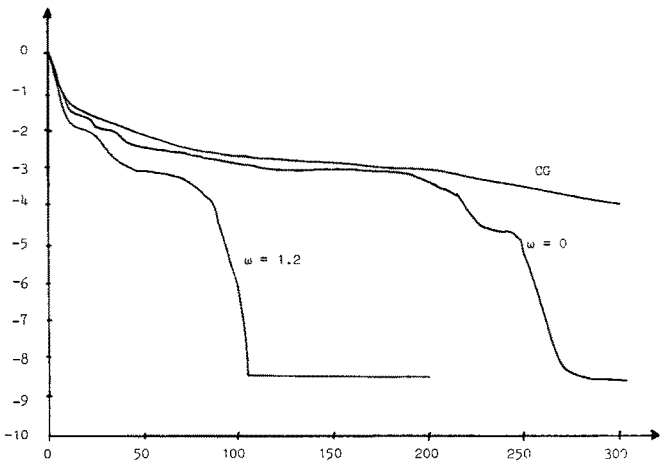


Fig. 4. Behaviour of $\log \|r_k\|_2/\|r_0\|_2$ when solving a consistent surface fitting problem of order 200×100 using *CGPCNE* with $\omega = 0$ and $\omega = 1.2$ and the *CG*-method.

were randomly distributed and z_i chosen to yield a consistent overdetermined system of order $m \times N^2$ for the unknown coefficients c_{ij}. In fig. 4 the error is shown for the cg-method and for *CGPCNE* with $\omega = 0$ and the experimentally found optimal value $\omega = 1.2$.

For the problems we have tested so far, the rate of convergence has varied relatively slowly with ω around an optimal value not far from $\omega = 1$. However, there are certain classes of diagonal-dominant matrices for which the rate of convergence for the optimal value of ω in the *SSOR*-cg method is known to be an order of magnitude better than for $\omega = 1$, see Axelsson [4]. Further investigation in the choice of ω for different applications of our methods is obviously needed.

Acknowledgements.

We wish to thank Per Tengdahl for performing the computations for the least squares surface approximations.

REFERENCES

1. R. S. Anderssen and G. H. Golub, *Richardson's non-stationary matrix iterative procedure*, Report STAN-CS-72-304, Stanford (1972).
2. V. Ashkenazi, *Geodetic normal equations*, 57–74, in *Large Sparse Sets of Linear Equations*, ed. J. K. Reid, Academic Press, New York (1971).
3. O. Axelsson, *On preconditioning and convergence accelerations in sparse matrix problems*, CERN 74-10, Geneva (1974).
4. O. Axelsson, *Solution of linear systems of equations: iterative methods*, in *Sparse Matrix Techniques*, ed. V. A. Barker, Lectures Notes in Mathematics 572, Springer-Verlag (1977).
5. Å. Björck, *Methods for sparse linear least squares problems*, in *Sparse Matrix Computations*, eds. J. R. Bunch and D. J. Rose, Academic Press, New York (1976).

6. Y. T. Chen, *Iterative methods for linear least squares problems*, Ph. D. dissertation, Dep. Comput. Sci., Waterloo, Report CS-75-04 (1975).
7. R. J. Clasen, *A note on the use of the conjugate gradient method in the solution of a large system of sparse equations*, Computer J. 20 (1977), 185–186.
8. P. Concus, G. H. Colub and D. O'Leary, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, Report STAN-CS-75-535, Stanford (1975).
9. J. Dyer, *Acceleration of the convergence of the Kaczmarz method and iterated homogeneous transformation*, Ph.D. thesis, UCLA, Los Angeles (1965).
10. T. Elfving, *On the conjugate gradient method for solving linear least squares problems*, Report LiTH-MAT-R-1978-3, Linköping (1978).
11. T. Elfving, *Group-iterative methods for consistent and inconsistent linear equations*, Report LiTH-MAT-R-1977-11, Revised 1978-02-10, Linköping (1978).
12. S. Erlander, *Entropy in linear programs — an approach to planning*, Report LiTH-MAT-R-1977-3, Linköping (1977).
13. V. Friedrich, *Zur iterativen Behandlung unterbestimmter und nichtkorrekter linearen Aufgaben*, Beiträge zur Numerischen Mathematik, 3 11–20, Oldenburg Verlag, München - Wien (1975).
14. A. de la Garza, *An iterative method for solving systems of linear equations*, Union Carbide, Oak Ridge, Report K-731, Tennessee (1951).
15. T. Ginsburg, *The conjugate gradient method*, in *Handbook for Automatic Computation* Vol. II, *Linear Algebra*, eds. J. H. Wilkinson and C. Reinsch, Springer-Verlag (1971).
16. G. T. Herman, A. Lent and S. W. Rowland, *ART: Mathematics and Applications*, J. Theor, Biol. 42 (1973), 1–32.
17. M. R. Hestenes, *Pseudoinverses and conjugate gradients*, Comm. of the ACM 18 (1975), 40–43.
18. M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standards, Sect. B 49 (1952), 409–436.
19. A. S. Householder and F. L. Bauer, *On certain iterative methods for solving linear systems*, Numer. Math. 2 (1960), 55–59.
20. S. Kaczmarz, *Angenäherte Auflösung von Systemen linearer Gleichungen*, Bull. Internat. Acad. Polon. Sciences et Lettres (1937), 355–357.
21. W. J. Kammerer and M. Z. Nashed, *On the convergence of the conjugate gradient method for singular linear operator equations*, SIAM J. Numer. Anal. 9 (1972), 165–181.
22. H. B. Keller, *On the solution of singular and semidefinite linear systems by iteration*, J. SIAM Numer. Anal. 2 (1965), 281–290.
23. C. Lanczos, *Solution of linear equations by minimized iteration*, Journal of Research of the National Bureau of Standards 49 Sect. B, (1952), 33–53.
24. P. Läuchli, *Iterative Lösung und Fehlerabschätzung in der Ausgleichsrechnung*, Z. für angew. Math. und Physik 10 (1959), 245–280.
25. C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal. 12 (1975), 617–629.
26. W. Peters, *Lösung linearer Gleichungssysteme durch Projektion auf Schnitträume von Hyperebenen und Berechnung einer verallgemeinerten Inversen*, Beiträge zur Numerischen Mathematik 5 (1976), 129–146.
27. R. J. Plemmons, *Stationary iterative methods for linear systems with non-Hermitian singular matrices*, Report from Dept. of Comp. Science, The University of Tennessee, Knoxville, Tennessee.
28. J. K. Reid, *On the method of conjugate gradients for the solution of large sparse systems of linear equations*, in *Large Sparse Sets of Linear Equations*, ed. J. K. Reid, Academic Press, New York (1971).
29. H. Rutishauser, *Theory of gradient methods*, in *Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems*, M. Engeli et al., Birkhäuser, Basel (1959).

30. H. R. Schwarz, *Die Methode der konjugierten Gradienten in der Ausgleichsrechnung*, Zeitschrift für Vermessungswesen 95 (1970), 130–140.
31. E. Stiefel, *Ausgleichung ohne Aufstellung der Gausschen Normalgleichungen*, Wiss. Z. Technische Hochschule Dresden 2 (1952/3), 441–442.
32. G. W. Stewart, *Introduction to Matrix Computation*, Academic Press, New York (1973).
33. K. Tanabe, *Projection method for solving a singular system of linear equations and its application*, Numer. Math. 17 (1971), 203–217.
34. A. van der Sluis, *Condition numbers and equilibration of matrices*, Numer. Math. 14 (1969), 14–23.
35. T. M. Whitney and R. K. Meany, *Two algorithms related to the method of steepest descent*, SIAM J. Numer. Anal. 4 (1967), 109–118.
36. H. Wozniakowski, *Numerical stability of the Chebyshev method for the solution of large linear systems*, Numer. Math. (1977), 191–209.
37. D. M. Young, *Iterative solution of large linear systems*, Academic Press, New York (1971).

DEPARTMENT OF MATHEMATICS
LINKÖPING UNIVERSITY
S-581 83 LINKÖPING
SWEDEN