BIT Numerical Mathematics **43:** 975–989, 2003.
© 2003 *Kluwer Academic Publishers. Printed in the Netherlands.*

SUBSPACE PRECONDITIONED LSQR FOR DISCRETE ILL-POSED PROBLEMS* ⋆

M. JACOBSEN[1], P. C. HANSEN[1] and M. A. SAUNDERS[2]

[1] *Informatics and Mathematical Modelling, Technical University of Denmark, Building 321, DK-2800 Kgs. Lyngby, Denmark. email: {mj,pch}@imm.dtu.dk*

[2] *Department of Management Science and Engineering, Terman Building, Stanford University, Stanford, CA 94305-4026, USA. email: saunders@stanford.edu*

Abstract.

We present a novel implementation of a two-level iterative method for the solution of discrete linear ill-posed problems. The algorithm is algebraically equivalent to the two-level Schur complement CG algorithm of Hanke and Vogel, but involves less work per iteration. We review the algorithm, discuss our implementation, and show promising results from numerical experiments that give insight into the proper use of the algorithm.

AMS subject classification (2000): 65F10, 65F22.

Key words: ill-posed problems, regularization, two-level iterative methods, Schur complement CG method, LSQR.

1 Introduction.

Linear ill-posed problems arise in a variety of applications in science and engineering, and reliable computational algorithms are available for small- and medium-scale problems [6, 18]. However, there is a need for iterative algorithms that can treat large-scale problems, i.e., those with sparse or structured matrices, or where the action of the linear operator is provided by some computational scheme.

This work focuses on a large-scale iterative algorithm for computing solutions to the Tikhonov regularization problem

$$\min_{\mathbf{x}} \quad \|\mathbf{K}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda^2 \|\mathbf{L}\mathbf{x}\|_2^2,$$

in which $\mathbf{K} \in \mathbb{R}^{m \times n}$ is a given matrix, $\mathbf{L} \in \mathbb{R}^{p \times n}$ is a matrix that defines a suitable smoothing norm for the problem, $\mathbf{y} \in \mathbb{R}^m$ is a vector of observed data, and the regularization parameter λ controls the weight between the residual and

* Received October 2002. Revised December 2002. Communicated by Zdeněk Strakoš.

⋆ Partially supported by Danish Natural Science Research Foundation grant 9901581, U.S. National Science Foundation grant CCR-9988205, and U.S. Office of Naval Research grants N00014-96-1-0274 and N00014-02-1-0076.

smoothing norms. We note that both $\mathbf{K}$ and $\mathbf{L}$ may be available either explicitly (as sparse or structured matrices) or implicitly through their action on a vector.

We present an efficient implementation of an iterative algorithm developed by Hanke and Vogel [5], who discussed several 'two-level methods' for the solution of the Tikhonov problem. The algorithm is based on the least squares formulation of the Tikhonov problem,

$$\mathbf{x}_\lambda = \operatorname*{argmin}_{\mathbf{x}} \left\| \widehat{\mathbf{K}}\mathbf{x} - \hat{\mathbf{y}} \right\|_2, \tag{1.1}$$

where we introduce the 'stacked' matrix and vector

$$\widehat{\mathbf{K}} = \begin{bmatrix} \mathbf{K} \\ \lambda\mathbf{L} \end{bmatrix}, \qquad \hat{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}.$$

The key idea in the work of Hanke and Vogel is a splitting of the solution space $\mathbb{R}^n$ into two subspaces, one of them with a small dimension and with basis vectors chosen such that this subspace represents approximate regularized solutions. Similarly to multi-level methods, the residual is projected into a small-dimensional subspace (the coarse grid) in which the problem is solved with a direct method, while the component of the solution in the remaining subspace is computed by an iterative algorithm. Because of this analogy, the methods are denoted two-level methods. The obvious extension to a multi-level method was tested but without success [17].

Two of the three methods in [5] are based on the preconditioning idea for the conjugate gradient (CG) method [2, 8], but we do not treat these preconditioning methods here because the third method performs better, in theory and in practice [5, 9].

The third method in [5] – the only one considered here – is based on the Schur complement CG algorithm; see Axelsson [1]. When the subspace splitting idea is combined with the Schur complement CG algorithm, one obtains a method in which a special Schur complement system is solved iteratively by means of CG.

Our main contribution is to show that this Schur complement system is, in fact, the normal equations for an underlying least squares problem, and to demonstrate how this problem can be solved efficiently by means of the LSQR algorithm [12, 13]. There are two advantages in this approach: the implementation is more efficient, and the algorithm is numerically more robust because it avoids the use of the normal equations. (We focus on the use of LSQR because of its superior stability, and because we want to give all the details of the implementation. Any similar iterative method for least squares problems could be used, such as CGLS [8, 12], and the efficiency would be the same.)

In addition, we discuss various strategies for choosing the basis vectors of the subspace with small dimension, and we demonstrate by examples how this choice affects the convergence of the method.

We first summarize the derivation of the Schur CG algorithm from [5] in Section 2. Next, in Section 3 we show how an equivalent method can be obtained by means of a certain QR factorization, with the advantage of a simpler algorithm with fewer operations per iteration. Then in Section 4 we discuss various details

on how to select the subspace splitting, and in Section 5 we show numerical tests on two test problems.

2 Two-level Schur complement CG.

For the presentation of the Hanke–Vogel algorithm it is convenient to introduce the quantities

$$\mathbf{A} = \mathbf{K}^{\mathrm{T}}\mathbf{K} + \lambda^2\mathbf{L}^{\mathrm{T}}\mathbf{L} = \widehat{\mathbf{K}}^{\mathrm{T}}\widehat{\mathbf{K}}, \qquad \mathbf{b} = \mathbf{K}^{\mathrm{T}}\mathbf{y} = \widehat{\mathbf{K}}^{\mathrm{T}}\mathbf{y}$$

associated with the normal equations for the least squares problem (1.1). For ease of presentation we assume in this section that the matrix $\mathbf{L}$ has full column rank; otherwise the algorithm becomes substantially more complicated [15].

The two levels are created by a splitting of the solution space $\mathbb{R}^n$ into two subspaces $\mathcal{V}$ and $\mathcal{W}$ of dimensions k and $n-k$, respectively. To make the algorithm practical one should choose $k \ll n$. Let the columns of the matrix $\mathbf{V} \in \mathbb{R}^{n\times k}$ span the subspace $\mathcal{V}$, and let the columns of the matrix $\mathbf{W} \in \mathbb{R}^{n\times(n-k)}$ span the subspace $\mathcal{W}$. The solution is obtained in the form

$$\mathbf{x}_\lambda = \mathbf{V}\mathbf{v} + \mathbf{W}\mathbf{w}. \tag{2.1}$$

The columns of $\mathbf{V}$ are not required to be orthonormal, although this is preferable in a numerical implementation. As we shall see, the matrix $\mathbf{W}$ is not needed in the algorithm – it is only necessary for its derivation.

Hanke and Vogel choose the two subspaces to be $\mathbf{L}^{\mathrm{T}}\mathbf{L}$-orthogonal, and the columns of $\mathbf{W}$ to be $\mathbf{L}^{\mathrm{T}}\mathbf{L}$-orthonormal. Thus,

$$\mathbf{V}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{L}\mathbf{W} = \mathbf{0} \quad \text{and} \quad \mathbf{W}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{L}\mathbf{W} = \mathbf{I}.$$

These properties are not needed for the final algorithm as presented below, but they are necessary for the omitted details in the derivation of the algorithm.

First we transform the normal equations $\mathbf{A}\mathbf{x}_\lambda = \mathbf{b}$ to the equivalent form

$$[\mathbf{V}\ \mathbf{W}]^{\mathrm{T}}\mathbf{A}[\mathbf{V}\ \mathbf{W}]\begin{bmatrix}\mathbf{v}\\ \mathbf{w}\end{bmatrix} = [\mathbf{V}\ \mathbf{W}]^{\mathrm{T}}\mathbf{b}, \tag{2.2}$$

which expands into the block system

$$\begin{bmatrix}\mathbf{A}_{11} & \mathbf{V}^{\mathrm{T}}\mathbf{A}\mathbf{W}\\ \mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{V} & \mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{W}\end{bmatrix}\begin{bmatrix}\mathbf{v}\\ \mathbf{w}\end{bmatrix} = \begin{bmatrix}\mathbf{V}^{\mathrm{T}}\mathbf{b}\\ \mathbf{W}^{\mathrm{T}}\mathbf{b}\end{bmatrix}, \tag{2.3}$$

where $\mathbf{A}_{11} = \mathbf{V}^{\mathrm{T}}\mathbf{A}\mathbf{V} \in \mathbb{R}^{k\times k}$. The algorithm in its final form requires $\mathbf{A}_{11}$, and otherwise only uses $\mathbf{K}, \mathbf{L}, \mathbf{b}$ and $\mathbf{V}$.

We now apply block Gaussian elimination to (2.3) as in [1] to obtain

$$\begin{bmatrix}\mathbf{A}_{11} & \mathbf{V}^{\mathrm{T}}\mathbf{A}\mathbf{W}\\ \mathbf{0} & \mathbf{S}\end{bmatrix}\begin{bmatrix}\mathbf{v}\\ \mathbf{w}\end{bmatrix} = \begin{bmatrix}\mathbf{V}^{\mathrm{T}}\mathbf{b}\\ \mathbf{s}\end{bmatrix}$$

with

$$\begin{aligned} \mathbf{S} &= \mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{W} - \mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{V}\mathbf{A}_{11}^{-1}\mathbf{V}^{\mathrm{T}}\mathbf{A}\mathbf{W}, \\ \mathbf{s} &= \mathbf{W}^{\mathrm{T}}\mathbf{b} - \mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{V}\mathbf{A}_{11}^{-1}\mathbf{V}^{\mathrm{T}}\mathbf{b}, \end{aligned}$$

where $\mathbf{S}$ is the Schur complement of $\mathbf{A}_{11}$ in the matrix (2.3). We see that $\mathbf{x}_\lambda$ is obtained from

$$\mathbf{S}\mathbf{w} = \mathbf{s}, \tag{2.4}$$

$$\mathbf{A}_{11}\mathbf{v} = \mathbf{V}^{\mathrm{T}}(\mathbf{b} - \mathbf{A}\mathbf{W}\mathbf{w}), \tag{2.5}$$

$$\mathbf{x}_\lambda = \mathbf{V}\mathbf{v} + \mathbf{W}\mathbf{w}. \tag{2.6}$$

The Schur complement $\mathbf{S} \in \mathbb{R}^{(n-k)\times(n-k)}$ is symmetric positive definite because $\mathbf{A}$ is so [1, Theorem 3.9], and therefore CG comes to mind as a choice for solving the Schur system (2.4) when $n - k$ is large. The resulting algorithm (see below) is expressed in terms of the original matrices $\mathbf{K}$ and $\mathbf{L}$ only. The multiplication with $\mathbf{V}\mathbf{A}_{11}^{-1}\mathbf{V}^{\mathrm{T}}$ can be considered a coarse grid correction step. The rather lengthy details of this derivation are omitted here, but they are available in [5] and [9].

SCHUR COMPLEMENT CG. Computes the Tikhonov regularized solution $\mathbf{x}_\lambda = \operatorname{argmin}_{\mathbf{x}} \|\widehat{\mathbf{K}}\mathbf{x} - \hat{\mathbf{y}}\|_2$. The product $\mathbf{L}^{\mathrm{T}}\mathbf{L}$ must be invertible.

1: $\mathbf{x}_0 = \mathbf{V}\mathbf{A}_{11}^{-1}\mathbf{V}^{\mathrm{T}}\mathbf{b}$
2: $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
3: $\mathbf{y}_0 = (\mathbf{L}^{\mathrm{T}}\mathbf{L})^{-1}\mathbf{r}_0$
4: $\mathbf{d}_0 = \mathbf{r}_0$
5: $\mathbf{z}_0 = \mathbf{y}_0$
6: $\rho_0 = (\mathbf{y}_0)^{\mathrm{T}}\mathbf{r}_0$
7: $i = 0$
8: **repeat**
9: $\quad\mathbf{v}_i = (\mathbf{I} - \mathbf{V}\mathbf{A}_{11}^{-1}\mathbf{V}^{\mathrm{T}}\mathbf{A})\mathbf{z}_i$
10: $\quad\mathbf{w}_i = \mathbf{A}\mathbf{v}_i$
11: $\quad\mathbf{g}_i = (\mathbf{L}^{\mathrm{T}}\mathbf{L})^{-1}\mathbf{w}_i$
12: $\quad\beta_i = \rho_i/(\mathbf{d}_i)^{\mathrm{T}}\mathbf{g}_i$
13: $\quad\mathbf{x}_{i+1} = \mathbf{x}_i + \beta_i\mathbf{v}_i$
14: $\quad\mathbf{r}_{i+1} = \mathbf{r}_i - \beta_i\mathbf{w}_i$
15: $\quad\mathbf{y}_{i+1} = \mathbf{y}_i - \beta_i\mathbf{g}_i$
16: $\quad\rho_{i+1} = (\mathbf{y}_{i+1})^{\mathrm{T}}\mathbf{r}_{i+1}$
17: $\quad\gamma_{i+1} = \rho_{i+1}/\rho_i$
18: $\quad\mathbf{d}_{i+1} = \mathbf{r}_{i+1} + \gamma_{i+1}\mathbf{d}_i$
19: $\quad\mathbf{z}_{i+1} = \mathbf{y}_{i+1} + \gamma_{i+1}\mathbf{z}_i$
20: $\quad i = i + 1$
21: **until** $\|r_i\|_2 > \|\mathbf{r}_{i-1}\|_2$ or $i =$ iteration limit
22: $\mathbf{x}_\lambda = \mathbf{x}_i$

Each iteration involves solving one system with $\mathbf{A}_{11}$ (in step 9) and one with $\mathbf{L}^{\mathrm{T}}\mathbf{L}$ (in step 11). The Cholesky factors of these matrices are the triangular QR

factors of $\widehat{\mathbf{K}}\mathbf{V}$ and $\mathbf{L}$, respectively. If $\mathbf{L}$ is large and sparse then the solution in step 11 can also be computed via an iterative method [14].

Note that one iteration also requires two applications of $\mathbf{A}$ (i.e., two applications of $\widehat{\mathbf{K}}$ and $\widehat{\mathbf{K}}^{\mathrm{T}}$ each) and an application of both $\mathbf{V}$ and $\mathbf{V}^{\mathrm{T}}$. Hence one iteration is at least twice as expensive as one iteration of LSQR or CGLS applied to the original least squares problem (1.1), and the rate of convergence for the two-level method must therefore be at least twice that of LSQR and CGLS for the two-level method to be a better choice.

The stopping criterion is based on the decrease of the residual norm. Since the residual is given in terms of the original system and not the Schur complement a 'breakdown' occurs when the Schur complement residual reaches the machine precision. We observed that the breakdown destroys the assumed relationship between the updated vectors, and the method diverges for the next couple of iterations until the relationships are reestablished.

3 The subspace preconditioned LSQR algorithm.

It is well known that the LSQR algorithm is algebraically identical to CGLS and to the CG algorithm on the normal equation system. However, LSQR can be considered as treating the least squares problem in a more direct way, and in finite precision LSQR is often capable of achieving more accurate results than CGLS and in particular CG. Therefore we wish to find the LSQR equivalent of the Schur complement CG algorithm.

As before we want a solution of the form $\mathbf{x}_\lambda = \mathbf{V}\mathbf{v} + \mathbf{W}\mathbf{w}$ to the least squares system (1.1). Thus we need to obtain

$$\begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} = \underset{\mathbf{v},\mathbf{w}}{\operatorname{argmin}} \left\| \widehat{\mathbf{K}}[\mathbf{V}\ \mathbf{W}] \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} - \hat{\mathbf{y}} \right\|_2 .$$

Introducing the QR factorization

$$\widehat{\mathbf{K}}\mathbf{V} = \mathbf{Q}\widehat{\mathbf{R}} = [\mathbf{Y}\ \mathbf{Z}] \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} = \mathbf{Y}\mathbf{R} \tag{3.1}$$

with $\mathbf{Q}$ orthogonal and $\mathbf{R}$ upper triangular, we get

$$\begin{aligned} \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} &= \underset{\mathbf{v},\mathbf{w}}{\operatorname{argmin}} \left\| \mathbf{Q}^{\mathrm{T}}\widehat{\mathbf{K}}[\mathbf{V}\ \mathbf{W}] \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} - \mathbf{Q}^{\mathrm{T}}\hat{\mathbf{y}} \right\|_2 \\ &= \underset{\mathbf{v},\mathbf{w}}{\operatorname{argmin}} \left\| \begin{bmatrix} \mathbf{Y}^{\mathrm{T}}\widehat{\mathbf{K}}\mathbf{V} & \mathbf{Y}^{\mathrm{T}}\widehat{\mathbf{K}}\mathbf{W} \\ \mathbf{Z}^{\mathrm{T}}\widehat{\mathbf{K}}\mathbf{V} & \mathbf{Z}^{\mathrm{T}}\widehat{\mathbf{K}}\mathbf{W} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} - \begin{bmatrix} \mathbf{Y}^{\mathrm{T}}\hat{\mathbf{y}} \\ \mathbf{Z}^{\mathrm{T}}\hat{\mathbf{y}} \end{bmatrix} \right\|_2 \\ &= \underset{\mathbf{v},\mathbf{w}}{\operatorname{argmin}} \left\| \begin{bmatrix} \mathbf{R} & \mathbf{Y}^{\mathrm{T}}\widehat{\mathbf{K}}\mathbf{W} \\ \mathbf{0} & \mathbf{Z}^{\mathrm{T}}\widehat{\mathbf{K}}\mathbf{W} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} - \begin{bmatrix} \mathbf{Y}^{\mathrm{T}}\hat{\mathbf{y}} \\ \mathbf{Z}^{\mathrm{T}}\hat{\mathbf{y}} \end{bmatrix} \right\|_2 . \end{aligned}$$

Thus, the Tikhonov solution is found from

$$\mathbf{w} = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{Z}^{\mathrm{T}}\widehat{\mathbf{K}}\mathbf{W}\mathbf{w} - \mathbf{Z}^{\mathrm{T}}\hat{\mathbf{y}}\|_2, \tag{3.2}$$

$$\mathbf{R}\mathbf{v} = \mathbf{Y}^{\mathrm{T}}(\hat{\mathbf{y}} - \widehat{\mathbf{K}}\mathbf{W}\mathbf{w}), \tag{3.3}$$

$$\mathbf{x}_\lambda = \mathbf{V}\mathbf{v} + \mathbf{W}\mathbf{w}. \tag{3.4}$$

The partial QR factors of $\widehat{\mathbf{K}}[\mathbf{V}\ \mathbf{W}]$ give $[\mathbf{R}\ \mathbf{Y}^{\mathrm{T}}\widehat{\mathbf{K}}\mathbf{W}]$, which would be the top part of a full QR factorization. After $\mathbf{w}$ is determined, $\mathbf{v}$ is found as if one were continuing back-substitution with the full $\mathbf{R}$ factor in the normal way.

The QR factors in (3.1) can be computed conveniently by means of Householder transformations. Since the number of columns k in the matrix product $\widehat{\mathbf{K}}\mathbf{V}$ is small compared to the dimensions of the problem, the k Householder vectors efficiently represent the orthogonal matrix $\mathbf{Q} = [\mathbf{Y}\ \mathbf{Z}]$ in product form. Then products of the form $\mathbf{Zu}$ and $\mathbf{Z}^{\mathrm{T}}\mathbf{v}$ are done by actually multiplying with $\mathbf{Q}$ and $\mathbf{Q}^{\mathrm{T}}$, which in turn uses the stored Householder transformations, requiring about $4k(m+p)$ flops per product.

3.1 Equivalence.

We now show that the Schur complement system $\mathbf{Sw} = \mathbf{s}$ (2.4) is indeed the normal equations of the least squares system (3.2). First note from the QR factorization of $\widehat{\mathbf{K}}\mathbf{V}$ that $\mathbf{A}_{11} = \mathbf{V}^{\mathrm{T}}\widehat{\mathbf{K}}^{\mathrm{T}}\widehat{\mathbf{K}}\mathbf{V} = \mathbf{R}^{\mathrm{T}}\mathbf{R}$. Hence

$$\begin{aligned}
\mathbf{S} &= \mathbf{W}^{\mathrm{T}}(\mathbf{A} - \mathbf{A}\mathbf{V}\mathbf{A}_{11}^{-1}\mathbf{V}^{\mathrm{T}}\mathbf{A})\mathbf{W} \\
&= \mathbf{W}^{\mathrm{T}}\widehat{\mathbf{K}}^{\mathrm{T}}(\mathbf{I} - \widehat{\mathbf{K}}\mathbf{V}\mathbf{A}_{11}^{-1}\mathbf{V}^{\mathrm{T}}\widehat{\mathbf{K}}^{\mathrm{T}})\widehat{\mathbf{K}}\mathbf{W} \\
&= \mathbf{W}^{\mathrm{T}}\widehat{\mathbf{K}}^{\mathrm{T}}(\mathbf{I} - \mathbf{Y}\mathbf{R}(\mathbf{R}^{\mathrm{T}}\mathbf{R})^{-1}\mathbf{R}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}})\widehat{\mathbf{K}}\mathbf{W} \\
&= \mathbf{W}^{\mathrm{T}}\widehat{\mathbf{K}}^{\mathrm{T}}(\mathbf{I} - \mathbf{Y}\mathbf{Y}^{\mathrm{T}})\widehat{\mathbf{K}}\mathbf{W} \\
&= \mathbf{W}^{\mathrm{T}}\widehat{\mathbf{K}}^{\mathrm{T}}\mathbf{Z}\mathbf{Z}^{\mathrm{T}}\widehat{\mathbf{K}}\mathbf{W},
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{s} &= \mathbf{W}^{\mathrm{T}}(\mathbf{I} - \mathbf{A}\mathbf{V}\mathbf{A}_{11}^{-1}\mathbf{V})\mathbf{b} \\
&= \mathbf{W}^{\mathrm{T}}\widehat{\mathbf{K}}^{\mathrm{T}}(\mathbf{I} - \widehat{\mathbf{K}}\mathbf{V}\mathbf{A}_{11}^{-1}\mathbf{V}^{\mathrm{T}}\widehat{\mathbf{K}}^{\mathrm{T}})\hat{\mathbf{y}} \\
&= \mathbf{W}^{\mathrm{T}}\widehat{\mathbf{K}}^{\mathrm{T}}\mathbf{Z}\mathbf{Z}^{\mathrm{T}}\hat{\mathbf{y}},
\end{aligned}$$

as required. Hence CG applied to the Schur system (2.4) and LSQR or CGLS applied to the least squares system (3.2) produce the same results in infinite precision. In finite precision LSQR usually yields results with slightly higher precision.

We can also show that systems (2.5) and (3.3) are both equivalent to the least squares problem $\min_{\mathbf{v}} \|\widehat{\mathbf{K}}\mathbf{V}\mathbf{v} - (\hat{\mathbf{y}} - \widehat{\mathbf{K}}\mathbf{W}\mathbf{w})\|_2$. The Schur complement approach uses the associated normal equations to solve for $\mathbf{v}$, while our approach uses the QR factorization (3.1).

More important, however, is that the QR approach leads to a simpler iteration scheme and thus a simpler implementation. In particular, in each iteration we only need one multiplication with $\widehat{\mathbf{K}}$ and $\widehat{\mathbf{K}}^{\mathrm{T}}$ (while the Schur complement CG algorithm needs two multiplications). Moreover, there is no linear equation system involving $\mathbf{L}^{\mathrm{T}}\mathbf{L}$, and there is no requirement on the rank of $\mathbf{L}$ (as long as $\widehat{\mathbf{K}}$ has full rank).

3.2 Solving for $\mathbf{Ww}$ *and* $\mathbf{v}$.

Defining $\mathbf{p} = \mathbf{Ww}$, we first show how to proceed in cases where $\mathbf{W}$ is not available. Substituting into (3.2)–(3.4) gives the equations

$$\mathbf{p} = \operatorname*{argmin}_{\mathbf{p}} \|\mathbf{Z}^{\mathrm{T}}\widehat{\mathbf{K}}\mathbf{p} - \mathbf{Z}^{\mathrm{T}}\hat{\mathbf{y}}\|_2, \tag{3.5}$$

$$\mathbf{Rv} = \mathbf{Y}^{\mathrm{T}}(\hat{\mathbf{y}} - \widehat{\mathbf{K}}\mathbf{p}), \tag{3.6}$$

$$\mathbf{x}_\lambda = \mathbf{Vv} + \mathbf{p}, \tag{3.7}$$

in which the main computation is solving for $\mathbf{p}$. The resulting algorithm is displayed below, where for completeness all details of the LSQR iterations are included. The Lanczos vectors are denoted by $\bar{\mathbf{u}}_i$ and $\bar{\mathbf{v}}_i$, and the $\bar{\mathbf{w}}_i$, are work vectors.

Observe that $\mathbf{Z}^{\mathrm{T}}\widehat{\mathbf{K}}\mathbf{V} = \mathbf{Z}^{\mathrm{T}}\mathbf{YR} = \mathbf{0}$, and therefore by construction, the matrix $\mathbf{Z}^{\mathrm{T}}\widehat{\mathbf{K}}$ has a nontrivial null space spanned by the columns of $\mathbf{V}$. For this reason LSQR will eventually diverge as a result of finite precision effects, cf. [12, §6.2]. Our experiments agree with those in [12], leading to a stopping criterion based on the approximation $\bar{\phi}\alpha|c| \approx \|\widehat{\mathbf{K}}^{\mathrm{T}}\mathbf{Zr}\|_2 < \tau$, where $\mathbf{r} = \mathbf{Z}^{\mathrm{T}}(\hat{\mathbf{y}} - \widehat{\mathbf{K}}\mathbf{p})$ is the residual for the least squares problem (3.2), and τ is a tolerance of order 10^{-12}, for example. This stopping criterion ends the iterations just before divergence sets in.

3.3 Solving for $\mathbf{w}$ *and* $\mathbf{v}$.

The difficulties with the rank deficiency in $\mathbf{Z}^{\mathrm{T}}\widehat{\mathbf{K}}$ can be avoided by returning to (3.2)–(3.4) and working with the slightly smaller vector $\mathbf{w}$ and matrix $\mathbf{Z}^{\mathrm{T}}\widehat{\mathbf{K}}\mathbf{W}$. The following lines must be changed:

4: $\alpha_1\bar{\mathbf{v}}_1 = \mathbf{W}^{\mathrm{T}}\widehat{\mathbf{K}}^{\mathrm{T}}\mathbf{Z}\bar{\mathbf{u}}_1$
6: $\mathbf{w}_0 = \mathbf{0}$
11: $\beta_{i+1} = \bar{\mathbf{u}}_{i+1}\mathbf{Z}^{\mathrm{T}}\widehat{\mathbf{K}}\mathbf{W}\bar{\mathbf{v}}_i - \alpha_i\bar{\mathbf{u}}_i$
12: $\alpha_{i+1} = \bar{\mathbf{v}}_{i+1}\mathbf{W}^{\mathrm{T}}\widehat{\mathbf{K}}^{\mathrm{T}}\mathbf{Z}\bar{\mathbf{u}}_{i+1} - \beta_{i+1}\bar{\mathbf{v}}_i$
20: $\mathbf{w}_i = \mathbf{w}_{i-1} + (\phi_i/\rho_i)\bar{\mathbf{w}}_i$
24: Solve $\mathbf{Rv} = \mathbf{Y}^{\mathrm{T}}(\hat{\mathbf{y}} - \widehat{\mathbf{K}}\mathbf{W}\mathbf{w}_{i-1})$ for $\mathbf{v}$
25: $\mathbf{x}_\lambda = \mathbf{Vv} + \mathbf{W}\mathbf{w}_{i-1}$

This approach is practical only if operations with $\mathbf{W}$ and $\mathbf{W}^{\mathrm{T}}$ can be done quickly.

4 The subspace splitting.

The optimal subspace $\mathcal{V}$ consists of the principal k right singular vectors in the SVD of $\mathbf{K}$ if $\mathbf{L} = \mathbf{I}$, or generalized right singular vectors in the GSVD of $(\mathbf{K}, \mathbf{L})$ if $\mathbf{L} \neq \mathbf{I}$. This choice diagonalizes the submatrix $\mathbf{A}_{11}$ and minimizes the condition number of $\mathbf{S}$, which indicates faster convergence of the algorithms. However, the optimal subspace created from the SVD or GSVD is usually not available, and we are forced to choose a less optimal subspace. The hope is that

SUBSPACE PRECONDITIONED LSQR. Computes the Tikhonov regularized solution $\mathbf{x}_\lambda = \operatorname{argmin}_{\mathbf{x}} \|\widehat{\mathbf{K}}\mathbf{x} - \hat{\mathbf{y}}\|_2$. There is no restriction on the rank of $\mathbf{L}$.

1: Compute QR factorization $\widehat{\mathbf{K}}\mathbf{V} = [\mathbf{Y}\ \mathbf{Z}] \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}$
2: $\mathbf{d} = \mathbf{Z}^{\mathrm{T}}\hat{\mathbf{y}}$
3: $\beta_1 \bar{\mathbf{u}}_1 = \mathbf{d}$ % So that $\|\bar{\mathbf{u}}_1\|_2 = 1$
4: $\alpha_1 \bar{\mathbf{v}}_1 = \widehat{\mathbf{K}}^{\mathrm{T}}\mathbf{Z}\bar{\mathbf{u}}_1$ % So that $\|\bar{\mathbf{v}}_1\|_2 = 1$
5: $\mathbf{w}_1 = \bar{\mathbf{v}}_1$
6: $\mathbf{p}_0 = \mathbf{0}$
7: $\bar{\phi}_1 = \beta_1$
8: $\bar{\rho}_1 = \alpha_1$
9: $i = 0$
10: **repeat**
11: $\beta_{i+1}\bar{\mathbf{u}}_{i+1} = \mathbf{Z}^{\mathrm{T}}\widehat{\mathbf{K}}\bar{\mathbf{v}}_i - \alpha_i \bar{\mathbf{u}}_i$ % So that $\|\bar{\mathbf{u}}_{i+1}\|_2 = 1$
12: $\alpha_{i+1}\bar{\mathbf{v}}_{i+1} = \widehat{\mathbf{K}}^{\mathrm{T}}\mathbf{Z}\bar{\mathbf{u}}_{i+1} - \beta_{i+1}\bar{\mathbf{v}}_i$ % So that $\|\bar{\mathbf{v}}_{i+1}\|_2 = 1$
13: $\rho_i = (\bar{\rho}_i^2 + \beta_{i+1}^2)^{1/2}$
14: $c_i = \bar{\rho}_i / \rho_i$
15: $s_i = \beta_{i+1} / \rho_i$
16: $\theta_{i+1} = s_i \alpha_{i+1}$
17: $\bar{\rho}_{i+1} = -c_i \alpha_{i+1}$
18: $\phi_i = c_i \bar{\phi}_i$
19: $\bar{\phi}_{i+1} = s_i \bar{\phi}_i$
20: $\mathbf{p}_i = \mathbf{p}_{i-1} + (\phi_i / \rho_i)\overline{\mathbf{w}}_i$
21: $\overline{\mathbf{w}}_{i+1} = \bar{\mathbf{v}}_{i+1} - (\theta_{i+1}/\rho_i)\overline{\mathbf{w}}_i$
22: $i \leftarrow i + 1$
23: **until** $\bar{\phi}_i \alpha_i |c_{i-1}| < \tau$
24: Solve $\mathbf{R}\mathbf{v} = \mathbf{Y}^{\mathrm{T}}(\hat{\mathbf{y}} - \widehat{\mathbf{K}}\mathbf{p}_{i-1})$ for $\mathbf{v}$
25: $\mathbf{x}_\lambda = \mathbf{V}\mathbf{v} + \mathbf{p}_{i-1}$

a subspace 'close to' the optimal (G)SVD-based subspace is good enough. We know that in most applications the right singular vectors of an ill-posed problem have increasingly more sign changes as the corresponding (generalized) singular values decrease, i.e., they become more oscillatory – a feature we should try to emulate.

In the previous sections the subspace operations were described by means of matrix multiplications with $\mathbf{V}$ and $\mathbf{V}^{\mathrm{T}}$. However, if we select a specific subspace $\mathcal{V}$ then the multiplications can often be done more cheaply by means of 'fast transforms.' Two examples come to mind.

- The wavelet transform; in this paper we use the Daubechies wavelets [3].
- The Fourier transform and variations thereof; to avoid complex results we restrict ourself to the use of the cosine transform denoted DCT-2 in [16].

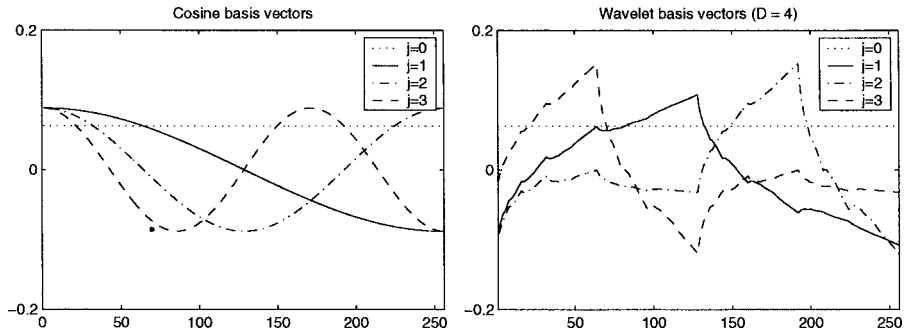

Figure 4.1. The first four basis vectors for DCT-2 (left) and wavelets (right).

Both transforms have the desired property that the basis vectors have higher frequency as the column index grows. This is obviously true for the cosine transform, which is spanned by the vectors

$$\mathbf{V}_{i,j} = \left(\frac{n}{2}\right)^{-1/2} \cos\left((i+1/2)(j-1)\frac{\pi}{2}\right), \quad \text{divided by } \sqrt{2} \text{ if } j = 1. \tag{4.1}$$

The Daubechie wavelets, on the other hand, have a local nature and oscillate only in their domain of support. The wavelet basis is composed of several subspaces of increasing detail,

$$\mathbb{R}^n = \mathcal{V}_0 \bigoplus_{j=1}^{k} \mathcal{W}_j, \quad k = \log_2 n.$$

Because all vectors spanning $\mathcal{W}_j$ must be included to describe all details at the given level, the subspace dimension k should preferably be a power of 2. Figure 4.1 shows the first four basis vectors of the two fast transforms.

The use of these fast transforms will benefit the Schur complement CG algorithm because $\mathbf{V}$ is used twice in each iteration. In the preconditioned LSQR algorithm, $\mathbf{V}$ only appears in the initialization and finalization stages. For comparison we also created preconditioning subspaces from

- The SVD; the optimal subspace is created from the principal k right singular vectors of $\mathbf{K}$.
- regutm; the Regularization Tools package [7] includes a test problem generator regutm that produces random singular vectors with $j-1$ sign changes (where j is the column index).

5 Numerical results.

The numerical tests were performed using matlab [10] with double precision IEEE arithmetic. The wavelet routines are from the wavelet package by Nielsen [11], and the discrete cosine transform is from MATLAB's Signal Processing Toolbox. The test problem heat is from Regularization Tools [7] and was selected because an unpreconditioned CG method shows slow convergence for this problem.

We also test the algorithms on an inverse geomagnetic problem originating from geoscience [4], where the goal is to compute the distribution of magnetic dipole moment from remote measurements. The relationship is described by a Fredholm integral equation of the first kind

$$\int_\Omega K(x,y,z,x',y',z')f(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z = g(x',y',z'),$$

where the kernel K is given by

$$K(x,y,z,x',y',z') = \frac{3(z-z')}{r^5} - \frac{1}{r^3}$$

with $r = ((x-x')^2 + (y-y')^2 + (z-z')^2)^{1/2}$.

The solution to the discretized problem consists of samples of f on a 3-D grid with $n = N^3$ points. The right-hand side consists of data measured at P layers, each layer consisting of N^2 data points and thus leading to a total of $m = PN^2$ data. The problem used here has parameters $N = 12$ and $P = 13$ leading to a matrix $\mathbf{K}$ of dimensions 1872×1728.

This formulation of the problem gives us the option to create a smaller underdetermined problem with $P = 1$, i.e., only one layer of data points, leading to a matrix of dimensions 144×1728. We use the right singular vectors of this matrix to create a preconditioning subspace for the overdetermined system.

More extensive numerical experiments with the Schur complement CG algorithm are available in [9].

5.1 Condition numbers.

Even though the condition number does not tell the entire truth about the convergence of CG and LSQR, it still carries information on what to expect. Furthermore it is independent of the right-hand side and we are able to draw conclusions on a more general level. The actual convergence depends on the distribution of *all* the singular values, i.e., not only the extreme singular values, as well as the actual right-hand side.

Figure 5.1 shows the condition number of the Schur complement $\mathbf{S}$ for the `heat` problem, as a function of the dimension k of the preconditioning subspace. The condition number using the wavelet basis decreases each time a detail level is 'completed,' while the other subspace types lead to smoother decay of the condition number. The basis derived from the SVD gives the smallest condition number, as expected.

5.2 Precision of subspace precondition LSQR vs. Schur CG.

It is known that LSQR often achieves higher precision than symmetric CG on the normal equations, even though the two methods are algebraically equivalent. Figure 5.2 shows the convergence histories and, as expected, we see that LSQR converges a bit faster than Schur CG. We emphasize that the LSQR algorithm is much faster, as the cost per iteration is about half that of Schur CG.

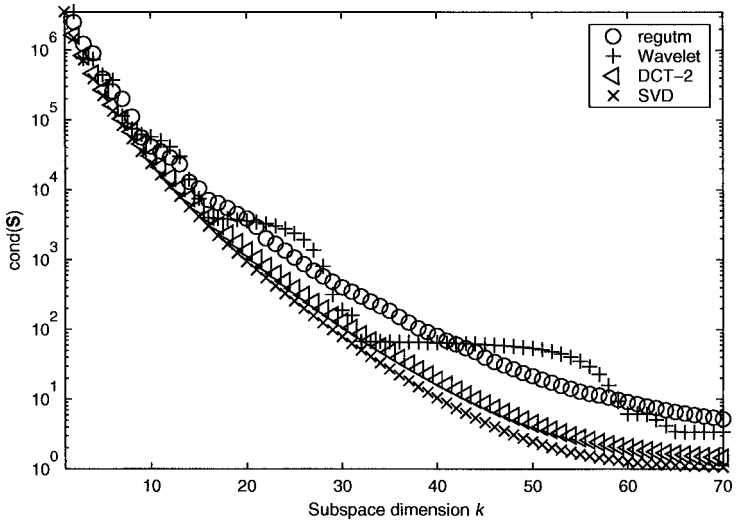

Figure 5.1. Condition number of the Schur complement $\mathbf{S}$ as a function of subspace dimension k. The test problem is heat with $m = n = 1024, \mathbf{L} = \mathbf{I}$ and $\lambda = 10^{-4}$.

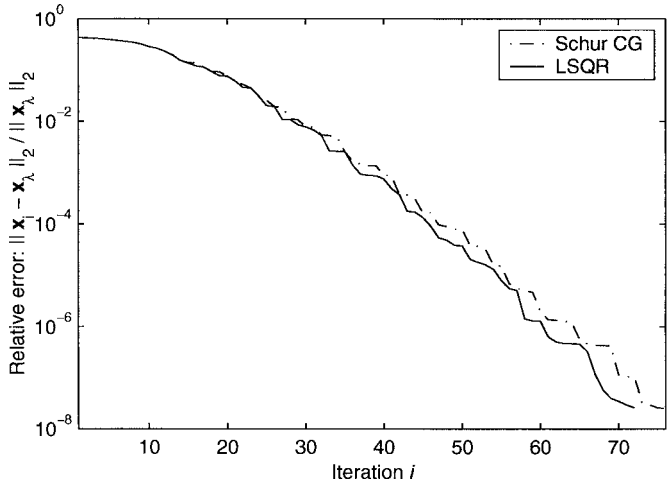

Figure 5.2. Schur complement CG and subspace preconditioned LSQR applied to heat with $m = n = 512, \mathbf{L} = \mathbf{I}$ and $\lambda = 10^{-4}$. The preconditioning subspace is a DCT-2 subspace of dimension $k = 20$.

5.3 Convergence with different subspace types.

Figure 5.3 shows convergence histories for the heat problem with fixed subspace dimensions 32. We see that the SVD-based preconditioner is fastest, followed by a DCT-2 based subspace. Compared to the unpreconditioned LSQR, convergence is very fast for this problem.

Turning to the geomagnetic problem we obtain the convergence history shown in Figure 5.4. Without the preconditioner convergence is almost nonexistent,

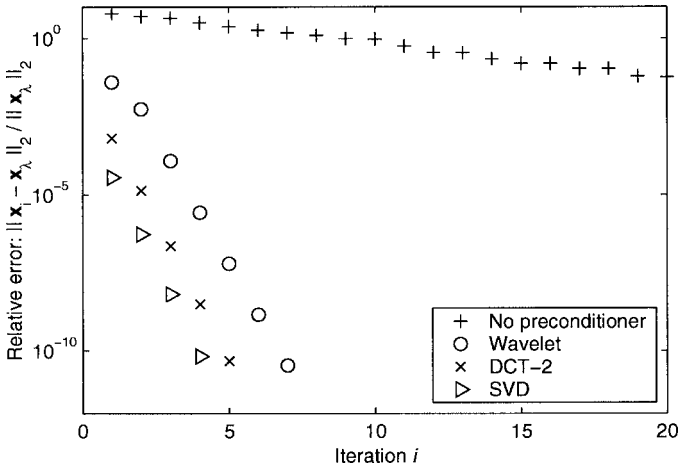

Figure 5.3. Relative error with respect to the *regularized solution* (not the true solution) as a function of iterations for three different subspace types with $k = 32$. The test problem is heat with $m = n = 1024, \mathbf{L} = \mathbf{I}$ and $\lambda = 10^{-5}$.

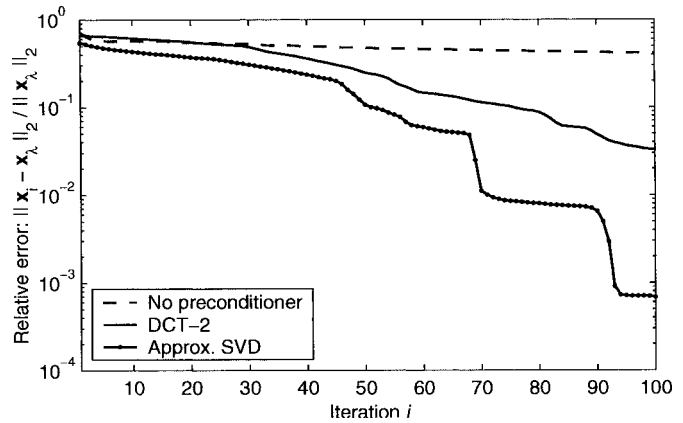

Figure 5.4. Subspace preconditioned LSQR on the geomagnetic problem. The matrix $\mathbf{K}$ is 1872×1728, i.e., 12^3 solution points and 13 layers of 12^2 measurements. The SVD-based preconditioning subspace consists of the right singular vectors from the SVD of a smaller 144×1728 single-layer problem. The regularization parameter is $\lambda = 10^{-2}$, and $\mathbf{L} = \mathbf{I}$.

while LSQR with subspace preconditioning does converge. The DCT-2 subspace works nicely even though it does not take the structure of the problem into account. We also tried the SVD-based subspace from the small version of the problem, which yields even faster convergence.

5.4 Preconditioned LSQR on the unregularized problem.

Some iterative methods in their unpreconditioned version exhibit 'semi-convergence' on unregularized ill-posed problems; i.e., during the initial iterations the solution approaches a regularized solution but at some point starts to converge

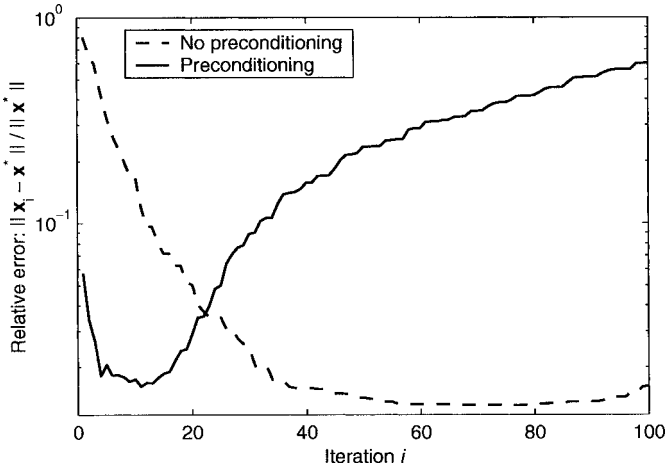

Figure 5.5. Plain and preconditioned LSQR applied to the unregularized problem heat with $m = n = 512, \mathbf{L} = \mathbf{0}$, noise level $\|\mathbf{e}\|_2 = 10^{-4}\|\mathbf{y}\|_2$, and a DCT-2 subspace of dimension $k = 20$. Both methods exhibit 'semi-convergence.'

towards the unwanted least squares solution, cf. §6 in [6]. This implies that early termination of the CG iterations is sometimes a regularization method in itself. CG and hence LSQR in their unpreconditioned versions are known to posses this semi-convergence property. Figure 5.5 illustrates that this is also the case for the preconditioned version of LSQR. Note that the smallest error of the preconditioned method is slightly larger than that of the plain (unpreconditioned) version, but it is reached in fewer iterations.

5.5 Timings.

In the previous sections we only took the iteration numbers into account when evaluating the algorithms. However, the time per iteration is also vital. In addition, the initialization phase, i.e., the creation and factorization of $\widehat{\mathbf{K}}\mathbf{V}$, should be taken into account.

If the subspace transformations are implemented with matrix multiplications we see that the initialization phase costs at least k multiplications with $\widehat{\mathbf{K}}$ to construct $\widehat{\mathbf{K}}\mathbf{V}$. This corresponds to $k/2$ iterations with a standard LSQR or CGLS. Using more efficient subspace transformations like the DCT reduces this startup penalty and is to be recommended. However, if $\widehat{\mathbf{K}}$ is sparse the matrix multiplication approach could be the best choice. Furthermore the initialization phase includes a QR factorization of $\widehat{\mathbf{K}}\mathbf{V}$.

In situations where the same system is to be solved several times one can save and reuse the factorized $\widehat{\mathbf{K}}\mathbf{V}$, thus eliminating the initialization step.

Table 5.1 shows timings for LSQR without preconditioning, Schur complement CG and subspace preconditioned LSQR, broken into three phases. The problem size was selected so that everything was contained in memory. In the Schur

Table 5.1. Timings on the heat problem with $m = n = 1024$, $\mathbf{L} = \mathbf{I}$, $\lambda = 10^{-5}$ and a DCT-2 basis of dimension $k = 8$. The iteration numbers are selected to give a relative error of 10^{-3} in the solution. All operations are dense, and the LSQR implementation is from Regularization Tools [7]

	LSQR	Schur CG with $\mathbf{g}^{(i)}$	Schur CG without $\mathbf{g}^{(i)}$	Precond. LSQR
Initialization (sec)	≈ 0	3	3	3
No. iterations	378	33	33	33
Iteration time (sec)	74	29	13	6
Finalization (sec)	0	0	0	≈ 0
Total time (sec)	74	32	16	9

complement CG algorithm one can skip step 11 and the variable $\mathbf{g}^{(i)}$ when $\mathbf{L} = \mathbf{I}$, which saves substantial time in our matlab implementation.

We see that the time for the initialization phase is equal for the two implementations of the two-level methods – essentially they both require the creation of $\widehat{\mathbf{K}}\mathbf{V}$ and a factorization of this matrix. The finalization stages are negligible.

In the iteration phase we see the strength of the new implementation; a preconditioned LSQR iteration is between two to four times faster than a Schur CG iteration, depending on the matrix $\mathbf{L}$ and the implementation of step 11. Both implementations require 33 iterations to achieve the desired accuracy, while plain LSQR requires 378 iterations.

Note that for preconditioned LSQR, the algorithm of Section 3.2 was used throughout. The same graphs would be obtained with the Section 3.3 modifications, but the operations with $\mathbf{W}$ would increase the timings slightly.

6 Conclusion.

Least squares problems may be preconditioned on the left by any orthogonal matrix, and on the right by any nonsingular matrix. Here we are using $\mathbf{Q}^{\mathrm{T}}$ and $[\mathbf{V}\ \mathbf{W}]$ respectively to decouple the problem into two parts: (3.2) and (3.3). We have shown that the resulting preconditioned LSQR method is equivalent to Schur complement CG. Both methods are adapted to avoid operations with $\mathbf{W}$, but the implementation is simpler for subspace preconditioned LSQR and the work per iteration is halved.

The subspace $\mathbf{V}$ must be chosen wisely for the preconditioning to give fast convergence. The DCT-2 subspace produced good results, while the wavelet subspace turned out to be weaker for the problems we considered. We have illustrated that in some cases a good subspace can be derived from a simpler problem. Because the initialization phase can be expensive in terms of operations, the subspace dimension should be chosen small.

We also observed that the Schur complement CG and subspace preconditioned LSQR algorithms both exhibit semi-convergence on unregularized problems. Thus, regularization can be achieved by early termination of the iterations.

Acknowledgements.

We are grateful to the referees for finding some important corrections. We also acknowledge the SIAM annual meeting (San Diego, 2001) and the IMM Sparse Matrix Computations summer school (Danish Technical University, 2001) for helping this collaboration evolve.

REFERENCES

1. O. Axelsson, *Iterative Solution Methods*, Cambridge Univ. Press, 1994.
2. R. Barret, M. Berry, T. F. Chan, J. Demmel, J. Danato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1994.
3. I. Daubechies, *Ten Lectures on Wavelets*, SIAM, Philadelphia, 1992.
4. M. Fedi, P. Hansen, and V. Paoletti, *SVD and GSVD analysis of depth resolution in potential field inversion*, Geophysics (submitted).
5. M. Hanke and C. R. Vogel, *Two-level preconditioners for regularized inverse problems I: Theory*, Numer. Math., 83 (1999), pp. 385–402.
6. P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*, SIAM, Philadelphia, 1998.
7. P. C. Hansen, *Regularization tools version 3.0 for Matlab 5.2*, Numer. Algorithms, 20 (1999), pp. 195–196.
8. M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Stand., 49 (1952), pp. 409–436.
9. M. Jacobsen, *Two-grid iterative methods for ill-posed problems*, Master's thesis, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark, September 2000.
10. The MathWorks, Inc., *Matlab Manual*, Natick, MA, 6th ed., 2000.
11. O. M. Nielsen, *Wavelets in scientific computing*, PhD thesis, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark, July 1998.
12. C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.
13. C. C. Paige and M. A. Saunders, *LSQR: Sparse linear equations and least squares problems*, ACM Trans. Math. Software, 8 (1982), pp. 195–209.
14. K. L. Riley, *Two-level preconditioners for regularized ill-posed problems*, PhD thesis, Montana State University, Bozeman, 1999.
15. K. L. Riley and C. R. Vogel, *Two-level preconditioners for ill-conditioned linear systems with semidefinite regularization*, J. Comp. Appl. Math. (submitted).
16. G. Strang, *The discrete cosine transform*, SIAM Rev., 41 (1999), pp. 135–147.
17. C. R. Vogel, *Negative results for multilevel preconditioners in image deblurring*, in M. Nielsen, P. Johansen, O. F. Olsen and J. Weickert (eds.), Scale-Space Theories in Computer Vision, Springer, 1999, pp. 489–494.
18. C. R. Vogel, *Computational Methods for Inverse Problems*, SIAM, Philadelphia, 2002.