Preconditioners for Linear Systems Arising in Image Reconstruction

Kyle Riley and Curtis R. Vogel

Department of Mathematical Sciences
Montana State University
Bozeman, MT 59717-0240 USA

ABSTRACT

For the numerical solution of large linear systems, the preconditioned conjugate gradient algorithm can be very effective if one has a good preconditioner. Two distinctly different approaches to preconditioning are discussed for solving systems derived from continuous linear operators of the form $\tilde{K} + \alpha L$, where $\tilde{K}$ is a convolution operator, L is a regularization operator, and α is a small positive parameter. The first approach is circulant preconditioning. The second, less standard, approach is based on a two-level decomposition of the solution space. A comparison of the two approaches is given for a model problem arising in atmospheric image deblurring.

Keywords: conjugate gradient, preconditioning, image deblurring

1. INTRODUCTION

In many applications, solutions to large linear systems

$$A\mathbf{u} = \mathbf{b} \tag{1.1}$$

are desired. If the coefficient matrix A is symmetric positive definite (SPD) then the conjugate gradient (CG) algorithm is guaranteed to yield the solution to this system. To obtain faster convergence, one can apply the preconditioned conjugate gradient (PCG) method. In principle this means finding a matrix M which is easy to invert and for which $M^{-1}A$ is well-conditioned. For details see Golub and Van Loan[3] or Saad.[7]

The choice of the preconditioner M should be motivated by the structure of the matrix A. In two dimensional image deblurring, one often obtains matrices of the form

$$A = K^*K + \alpha L, \tag{1.2}$$

where K is block Toeplitz with Toeplitz blocks (BTTB), L is symmetric positive semidefinite and sparse, and α is a small positive parameter. K is the discretization of a convolution integral operator, and hence it is a full (nonsparse) matrix and has eigenvalues which cluster at zero. On the other hand L is the discretization of a regularization operator, typically the identity operator or a diffusion operator. In the latter case, the eigenvalues of this operator cluster at infinity.

Circulant preconditioners have been shown to be effective for solving (1.1)-(1.2). The key ideas, first put forth by Strang,[8] are that (i) Toeplitz matrices can be well-approximated by circulant matrices and (ii) circulant matrices can be easily inverted using the fast Fourier transform (FFT). These basic ideas have been extended to the BTTB case by R. Chan and Jin[1] and T. Chan and Olkin.[2]

The two-level preconditioners presented here were recently developed by Hanke and Vogel.[5,9] These papers also contain a convergence analysis for these preconditioners. The key idea is that a matrix arising in the discretization of a compact operator can be well-approximated by a matrix of low rank. With an integral operator having a smooth kernel, coarse grid projections can be used to construct this low rank approximation. The BTTB structure plays no role, except to simplify the application of the coarse grid projection operators. Consequently, two-level preconditioners are useful for a much broader range of problems than are circulant preconditioners.

Other author information: (Send correspondence to Curtis R. Vogel)
Curtis R. Vogel: e-mail: vogel@math.montana.edu; world wide web url://www.math.montana.edu/∼vogel
Kyle Riley: e-mail: riley@math.montana.edu; world wide web url://www.math.montana.edu/∼imsgkri2

In this paper, we will compare the performance of two-level preconditioners and circulant preconditioners on a model problem arising in atmospheric image deblurring. In this problem, the image that is observed by a ground based telescope is degraded by the effects of atmospheric turbulence on the light emanating from an object in space.[6] The goal is to use the blurred image and a quantitative description of the blurring, the point spread function (PSF), to reconstruct the object. Since this problem is ill-posed, we apply regularized least squares to obtain a stable, accurate approximation. We choose $\mathbf{u}$ to minimize the quadratic functional

$$\|K\mathbf{u} - \mathbf{z}\|^2 + \alpha \mathbf{u}^T L \mathbf{u}. \tag{1.3}$$

Here the matrix K is the discretization of a convolution integral operator whose kernel is the PSF, $\mathbf{z}$ is the observed blurred image, L is a discretization of a regularization operator, and α is the regularization parameter. The solution $\mathbf{u}$ satisfies (1.1)-(1.2) with $\mathbf{b} = K^*\mathbf{z}$. The matrix A can be quite large. For example, for one of the 256×256 pixel images presented in the final section, the corresponding matrix A is $256^2 \times 256^2$, with more than 4 billion nonzero entries.

In the computations carried out below, we replace the matrix A in (1.2) by

$$A = \tilde{K} + \alpha L, \tag{1.4}$$

where $\tilde{K}$ is a BTTB approximation to K^*K (Note that products of (block) Toeplitz matrices need not be (block) Toeplitz, so in general $\tilde{K} \neq K^*K$.) In addition, if L is a discretized diffusion operator, we impose homogeneous Dirichlet boundary conditions on the solution u. These modifications greatly simplify the application of the projection operators, since L is then invertible and the projectors preserve BTTB structure. For the test problems considered in this paper, replacing (1.2) by (1.4) seems to have little effect on either the solution to (1.1) or the performance of the PCG methods used to obtain the solution. For results of numerical experiments where this replacement is *not* made, see.[9]

This paper is organized as follows. In Section 2 we review circulant preconditioners. Two-level preconditioners are discussed in Section 3. In Section 4 we present a comparison of the preconditioning methods applied to two different sets of data. The final section contains a summary.

2. CIRCULANT PRECONDITIONERS

We use the circulant preconditioner implementation presented by Hanke and Nagy.[4] Given a symmetric BTTB matrix A, these preconditioners are constructed as follows: Take the first column of A and construct from it an extended matrix A_{ext}, which is block circulant with circulant blocks. In principle, the action of the preconditioner on a given n-vector $\mathbf{r}$ can be carried out by applying the inverse of A_{ext} to the extension of $\mathbf{r}$, and then restricting the result. In practice, this inversion is carried out using fast Fourier transforms (FFT's), and the matrices are never stored. Instead, if A consists of n_x^2 blocks, each of which is $n_x \times n_x$ (so $n = n_x^2$), an $n_x \times n_x$ array is constructed from the first column of A. Let $\texttt{array}(A)$ denote this array. A_{ext} has a corresponding $2n_x \times 2n_x$ array, which we denote by $\texttt{array}(A_{ext})$. Let $\hat{A}_{ext}$ denote the two-dimensional FFT of $\texttt{array}(A_{ext})$. In the following algorithm, IFFT denotes the two-dimensional inverse fast Fourier transform, $\texttt{array}(\mathbf{r})$ denotes the $n_x \times n_x$ array associated with an n_x^2- vector $\mathbf{r}$, and $\texttt{vector}(R)$ denotes the n_x^2-vector corresponding to an $n_x \times n_x$ array. The `./` indicates component-wise division. The details of the extension operator `extend` and the restriction operator `restrict` are given in Hanke and Nagy.[4]

Algorithm 2.1 *Circulant Preconditioning*
$\mathbf{s} = M_{circ}^{-1}\mathbf{r}$

$r_{ext} = \texttt{extend}(\texttt{array}(\mathbf{r}))$;
$\hat{s}_{ext} = \texttt{FFT}(r_{ext})$ ./ $\hat{A}_{ext}$;
$s_{ext} = \texttt{IFFT}(\hat{s})$;
$\mathbf{s} = \texttt{vector}(\texttt{restrict}(s_{ext}))$;

These same operators are used in the construction and the application of the matrix $\tilde{K}$, which is a BTTB approximation to K^*K. Let k_1 denote the first column of K. We first compute the $2n_x \times 2n_x$ array

$$\hat{\tilde{K}} = |\texttt{FFT}(\texttt{extend}(\texttt{array}(k_1)))|^2. \tag{2.1}$$

Then given an n-vector $\mathbf{v}$, we compute matrix-vector products via

$$\tilde{K}\mathbf{v} = \texttt{vector}(\texttt{restrict}(\texttt{IFFT}(\hat{\tilde{K}}.*\texttt{FFT}(\texttt{extend}(\texttt{array}(\mathbf{v})))))), \tag{2.2}$$

where `.*` denotes component-wise multiplication.

3. TWO-LEVEL PRECONDITIONERS

In this section, we will describe three separate algorithms: additive Schwarz PCG, multiplicative Schwarz PCG, and Schur complement CG. We will assume that A is an $n \times n$ SPD matrix of the form (1.4), L is SPD, and $1 \leq N < n$. The $k \times k$ identity matrix will be denoted by I_k. Consider the decomposition $\mathbb{R}^n = V_\Phi \oplus V_\Psi$, where $V_\Phi = \text{span}\{\phi_i\}_{i=1}^N$, $V_\Psi = \text{span}\{\psi_j\}_{j=N+1}^n$, and the basis vectors ϕ_i and ψ_j are picked so that

$$\phi_i^T L \psi_j = 0 \tag{3.1a}$$
$$\psi_i^T L \psi_j = \delta_{ij}. \tag{3.1b}$$

Let Φ and Ψ represent the matrices whose columns are made of the ϕ_i's and ψ_j's respectively. Define the matrices G, P, and Q by

$$G = \Phi^* L \Phi \tag{3.2}$$
$$P = \Phi G^{-1} \Phi^* L \tag{3.3}$$
$$Q = I_n - P = \Psi \Psi^* L. \tag{3.4}$$

The matrix P is an L-orthogonal projector onto V_Φ and Q is the complementary L-orthogonal projector onto V_Ψ.[9] The matrices Φ and Ψ can be adjoined to form the matrix $[\Phi\Psi]$ that will transform the system (1.1) into

$$[\Phi\Psi]^* A [\Phi\Psi]\mathbf{x} = [\Phi\Psi]^* \mathbf{b},$$

where

$$\mathbf{x} = [\Phi\Psi]^{-1}\mathbf{u}. \tag{3.5}$$

Taking A as in (1.4), carrying out the necessary multiplications in the transformation, and applying properties (3.1), the system takes the form

$$\begin{bmatrix} \Phi^* A \Phi & \Phi^* \tilde{K} \Psi \\ \Psi^* \tilde{K} \Phi & \Psi^* \tilde{K} \Psi + \alpha I_{n-N} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \Phi^* \mathbf{b} \\ \Psi^* \mathbf{b} \end{bmatrix}. \tag{3.6}$$

By choosing ψ_j's so that $\Psi^* \tilde{K} \Psi$ is small, an approximation to the block matrix in (3.6) is

$$\tilde{A} = \begin{bmatrix} \Phi^* A \Phi & \Phi^* \tilde{K} \Psi \\ \Psi^* \tilde{K} \Phi & \alpha I_{n-N} \end{bmatrix} \tag{3.7}$$

Denoting the entries by A_{ij}, this matrix has the standard splitting $\tilde{A} = \tilde{L} + \tilde{D} + \tilde{U}$ with

$$\tilde{L} = \begin{bmatrix} 0 & 0 \\ A_{21} & 0 \end{bmatrix} \qquad \tilde{D} = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix} \qquad \tilde{U} = \begin{bmatrix} 0 & A_{12} \\ 0 & 0 \end{bmatrix} \tag{3.8}$$

3.1. Additive Schwarz Preconditioning

In principle the application of one step of the additive Schwarz preconditioner to system (1.1) is carried out in the following manner. First transform the system to (3.6). Next, motivated by classical Jacobi iteration,[7] replace the coefficient matrix by the block diagonal matrix $\tilde{D}$ obtained from (3.7)-(3.8) and solve the resulting system

$$\begin{aligned} A_{11}\mathbf{x}_1 &= \Phi^*\mathbf{b} \\ \alpha\mathbf{x}_2 &= \Psi^*\mathbf{b}, \end{aligned}$$

where

$$A_{11} = \Phi^* A \Phi. \tag{3.9}$$

Using the equations (3.2)-(3.4), the following transpires:

$$\begin{aligned} \Psi\mathbf{x}_2 &= \frac{1}{\alpha}\Psi\Psi^* L L^{-1}\mathbf{b} \\ &= \frac{1}{\alpha} Q L^{-1}\mathbf{b} \\ &= \frac{1}{\alpha}(I - \Phi G^{-1}\Phi^* L)L^{-1}\mathbf{b} \\ &= \frac{1}{\alpha}\left(L^{-1}\mathbf{r} - \Phi G^{-1}\Phi^*\mathbf{b}\right). \end{aligned} \tag{3.10}$$

Finally, back-transform using (3.5) to obtain

$$\mathbf{u} = \Phi\mathbf{x}_1 + \Psi\mathbf{x}_2. \tag{3.11}$$

It follows that the application of the additive Schwarz preconditioner to a vector $\mathbf{r}$ is given by

$$M_{AS}^{-1}\mathbf{r} = \Phi(A_{11}^{-1}\Phi^*\mathbf{r} - \frac{1}{\alpha}G^{-1}\Phi^*\mathbf{r}) + \frac{1}{\alpha}L^{-1}\mathbf{r}. \tag{3.12}$$

This preconditioner works well when the transformed system (3.6) is nearly diagonal and the term $\Psi^*\tilde{K}\Psi$ is small. When the off-diagonal terms in (3.6) are significant, a preconditioner motivated by classical symmetric Gauss Seidel iteration[7] is more effective.

3.2. Symmetric Multiplicative Schwarz Preconditioning

Instead of the diagonal matrix $\tilde{D}$ used in the derivation of additive Schwarz, replace the block 2×2 matrix in (3.6) by

$$\tilde{M} = (\tilde{D} + \tilde{U})\tilde{D}^{-1}(\tilde{D} + \tilde{L}),$$

where $\tilde{L}$, $\tilde{D}$, $\tilde{U}$ are the terms in the splitting (3.7)-(3.8). It follows that the application of the symmetric multiplicative Schwarz preconditioner to a vector $\mathbf{r}$ can be expressed as

$$\begin{aligned} M_{SMS}^{-1}\mathbf{r} &= \Phi A_{11}^{-1}(\mathbf{r}_1 - A_{12}A_{22}^{-1}(\mathbf{r}_2 - A_{21}A_{11}^{-1}\mathbf{r}_1)) + A_{22}^{-1}(\mathbf{r}_2 - A_{21}A_{11}^{-1}\mathbf{r}_1) \\ &= \Phi A_{11}^{-1}(\mathbf{r}_1 - \alpha^{-1}\Phi^*\tilde{K}\Psi(\mathbf{r}_2 - \Psi^*\tilde{K}\Phi A_{11}^{-1}\mathbf{r}_1)) + \alpha^{-1}(\mathbf{r}_2 - \Psi^*\tilde{K}\Phi A_{11}^{-1}\mathbf{r}_1), \end{aligned} \tag{3.13}$$

where $\mathbf{r}_1 = \Phi\mathbf{r}$ and $\mathbf{r}_2 = \Psi\mathbf{r}$. We again use (3.2)-(3.4) to derive

Algorithm 3.1 *Symmetric Multiplicative Schwarz Preconditioning*
$\mathbf{s} = M_{SMS}^{-1}\mathbf{r}$

$\mathbf{v} = \Phi A_{11}^{-1}\Phi^*\mathbf{r}$;
$\mathbf{e} = \mathbf{r} - \tilde{K}\mathbf{v}$;
$\mathbf{w} = L^{-1}\mathbf{e}$;
$\mathbf{x} = \mathbf{w} - \Phi G^{-1}\Phi^*\mathbf{e}$;
$\mathbf{s}_Q = \mathbf{x}/\alpha$;
$\mathbf{y} = \mathbf{r} - \tilde{K}\mathbf{s}_Q$;
$\mathbf{s}_P = \Phi A_{11}^{-1}\Phi^*\mathbf{y}$;
$\mathbf{s} = \mathbf{s}_P + \mathbf{s}_Q$;

3.3. Schur Complement CG Iteration

The derivation of this method is somewhat different from the previous two in that neither the approximation (3.7) nor the splitting (3.8) are used. Let A_{ij} denote the components of the 2×2 block matrix in (3.6) (in particular, $A_{22} = \Psi^* \tilde{K} \Psi + \alpha I_{n-N}$). In principle, we first apply block Gaussian elimination to the transformed system (3.6). This gives

$$S\mathbf{x}_2 = \Psi^*\mathbf{b} - A_{21}A_{11}^{-1}\Phi^*\mathbf{b} \tag{3.14}$$

$$\mathbf{x}_1 = A_{11}^{-1}\Phi^*\mathbf{b} - A_{11}^{-1}A_{12}\mathbf{x}_2, \tag{3.15}$$

where

$$S = A_{22} - A_{21}A_{11}^{-1}A_{12} \tag{3.16}$$

is the Schur Complement of A_{22} relative to the coefficient matrix in (3.6). We can solve (3.14) using standard CG iteration, and then back-transforms using (3.5). As with the previous two methods, (3.2)-(3.4) can be used to express the computations in terms of the original matrices.[5]

Algorithm 3.2 *Schur Complement Conjugate Gradient Iteration*

$\mathbf{u} = \Phi A_{11}^{-1}\Phi^*\mathbf{b}$;
$\mathbf{r} = \mathbf{b} - A\mathbf{u}$; $\mathbf{y} = L^{-1}\mathbf{r}$;
$\delta_0 = \mathbf{r}^{\mathbf{T}}\mathbf{y}$;
$\mathbf{z} = \mathbf{r}$; $\mathbf{d} = \mathbf{y}$;
Begin CG iterations
  $\mathbf{v} = (I - \Phi A_{11}^{-1}\Phi^* A)\mathbf{d}$;
  $\mathbf{w} = A\mathbf{v}$; $\mathbf{g} = L^{-1}\mathbf{w}$;
  $\tau = \delta_0/(\mathbf{z}^{\mathbf{T}}\mathbf{g})$;
  $\mathbf{u} = \mathbf{u} + \tau\mathbf{v}$;
  $\mathbf{r} = \mathbf{r} - \tau\mathbf{w}$; $\mathbf{y} = \mathbf{y} - \tau\mathbf{g}$;
  $\delta_1 = \mathbf{r}^T\mathbf{y}$;
  $\beta = \delta_1/\delta_0$;
  $\delta_0 = \delta_1$;
  $\mathbf{z} = \mathbf{r} + \beta\mathbf{z}$; $\mathbf{d} = \mathbf{y} + \beta\mathbf{d}$;
end CG iterations

Note that this algorithm requires only one inversion of A_{11} and one inversion of L per CG iteration. Moreover, it does not use the matrix G at all, whereas each application of the symmetric multiplicative Schwarz preconditioner requires two inversions of A_{11}, an inversion of L, and an inversion of G for every iteration. Perhaps more significant in terms of computational cost is the fact that one Schur complement PCG iteration requires 2 applications of A, while symmetric multiplicative Schwarz CG requires 1 application of A and 2 applications of $\tilde{K}$. Since L is sparse, A and $\tilde{K}$ cost about the same to apply. Note that $\tilde{K}$ can be applied using one two-dimensional FFT/IFFT pair.

4. NUMERICAL RESULTS

A 256×256 image, simulating the blurring effects of atmospheric turbulence on light from a satellite in earth orbit, was provided to us by the Starfire Optical Range (SOR), USAF Phillips Laboratory at Kirkland AFB, New Mexico. The satellite object and the noisy blurred image are both displayed in Fig. 1. Also shown in this figure are the reconstructions obtained using identity (i.e., $L = I$) and negative Laplacian regularization. The number of pixels in the image array is $n = 256^2$. However, with Laplacian regularization we used a slightly smaller 255×255 image array. This allowed us to use standard piecewise linear multigrid intergrid transfer operators to compute projections in our two-level preconditioning schemes. The regularization parameters α used were close to optimal in the sense of minimizing the L^2 norm of the difference between the true object (upper left) and the reconstructed objects (lower row).

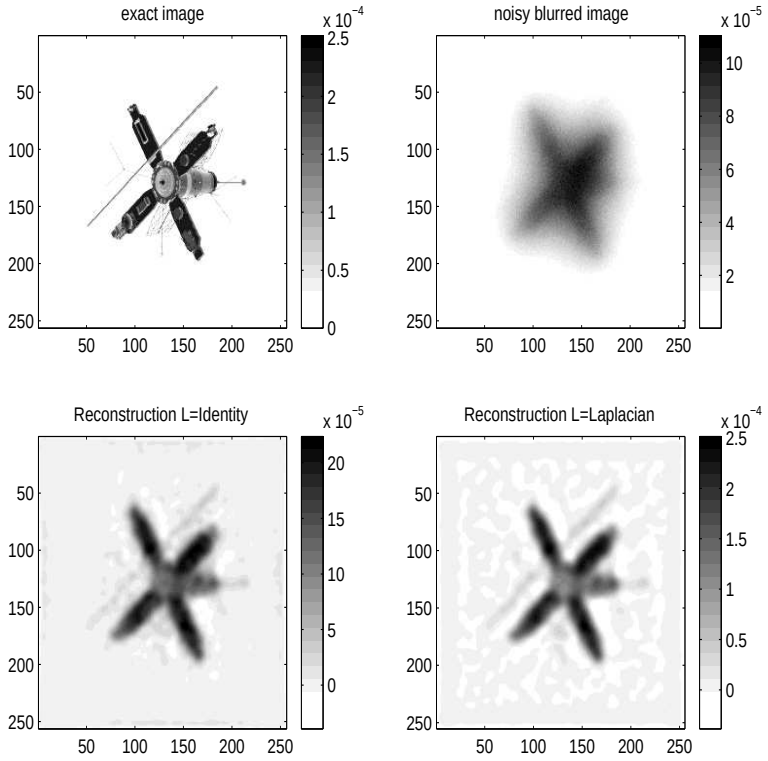

Figure 1. Simulated Data and Reconstructions. At the top left is the true object. Noisy, blurred image data appears at the top right. At the lower left is the reconstruction obtained with identity regularization, piecewise constant basis functions, and $\alpha = 2 \times 10^{-13}$. The lower right reconstruction used Laplacian regularization, piecewise linear basis functions, and $\alpha = 10^{-12}$.

A comparison of the various two-level preconditioning schemes is presented in Fig. 2. Also shown is the performance of plain CG. Plotted against iteration count is the relative solution error norm, $||u_k - u_*||/||u_*||$, where u_* denotes the exact solution to (1.1), and u_k represents the k^{th} iterate. Note that additive Schwarz PCG performs significantly worse than plain CG. Initially, Schur complement CG performs somewhat better than symmetric multiplicative Schwarz PCG, but the discrepancy between the two disappears with increasing iteration count. Both these methods converge far more rapidly than plain CG. In addition, each iteration of Schur complement CG is computationally much less expensive than symmetric multiplicative Schwarz PCG. Hence, it is the method of choice for this particular problem, and the remaining comparisons will show only the two-level Schur complement CG results.

Figures 3 through 5 compare circulant PCG performance against that of two-level Schur complement CG. In each figure, results for two separate coarse grid levels are presented, and simulated data was used. The differences between the various figures lie in the type of regularization used (identity vs. Laplacian) and the type of coarse grid basis functions used (piecewise constant vs. piecewise linear). Schur complement CG with $N = 31^2$ (or $N = 32^2$ in Fig. 3) coarse grid functions clearly shows the most rapid convergence in all 3 figures. Circulant PCG clearly shows the slowest convergence in all but the last figure, with Laplacian regularization, where there is a cross-over with Schur complement/$N = 15^2$ after 9 iterations. At this point, the relative solution error is negligible, so that one might argue that Schur complement is better from a convergence standpoint even in this case with the smaller number of coarse grid functions. It is striking that circulant PCG performs so poorly at early iterations. An examination of error plots reveals that the error in the first few iterations is concentrated on the boundary.

Our final Schur complement vs. circulant comparison was generated with observed data obtained from the SOR. The results are much the same as in Fig. 5, even though the system size, $n = 128^2$ is somewhat smaller.

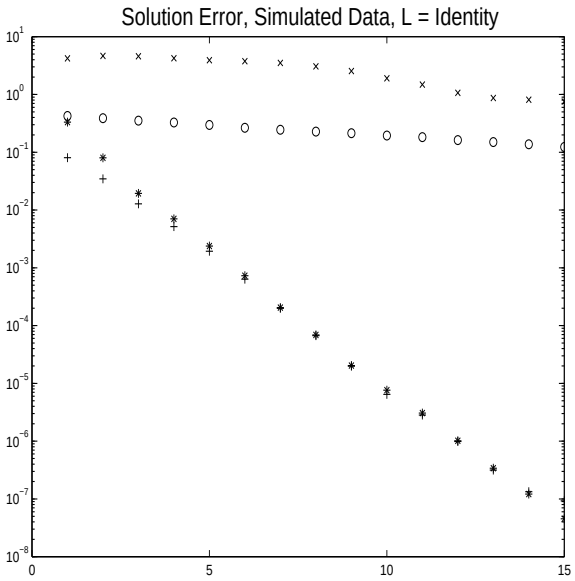

Figure 2. Performance of two-level preconditioners, showing relative solution error norm vs. iteration count. Circles (o) represent plain CG; crosses (x) represent additive Schwarz PCG; stars ($*$) represent symmetric multiplicative Schwarz PCG; and pluses ($+$) represent Schur complement CG. Simulated data was used, with $n = 256^2$, $\alpha = 2 \times 10^{-13}$, and $N = 16^2$ piecewise constant coarse grid basis functions.

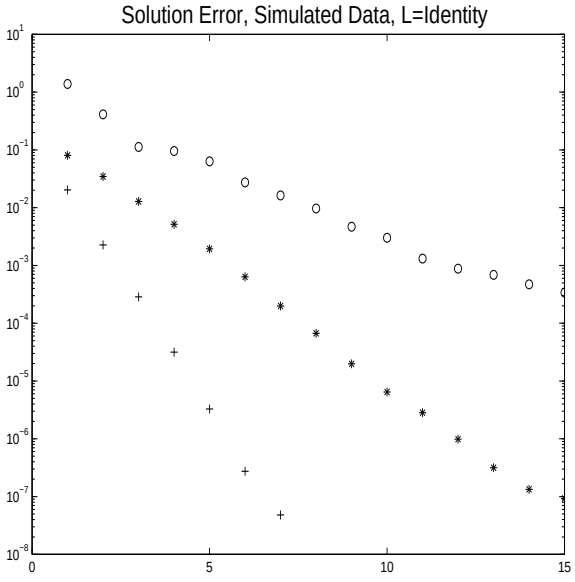

Figure 3. Comparison of Schur complement PCG vs circulant PCG on simulated data with $n = 256^2$, identity regularization with $\alpha = 2 \times 10^{-13}$, and piecewise constant coarse grid basis functions. Circles (o) represent circulant PCG; stars ($*$) represent Schur complement CG with $N = 16^2$ coarse grid basis functions; and pluses ($+$) representSchur complement CG with $N = 32^2$.

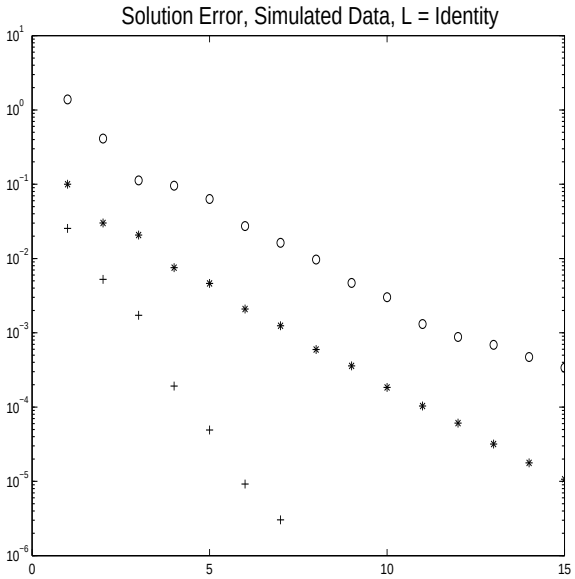

Figure 4. Comparison of Schur complement PCG vs circulant PCG on simulated data with $n = 255^2$, identity regularization with $\alpha = 2 \times 10^{-13}$, and piecewise linear coarse grid basis functions. Circles (o) represent circulant PCG; stars ($*$) represent Schur complement CG with $N = 15^2$ coarse grid basis functions; and pluses ($+$) representSchur complement CG with $N = 31^2$.

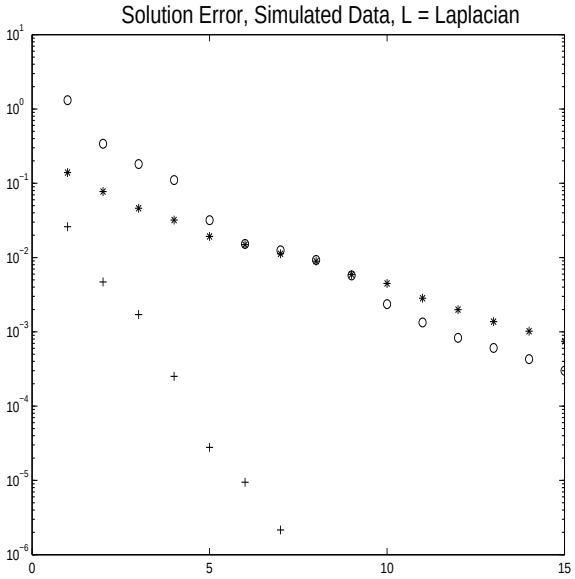

Figure 5. Comparison of Schur complement PCG vs circulant PCG on simulated data with $n = 255^2$, Laplacian regularization with $\alpha = 10^{-12}$, and piecewise linear coarse grid basis functions. Circles (o) represent circulant PCG; stars ($*$) represent Schur complement CG with $N = 15^2$ coarse grid basis functions; and pluses ($+$) representSchur complement CG with $N = 31^2$.

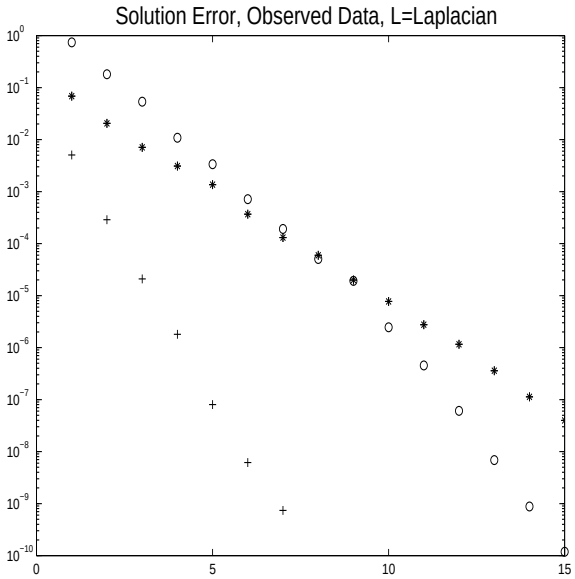

Figure 6. Comparison of Schur complement PCG vs circulant PCG on observed data with $n = 127^2$, Laplacian regularization with $\alpha = 10^{-10}$, and piecewise linear coarse grid basis functions. Circles (o) represent circulant PCG; stars ($*$) represent Schur complement CG with $N = 15^2$ coarse grid basis functions; and pluses ($+$) representSchur complement CG with $N = 31^2$.

5. SUMMARY

A comparison was made of several preconditioned CG methods on a number of large structured linear systems arising in image deblurring. Of the two-level methods tested, the Schur complement CG was superior due to its rapid convergence and relatively low cost per iteration. In terms of convergence rates, in these tests Schur complement CG was also superior to circulant PCG, except for one case (Laplacian regularization) at relatively large iterations counts with correspondingly tiny solution error levels. However, each Schur complement CG iteration is somewhat more expensive than a circulant PCG iteration, particularly when a relatively large number of coarse grid basis functions are used.

REFERENCES

1. R. Chan and X. Jin, "A Family of Block Preconditioners for Block Systems", *SIAM J. Sci. Stat. Comput.*, **13**, pp. 1218-1235, 1992.
2. T. Chan and J. Olkin, "Circulant Preconditioners for Toeplitz-block matrices", *Numerical Algorithms*,**6**, pp. 89-101, 1994.
3. G. H. Golub and C. F. Van Loan, *Matrix Computations*, John Hopkins University Press, 1996.
4. M. Hanke and J. Nagy, "Restoration of Atmospherically Blurred Images by Symmetric Indefinite Conjugate Gradient Techniques", *Inverse Problems*, **12**, pp. 157-173, 1996.
5. M. Hanke and C. R. Vogel, "Restoration of Atmospherically Blurred Images by Symmetric Indefinite Conjugate Gradient Techniques", preprint, 1998.
6. M. C. Roggeman and B. Welsh, *Imaging Through Turbulence*, CRC Press, 1996.
7. Y. Saad, *Iterative Methods for Sparse linear systems*, PWS Publishing Company, 1996.
8. G. Strang, "A Proposal for Teoplitz Matrix Calculations", *Stud. Appl. Math*, **74**, pp. 171-176, 1986.
9. C. R. Vogel and M. Hanke "Two-Level Preconditioners for Regularized Inverse Problems 2: Implementation and Numerical Results", preprint, 1998.