CAUCHY-LIKE PRECONDITIONERS FOR 2-DIMENSIONAL ILL-POSED PROBLEMS *

MISHA E. KILMER†

Abstract. Ill-conditioned matrices with block Toeplitz, Toeplitz block (BTTB) structure arise from the discretization of certain ill-posed problems in signal and image processing. We use a preconditioned conjugate gradient algorithm to compute a regularized solution to this linear system given noisy data. Our preconditioner is a Cauchy-like block diagonal approximation to an orthogonal transformation of the BTTB matrix. We show the preconditioner has desirable properties when the kernel of the ill-posed problem is smooth: the largest singular values of the preconditioned matrix are clustered around one, the smallest singular values remain small, and the subspaces corresponding to the largest and smallest singular values, respectively, remain unmixed. For a system involving np variables, the preconditioned algorithm costs only $O(np(\lg n + \lg p))$ operations per iteration. We demonstrate the effectiveness of the preconditioner on three examples.

Key words. Regularization, ill-posed problems, Toeplitz, Cauchy-like, preconditioner, conjugate gradient, normal equations, image processing, deblurring

AMS(MOS) subject classifications. 65R20, 45L10, 94A12

March 28, 1997

1. Introduction. The two-dimensional integral equation

$$\int_{\alpha_l}^{\alpha_u} \int_{\beta_l}^{\beta_u} t(\alpha,\beta,\gamma,\delta)\hat{f}(\alpha,\beta)d\alpha d\beta = \hat{g}(\gamma,\delta)$$

is often used to describe the process by which data in signal and image processing applications is acquired. In optics, for example, t is called the *point spread function* and describes the response of the system or measuring device to a single point of light at coordinates (α,β) [10]. Thus if the values $\hat{f}(\alpha,\beta)$ represent light intensities reflected from a three-dimensional object, the integral equation might be used to model the blurring of that object when its picture is taken using a camera with a warped lens.

For simplicity, suppose quadrature is used to discretize the integral, and suppose p is the number of grid points α_j in the α direction and n is the number of grid points β_l in the β direction. The integral equation becomes a system of np linear equations of the form

$$T\hat{f} = \hat{g} \tag{1}$$

where $\hat{f}$ is $np \times 1$ with entries $\hat{f}(\alpha_j,\beta_l)$, $1 \le j \le p$, $1 \le l \le n$. We note that many other discretization methods for the integral equation yield a system of np linear equations in which p and n have analogous definitions.

A Toeplitz matrix T_i is one whose elements are constant along diagonals; that is, the (k,j) entry in T_i is given by $t^{(i)}_{k-j}$. In applications, properties of the kernel, discretization process, and a suitable ordering of unknowns can cause T to have a block Toeplitz structure in which each $p \times p$ block is Toeplitz. This structure arises, for

* This work was supported by the National Science Foundation under Grant CCR 95-03126.

† Applied Mathematics Program, University of Maryland, College Park, MD 20742 (mek@math.umd.edu)

example, by applying quadrature to a kernel t of the form $t(\alpha, \beta, \gamma, \delta) = t(\gamma - \alpha, \delta - \beta)$, ordering the unknowns $\hat{f}(\alpha_j, \beta_l)$ first by increasing j, then by increasing l. In this case, the (i, j) component in the (k, l) block is given by $(T_{kl})_{ij} = t_{i-j}^{(k-l)}$ for $1 \leq i, j \leq p$, $1 \leq k, l \leq n$. We then say that T is a block Toeplitz, Toeplitz block (BTTB) matrix.

Given $\hat{g}$ and T in (1), the discrete inverse problem is to recover $\hat{f}$. However, the continuous problem is generally ill-posed in the sense that small changes in $\hat{g}$ cause arbitrarily large changes in $\hat{f}$. Consequently, the matrix T will be ill-conditioned. Recovery of $\hat{f}$ is then complicated by the fact that noise e is also present in the measured data. That is, we have measured g rather than $\hat{g}$, where

$$Tf = \hat{g} + e = g. \tag{2}$$

Given the ill-conditioning of T, the exact solution, f, to (2) is not a reasonable approximation to $\hat{f}$. We instead seek an approximate solution f by solving a nearby, more well-posed problem. This method of approximating $\hat{f}$ is called *regularization*. We use a preconditioned conjugate gradient algorithm to compute such a regularized solution. A discussion of the methods of direct and iterative regularization techniques can be found in [15].

Iterative methods like conjugate gradients can take advantage of the well-known fact that matrix-vector products involving BTTB matrices with n blocks of size p can be computed in $O(np(\lg p + \lg n))$ operations by embedding the matrix inside a $2pn \times 2pn$ block circulant matrix with circulant blocks [4]. Also, preconditioners for BTTB matrices which are block circulant (BC), circulant block (CB), or block circulant with circulant blocks (BCCB) have been found to be very efficient [4, 23, 1]. For example, if the preconditioner is determined to be block Toeplitz with circulant blocks (BTCB), applying the preconditioner can be reduced to solving p systems involving $n \times n$ Toeplitz matrices [4]. However, for indefinite and/or ill-conditioned systems, the $O(n \lg^2 n)$ and $O(n^2)$ factorization algorithms for Toeplitz matrices can be numerically unstable; these algorithms can require as many as $O(n^3)$ operations in order to maintain stability [25, 12, 7].

To overcome this difficulty, we make use of the fact that Toeplitz matrices are related to *Cauchy-like* matrices by fast orthogonal transformations [17, 9, 11]. The particular Cauchy-like matrices discussed in §2 permit fast matrix-vector multiplication. An advantage of Cauchy-like matrices is that their inverses are also Cauchy-like, unlike Toeplitz matrices whose inverses are not generally Toeplitz. In addition, modified complete pivoting can be incorporated in the LDU factorization of a Cauchy-like matrix for a total cost of only $O(n^2)$.

In the course of this paper we develop a block Cauchy-like preconditioner that can be used to filter noise and accelerate convergence of the conjugate gradient iteration to an approximate solution of (2) when T is BTTB. This preconditioner is the two-dimensional generalization of the preconditioner for Toeplitz matrices discussed in [22]. We begin with a discussion in §2 of Cauchy-like matrices and some of their important properties. We discuss the regularizing properties of conjugate gradients and our choice of preconditioner in §3. In §4 we show that our preconditioner has desirable properties. Computational issues are the focus of §5, where it is shown that each iteration can be completed in $O(pn(\lg p + \lg n))$ operations. Section 6 contains numerical results for several examples, and §7 presents conclusions and future work.

2. Transformation from Toeplitz to Cauchy-like structure. A matrix C having the form

$$C = \left(\frac{a_i^T b_j}{\omega_i - \theta_j} \right)_{1 \le i,j \le n} \quad (a_i, b_j \in \mathcal{C}^{\ell \times 1}; \omega_i, \theta_j \in \mathcal{C}; \omega_i \ne \theta_j) \tag{3}$$

is called a Cauchy-like, or generalized Cauchy, matrix. If $\ell = 1$ and $a_i b_j = 1$, then the matrix is said to be Cauchy. The matrix C can also be identified as the unique solution of the *displacement equation*

$$\Omega C - C\Theta = AB^T \tag{4}$$

where

$$\Omega = \text{diag}(\omega_1, \ldots, \omega_n), \Theta = \text{diag}(\theta_1, \ldots, \theta_n), A = \begin{pmatrix} a_1^T \\ \vdots \\ a_n^T \end{pmatrix}, B = \begin{pmatrix} b_1^T \\ \vdots \\ b_n^T \end{pmatrix}.$$

The matrices A and B are called the *generators* of C with respect to Ω and Θ, and $\ell \le n$ is called the *displacement rank*. Notice that only the $2n\ell + 2n$ non-zero entries of A, B, Ω, Θ need to be stored to completely specify the entries of the matrix. Fortunately, certain properties of Cauchy-like matrices insure that LU factorizations of Cauchy-like matrices may be computed using only the matrices Ω, Θ and the generators without ever forming the matrix C; see [9], for example.

One disadvantage of Toeplitz matrices is that permutations of Toeplitz matrices are not necessarily Toeplitz, so that incorporating pivoting into fast factorization schemes becomes difficult and expensive. However, because of (4), it is easy to show the following (see [17, 11], for example):

PROPERTY 1. *Row and column permutations of Cauchy-like matrices are Cauchy-like, as are leading principal submatrices.*

This property and the fact that Schur complements of Cauchy-like matrices are Cauchy-like [9] lead to fast algorithms for factoring Cauchy-like matrices which can pivot for stability [9, 11].

We use the algorithm developed by Gu [11] which performs a fast $O(\ell n^2)$ variation of LU decomposition with modified complete pivoting. Recall that in complete pivoting, at every elimination step one chooses the largest element in the current submatrix as the pivot in order to reduce element growth. Gu proposes instead that one find an entry sufficiently large in magnitude by considering the largest 2-norm column of one of the generators that remains to be factored at each step. This algorithm computes the pivoted LU factorization ($C = PLUQ$ where P and Q are permutation matrices) [11, Alg. 2] using only the generators, which are easy to determine and to update (see §5), and Gu shows that the algorithm can be efficient and numerically stable. Although the Cauchy-like matrices of interest to us are full, they have displacement rank $\ell = 1$ or 2, which makes them both efficient to store using relation (4) and fast to factor. For our purposes it was convenient to set $D = diag(u_{11}, \ldots, u_{nn})$ and $U \leftarrow D^{-1}U$ to obtain the equivalent factorization $C = PLDUQ$.

We also exploit the following property of Cauchy-like matrices [17].

PROPERTY 2. *The inverse of a Cauchy-like matrix is Cauchy-like:*

$$C^{-1} = -\left(\frac{x_i^T w_j}{\theta_i - \omega_j} \right)_{1 \le i,j \le n} \quad (x_i, w_j \in \mathcal{C}^{\ell \times 1}). \tag{5}$$

The generators X and W can be determined from the relations [17]

$$CX = A, \quad W^T C = B^T. \tag{6}$$

Thus, given the LU factorization of C, solving for X and W requires only $O(\ell n^2)$ operations and is stable when C is well-conditioned.

The third important property is that Toeplitz matrices also satisfy certain displacement equations [21, 9] which allow them to be transformed via fast Fourier transforms into Cauchy-like matrices [17, 9]:

PROPERTY 3. *Every Toeplitz matrix T satisfies an equation of the form*

$$R_1 T - T R_{-1} = AB^T \tag{7}$$

where $A \in \mathcal{C}^{n\times\ell}$, $B \in \mathcal{C}^{n\times\ell}$, $1 \le \ell \le 2$, and

$$R_\delta = \begin{pmatrix} 0 & 0 & \dots & 0 & \delta \\ 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}.$$

The Toeplitz matrix T is orthogonally related to a Cauchy-like matrix

$$C = FTS_0^* F^*$$

that satisfies the displacement equation

$$S_1 C - C S_{-1} = (FA)(B^T S_0^* F^*)\,, \tag{8}$$

where

$$S_1 = \mathrm{diag}(1, e^{\frac{2\pi i}{n}}, \dots, e^{\frac{2\pi i}{n}(n-1)}),$$

$$S_{-1} = \mathrm{diag}(e^{\frac{\pi i}{n}}, \dots, e^{\frac{(2n-1)\pi i}{n}}),$$

$$S_0 = \mathrm{diag}(1, e^{\frac{\pi i}{n}}, \dots, e^{\frac{\pi i}{n}(n-1)}),$$

and F is the normalized inverse discrete Fourier transform matrix defined by

$$F = \frac{1}{\sqrt{n}} \left[\exp\left(\frac{2\pi i}{n}(j-1)(k-1) \right) \right]_{1\le j,k \le n}.$$

We note that Toeplitz matrices are orthogonally related to Cauchy-like matrices through other fast transformations as well [11]. However, the particular relation in Property 3 can be exploited to determine a $O(n \lg n)$ stable algorithm for multiplication by the inverse of the Cauchy-like matrix [22].

Property 3 implies that if T is a Toeplitz block matrix, it satisfies

$$(I \otimes F)T(I \otimes S_0^* F^*) = C$$

where $\otimes$ denotes the Kronecker tensor product and C is Cauchy-like. Also, since each block of C is of the form $FT_{ij}S_0^*F^*$, where T_{ij} is the (i,j) block of T, the blocks of C are themselves Cauchy-like. In this case, however, the displacement rank ℓ of C is between p and $2p$. Thus, algorithms like Gu's algorithm which rely on C having small displacement rank will become expensive if p is large, requiring $O(p(np)^2)$ operations for a full factorization. Instead, we seek an approximation to T which, under an appropriate transformation, becomes block diagonal with Toeplitz blocks. Then its associated Cauchy-like matrix will be block diagonal with Cauchy-like blocks. In order to obtain a full factorization of the latter matrix, one need only apply Gu's algorithm to each individual block on the block diagonal. Following the discussion in §3.2, we observe that a full factorization of the approximation can be obtained in only $O(pn^2)$ operations.

3. Regularization and preconditioning. We could solve the linear system (2) exactly by transforming the BTTB matrix T to a Cauchy-like matrix as mentioned above and factoring. However, the solution we would compute in this manner would be hopelessly contaminated with noise, as we now discuss. The analysis will be based on the following four assumptions:

1. The matrix T has been normalized so that its largest singular value is of order 1.
2. The uncontaminated data vector $\hat{g}$ satisfies the discrete Picard condition; i.e., the spectral coefficients of $\hat{g}$ decay in absolute value faster than the singular values [27, 14].
3. The additive noise is zero-mean white Gaussian. In this case, the components of the error e are independent random variables normally distributed with mean zero and variance ϵ^2.
4. The *noise level*, $\frac{\|e\|_2}{\|\hat{g}\|_2}$, is strictly less than one.

Let $T = U\Sigma V^T$ be the singular value decomposition of T and let f be the exact solution to the noisy system

$$Tf = g = \hat{g} + e. \tag{9}$$

The spectral coefficients of the exact solution $\hat{g}$ and noise e are $\zeta = U^T\hat{g}$ and $\eta = U^T e$, respectively. For the remainder of the paper we will assume that $N = pn$ is the dimension of T. Using (9), we observe that

$$f = \sum_{i=1}^{N} \frac{\zeta_i + \eta_i}{\sigma_i} v_i, \tag{10}$$

where v_i denotes the ith column of V and σ_i denotes the ith diagonal element of the diagonal matrix Σ.

Under the white noise assumption, the coefficients η_i are roughly constant in size, while the discrete Picard condition tells us that the ζ_i go to zero at least as fast as the singular values σ_i. Thus, components for which ζ_i is of the same order as η_i are obscured by noise.

By assumptions 2 and 4, there exists $\bar{m} > 0$ such that for all $i > \bar{m}$, the ζ_i are indeed indistinguishable from the η_i. Further, there exists $0 < m^* \leq \bar{m}$ such that for $i > m^*$ it is never the case that $|\zeta_i| \gg |\eta_i|$. We therefore choose to partition the columns of V into bases for the *upper, lower,* and *transition* subspaces as follows. We say that the *upper* subspace is the space spanned by the first m^* columns of V.

Hence the upper subspace corresponds to the largest m^* singular values. The *lower* subspace is the space spanned by the last $N - \bar{m}$ columns for V; i.e. those columns of V corresponding to the smallest singular values. Finally, the *transition* subspace is the space spanned by the remaining $\bar{m} - m^*$ columns of V. Since these columns correspond to the mid-range singular values, the transition subspace is generally difficult to resolve unless there is a gap in the singular value spectrum.

Comparing the exact solution $\hat{f}$ of (1) to f in (10), we see that the greatest difference is in the magnitude of the components in the lower subspace. Thus we choose to use an iterative method called CGLS which at early iterations produces a regularized solution with small components in the lower subspace and which resembles $\hat{f}$ in the upper subspace. An appropriate preconditioner will speed convergence to this approximate solution without adding components in the lower subspace.

3.1. Regularization by preconditioned conjugate gradients. The standard conjugate gradient (CG) method [18] is an iterative method for solving systems of linear equations for which the matrix is symmetric positive definite. If the matrix is not symmetric positive definite, one can use the CGLS algorithm [18], a variant of standard CG that solves the normal equations in factored form. If the discrete Picard condition holds, then CGLS acts as an iterative regularization method with the iteration index taking the role of the regularization parameter [8, 13, 15]. The spread and clustering of the singular values govern the speed and convergence of the algorithm [26]. Preconditioning is therefore often applied in an effort to cluster the singular values and thus, to speed convergence.

According to (10), we desire that the preconditioner cluster only the large singular values for which $|\zeta_i| \gg |\eta_i|$. Unfortunately, the indices for which this holds are difficult, if not impossible, to determine in advance. However, as we show in §4, it is possible to choose a preconditioner that clusters most of the largest m^* singular values while leaving the small singular values, and with them, the lower subspace, relatively unchanged. In this case, the first few iterations of CGLS will quickly capture the solution lying within the upper subspace. Ideally, a modest number of subsequent iterations will provide some improvement over the transition subspace without significant contamination from the noise contained in the lower subspace.

3.2. The preconditioner. The given BTTB matrix T has the following block structure:

$$T = \begin{bmatrix} T_0 & T_{-1} & T_{-2} & \dots & T_{1-n} \\ T_1 & T_0 & T_{-1} & \dots & T_{2-n} \\ T_2 & T_1 & \ddots & \ddots & T_{3-n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ T_{n-1} & \dots & \dots & T_1 & T_0 \end{bmatrix},$$

where each T_i is Toeplitz; that is, $(T_i)_{kl} = t^{(i)}_{k-l}$. For each T_i, let us define H_i to be its T. Chan circulant approximation [6], so that the diagonals of H_i are given by

$$h^{(i)}_j = \begin{cases} \frac{(n-j)t^{(i)}_j + j t^{(i)}_{j-n}}{n} & 0 \le j < n \\ h^{(i)}_{n+j} & 0 < -j < n \end{cases}.$$

The matrix H_i is the closest circulant matrix in the Frobenius norm to T_i [6]. Finally, we define H to be the BTCB matrix

$$H = \begin{bmatrix} H_0 & H_{-1} & H_{-2} & \dots & H_{1-n} \\ H_1 & H_0 & H_{-1} & \dots & H_{2-n} \\ H_2 & H_1 & \ddots & \ddots & H_{3-n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ H_{n-1} & \dots & \dots & H_1 & H_0 \end{bmatrix}.$$

It was shown in [4] that H is the closest BTCB matrix to T in the Frobenius norm. The goal is to develop a preconditioner from an appropriately transformed version of the matrix H.

We define the matrices F and S_0 as in Property 3, with the dimension being either p or n as is appropriate in context. Since the matrices T and H are block Toeplitz, the matrices $(I \otimes F)T(I \otimes F^*)$ and $(I \otimes F)H(I \otimes F^*)$ with their (i, j) blocks given by $FT_{i-j}F^*$ and $FH_{i-j}F^*$, respectively, are also block Toeplitz.

Now since the H_i are circulant, they can be diagonalized by the matrix F [4]; therefore, for each (i, j), $FH_{i-j}F^*$ is diagonal. In §1 we assumed that the unknowns are ordered first in the increasing α direction, then in order of increasing β. Let $\hat{P}$ be the $N \times N$ permutation matrix which reorders the unknowns in the increasing β direction first. Then

$$\tilde{T} = \hat{P}(I \otimes F)T(I \otimes F^*)\hat{P}^T$$

is a block matrix with Toeplitz $n \times n$ blocks while

$$\tilde{H} = \hat{P}(I \otimes F)H(I \otimes F^*)\hat{P}^T$$

is a block diagonal matrix with $n \times n$ Toeplitz blocks.

Since $\tilde{T}$ has size n Toeplitz blocks, $\tilde{T}$ is related to a Cauchy-like matrix $\tilde{C}$ as mentioned at the end of §2:

$$\tilde{C} = (I \otimes F)\tilde{T}(I \otimes S_0^* F^*)$$

where F and S_0 now have dimension n. Each block of $\tilde{C}$ is Cauchy-like. Likewise, $\tilde{H}$ is related to a Cauchy-like matrix with Cauchy-like blocks:

$$\tilde{K} = (I \otimes F)\tilde{H}(I \otimes S_0^* F^*).$$

Since $\tilde{H}$ is block diagonal with Toeplitz blocks, $\tilde{K}$ is block diagonal with Cauchy-like blocks. Finally, we observe that solving $Tf = g$ must be equivalent to solving

$$\tilde{C}\tilde{f} = \tilde{g}$$

where $\tilde{f} = (I \otimes FS_0)\hat{P}(I \otimes F)f$, $\tilde{g} = (I \otimes F)\hat{P}(I \otimes F)g$.

As mentioned before, since $\tilde{C}$ is Cauchy-like, we could apply Gu's factorization algorithm directly to it; however, the cost of a full factorization would be $O(p(np)^2)$ operations. Fortunately, $\tilde{K}$, our approximation of $\tilde{C}$, is block diagonal. Since each of the p blocks $\tilde{K}_{ii}$ is an $n \times n$ Cauchy-like matrix of displacement rank 2, to completely factor $\tilde{K}$ requires only $O(pn^2)$ operations.

A factorization of $\tilde{K}_{ii}$ using a modified complete pivoting strategy may lead to an interchange of rows (specified by a permutation matrix P_i) and columns (specified by a permutation matrix Q_i). Let $P = diag(P_1, \ldots, P_p)$ and $Q = diag(Q_1, \ldots, Q_p)$. We will use an appropriate piece of the matrix $P^T \tilde{K} Q^T$, to be defined shortly, to precondition the matrix $P^T \tilde{C} Q^T$. First we summarize the sequence of transformations which leads to the development of the preconditioner:

1. Transform the matrices T and H to the Toeplitz block matrices $\tilde{T}$ and $\tilde{H}$:

$$\begin{array}{rcl} \tilde{T} &=& \hat{P}(I \otimes F) T (I \otimes F^*) \hat{P}^T \\ \tilde{H} &=& \hat{P}(I \otimes F) H (I \otimes F^*) \hat{P}^T \end{array}.$$

   Note that $\tilde{H}$ is also block diagonal.

2. Transform the matrices $\tilde{T}$ and $\tilde{H}$ to Cauchy-like matrices with Cauchy-like blocks $\tilde{C}_{ij}, \tilde{K}_{ij}$, respectively:

$$\begin{array}{rcl} \tilde{C} &=& (I \otimes F) \tilde{T} (I \otimes S_0^* F^*) \\ \tilde{K} &=& (I \otimes F) \tilde{H} (I \otimes S_0^* F^*) \end{array}.$$

3. Permute the matrices $\tilde{C}$ and $\tilde{K}$ using the block diagonal permutation matrices P and Q:

$$\begin{array}{rcl} C &=& P^T \tilde{C} Q^T \\ K &=& P^T \tilde{K} Q^T \end{array}.$$

Note that since all the transformations are accomplished with unitary matrices, C and T have the same singular values, as do K and H.

Hence, setting $y = Q\tilde{f}$, and $z = P^T \tilde{g}$, the problem we wish to solve is

$$Cy = z. \tag{11}$$

We choose a left preconditioner M, determined from K, so that

$$M^{-1} C y = M^{-1} z$$

and use CGLS to solve the corresponding normal equations

$$(M^{-1}C)^*(M^{-1}C) y = (M^{-1}C)^* M^{-1} z. \tag{12}$$

Recall from §3.1 that we wish to design a preconditioner that clusters the largest m^* singular values while leaving the the small singular values unchanged. Notice also that the singular values of K, our approximation to C, are simply the union of the singular values of the $K_{ii} = P_i^T \tilde{K}_{ii} Q_i^T$. Let Γ be set of the largest m^* singular values of K. Then precisely m_i singular values of K_{ii} are in Γ, with $m^* = \sum_{i=1}^p m_i$. As a result of pivoting during Gu's factorization algorithm, the $m_i \times m_i$ leading submatrix of K_{ii} corresponds to the well-conditioned part of K while the rest contributes to the ill-conditioned part. Let $K_{ii} = L_{ii} D_{ii} U_{ii}$ and write this equation in block form, where the upper left blocks are $m_i \times m_i$:

$$\begin{bmatrix} K_{ii}^{(1)} & K_{ii}^{(2)} \\ K_{ii}^{(3)} & K_{ii}^{(4)} \end{bmatrix} = \begin{bmatrix} L_{ii}^{(1)} & 0 \\ L_{ii}^{(2)} & L_{ii}^{(3)} \end{bmatrix} \begin{bmatrix} D_{ii}^{(1)} & 0 \\ 0 & D_{ii}^{(2)} \end{bmatrix} \begin{bmatrix} U_{ii}^{(1)} & U_{ii}^{(2)} \\ 0 & U_{ii}^{(3)} \end{bmatrix}. \tag{13}$$

Here $L_{ii}^{(1)}, L_{ii}^{(3)}$ are lower triangular, $U_{ii}^{(1)}, U_{ii}^{(3)}$ are upper triangular, and $D_{ii}^{(1)}$ and $D_{ii}^{(2)}$ are diagonal. Then we define

$$M_i = \begin{bmatrix} L_{ii}^{(1)} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} D_{ii}^{(1)} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} U_{ii}^{(1)} & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} K_{ii}^{(1)} & 0 \\ 0 & I \end{bmatrix}.$$

Finally, we choose as our preconditioner the matrix

$$M = diag(M_1, \ldots, M_p).$$

Since leading principal submatrices of Cauchy-like matrices are Cauchy-like, M is a block diagonal matrix with Cauchy-like blocks each augmented by an identity.

Let us compare our preconditioning scheme with the preconditioning method given in [13] for the BTTB matrices of discrete ill-posed problems. In [13] the preconditioner is determined by forming the T. Chan BCCB approximant to T, computing its eigenvalues via 2-D fast Fourier transforms, and then replacing all the eigenvalues below a tolerance with ones. Therefore, our method is similar to their BCCB based preconditioner in that we also rely on a rank revealing factorization to determine the appropriate cutoff which is used to form the preconditioner. We choose our cutoff tolerance in a manner similar to that given in [13]. However, our preconditioner is formed from a BTCB approximant to T, which requires approximating T only on one level unlike the BCCB approximant which requires approximating T on two levels.

The most notable difference is that is we rely on a transformation to Cauchy-like matrices; therefore we may use a fast pivoted factorization scheme, rather than 2-D Fourier transforms, to generate the necessary rank revealing information. While the preconditioner in [13] requires $O(pn(\lg p + \lg n))$ operations to precompute, our preconditioner requires, in the worst case, $O(pb_w \lg p + pn \lg n + \sum_{i=1}^{p} m_i^2)$ operations to precompute, where b_w denotes the maximum block bandwidth of the matrix. However, in applications the block bandwidth is sometimes small compared to n and when the blocks of T are symmetric, the number of operations required to initialize our preconditioner can be reduced to $O(pn \lg n + b_w p \lg p + m_s)$ where $m_s = m_1^2 + m_{p/2}^2 + \sum_{i=1}^{p/2} m_i^2$ when p is even. In some cases (when the dimension of the upper subspace is small, for example) we have observed m^* is small relative to $pn \lg n$, which implies our preconditioner can be just as cheap to precompute. Our preconditioner is competitive with the BCCB matrix in that it is stable to compute and can be applied in at most $O(pn \lg n)$, rather than $O(pn(\lg n + \lg p))$, operations. In the next section, we show that our preconditioner is just as effective as the one in [13] in clustering the large singular values around one. Further, we show that the small singular values remain small and that the upper and lower subspaces remain unmixed.

4. Properties of the preconditioner. In this section we give theoretical results which show how successful our preconditioner is in filtering noise and accelerating convergence to a regularized solution.

4.1. Clustering. Under the assumptions in §3 for an ill-conditioned matrix C, in order for the first few iterations of CGLS to capture the solution corresponding to the largest m^* singular values, the preconditioner must cluster the majority of the m^* singular values while leaving the small singular values and lower subspace essentially unchanged. We show that the question of how well our preconditioner M clusters the singular values can be reduced to the question of how well K approximates C, or equivalently, how well H approximates T.

We argue as follows. We first note that to show that the largest m^* singular values of $M^{-1}C$ cluster around one, it suffices to show that the smallest m^* singular values of $I - M^{-1}C$ cluster around zero. We denote the k-th largest singular value of a matrix Z by $\sigma_k(Z)$, and the k-th largest eigenvalue by $\lambda_k(Z)$.

Let $K - C = R$. Now $K = M + S$, where S is block diagonal with blocks

$$S_i = \left[\begin{array}{cc} 0 & K_{ii}^{(2)} \\ K_{ii}^{(3)} & K_{ii}^{(4)} - I \end{array} \right].$$

Thus, $M - C = R - S$. We therefore obtain the equality

$$I - M^{-1}C = M^{-1}(R - S) = M^{-1}(K - C) - M^{-1}S. \tag{14}$$

Now let

$$Y_i = \left[\begin{array}{cc} 0 & 0 \\ K_{ii}^{(3)} & K_{ii}^{(4)} \end{array} \right] \text{ and } Z_i = \left[\begin{array}{cc} 0 & K_{ii}^{(1)-1} K_{ii}^{(2)} \\ 0 & -I \end{array} \right].$$

Define E_S and E_M to be the block diagonal matrices

$$E_S = diag(Y_1, \ldots, Y_p) \text{ and } E_M = diag(Z_1, \ldots, Z_p).$$

Then $M^{-1}S = E_S + E_M$, where E_S and E_M each have rank $N - m^*$. From Theorem 3.3.16 of [19] ,

$$\sigma_{k+N-m^*}(M^{-1}S) \leq \sigma_k(E_S) \quad k = 1, \ldots, m^*.$$

Applying the same theorem to Equation (14) with $2 \leq i+j \leq N+1$ for $N - m^* + 1 \leq j \leq N$ we have

$$\begin{array}{rcl} \sigma_{i+j-1}(I - M^{-1}C) & \leq & \sigma_i(M^{-1}(K - C)) + \sigma_j(M^{-1}S) \\ & \leq & \sigma_1(M^{-1})\sigma_i(K - C) + \sigma_j(M^{-1}S) \\ & \leq & \sigma_1(M^{-1})\sigma_i(H - T) + \sigma_{j+m^*-N}(E_S). \end{array}$$

In particular,

$$\sigma_{i+N-m^*}(I - M^{-1}C) \leq \frac{\sigma_i(H - T)}{\sigma_N(M)} + \sigma_1(E_S) \quad i = 1, \ldots, m^*. \tag{15}$$

Hence, under the assumptions that the preconditioner is well-conditioned and that the matrix E_S has sufficiently small elements, the clustering of the singular values of $I - M^{-1}C$ around zero depends on the clustering of the first m^* singular values of $H - T$ around zero. We now discuss two special cases for which $H - T$ has singular values clustered around zero. First we will need the following lemma from [4].

LEMMA 4.1 (R. CHAN AND Q. JIN). *Assume that the BTTB matrix T is symmetric. Let the entries of block T_i be denoted $t_{jk}^{(i)} = t_{|j-k|}^{(i)}$ for $1 \leq j, k \leq p$, $1 \leq i \leq n$. Assume that the generating sequence of T is absolutely summable, i.e.,*

$$\sum_{i=0}^{\infty} \sum_{j=0}^{\infty} |t_j^{(i)}| \leq J < \infty.$$

Then for all $\epsilon > 0$, there exists a $k^* > 0$ such that for all $p > k^*$ and $n > 0$, at most $O(n)$ eigenvalues of $H - T$ have absolute values exceeding ϵ.

Since $H - T$ is symmetric for symmetric BTTB matrices T, combining Equation (15) with Lemma 4.1, we obtain the following.

THEOREM 4.2. *Assume T is a symmetric BTTB matrix with an absolutely summable generating sequence. Then for all $\epsilon > 0$, there exists a $k^* > 0$ such that for all $p > k^*$ and $n > 0$, at most $O(m^* - n)$ singular values of $I - M^{-1}C$ exceed $\frac{\epsilon}{\sigma_N(M)} + \sigma_1(E_S)$.*

Let us consider another special case for which we are guaranteed clustering. Let $C_{2\pi}$ denote the Banach space of all 2π-periodic, continuous, complex-valued functions equipped with the norm $\|\cdot\|_\infty$. This class of functions contains the Wiener class [2]. For all $h \in C_{2\pi}$, let the Fourier coefficients of h be defined by

$$\bar{t}_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} h(\theta)e^{ik\theta}d\theta, \quad k = 0, \pm 1, \pm 2, \cdots,$$

where $i = \sqrt{-1}$. Let $\bar{T}$ be the $p \times p$ complex Toeplitz matrix whose diagonals are given by $\bar{t}_k$, and let $\bar{H}$ be its T. Chan circulant approximation. Finally, let the BTTB matrix T be given as $T = \bar{R} \otimes \bar{T}$, where $\bar{R}$ is a non-singular $n \times n$ matrix. A lemma proved by R. Chan and M. Yeung [3] will be useful.

LEMMA 4.3 (R. CHAN AND M. YEUNG). *Let $h \in C_{2\pi}$. Then for all $\epsilon > 0$, there exist k^* and $j^* > 0$, such that for all $p > k^*$,*

$$\bar{T} - \bar{H} = \bar{U} + \bar{V}$$

where

$$rank(\bar{U}) \le j^*$$

and

$$\|\bar{V}\|_2 \le \epsilon.$$

Applying this lemma to T we obtain the following result.

LEMMA 4.4. *Given $\epsilon > 0$, let k^*, j^*, $\bar{U}$, and $\bar{V}$ be defined as in Lemma 4.3 with $h \in C_{2\pi}$. Then*

$$\sigma_i(T - H) \le \|\bar{R}\|_2\epsilon, \quad N - nj^* + 1 \le i \le N.$$

Proof: Using [19, lemmas 3.3.16 and 4.2.15],

$$\sigma_i(T - H) \le \sigma_i(\bar{R} \otimes \bar{U}) + \sigma_1(\bar{R} \otimes \bar{V}) \le \sigma_i(\bar{R} \otimes \bar{U}) + \|\bar{R}\|_2\epsilon, \quad i = 1, \ldots, N.$$

However, since $\bar{U}$ has rank j^*, the rank of $\bar{R} \otimes \bar{U}$ is $N - nj^*$, so that

$$\sigma_i(T - H) \le \|\bar{R}\|_2\epsilon, \quad N - nj^* + 1 \le i \le N. \quad \square$$

We use Lemma 4.4 and Equation (15) to deduce the following.

THEOREM 4.5. *Let the BTTB matrix T be defined as $T = \bar{R} \otimes \bar{T}$ for a given $n \times n$ nonsingular matrix $\bar{R}$, and let the entries of the $p \times p$ matrix $\bar{T}$ be given by $\bar{t}_{k-j}$ defined above with $h \in C_{2\pi}$. Then for all $\epsilon > 0$, there exist k^* and $j^* > 0$, such that for all $p > k^*$, at most $m^* - nj^*$ singular values of $I - M^{-1}C$ exceed $\frac{\|\bar{R}\|_2\epsilon}{\sigma_N(M)} + \sigma_1(E_S)$.*

In both the aforementioned cases, assuming the values m_i were chosen appropriately, the preconditioner will cluster most of the m^* singular values of the preconditioned matrix around one when a sufficient number of the singular values of $T - H$ are small. As these special cases illustrate, a proof that our preconditioner is effective at clustering the large singular values is reduced to a proof that many of the singular values of $T - H$ are small for the given BTTB matrix T.

4.2. Unmixing results. Recall that the transformation from the problem involving T to one involving C was accomplished using a sequence of orthogonal transforms. Thus, the singular values of T and C are the same, as is our definition of the upper, lower, and transition spaces in §3. That is, we have changed the bases for the respective spaces, but we have not mixed them.

For the approximate solution generated by CGLS in early iterates to be essentially unaffected by noisy components in the lower subspace, we require that the preconditioner not mix the upper and lower subspaces. The following theorem tells the extent to which preconditioning by M mixes these subspaces.

THEOREM 4.6. *Let k be the dimension of the subspace corresponding to the smallest k singular values and let*

$$\begin{aligned} C &= [Q_1\, Q_2\, Q_3] \begin{bmatrix} \Sigma_1 & 0 & 0 \\ 0 & \Sigma_2 & 0 \\ 0 & 0 & \Sigma_3 \end{bmatrix} \begin{bmatrix} V_1^* \\ V_2^* \\ V_3^* \end{bmatrix}, \\ M^{-1}C &= \left[\hat{Q}_1\, \hat{Q}_2\, \hat{Q}_3\right] \begin{bmatrix} \hat{\Sigma}_1 & 0 & 0 \\ 0 & \hat{\Sigma}_2 & 0 \\ 0 & 0 & \hat{\Sigma}_3 \end{bmatrix} \begin{bmatrix} \hat{V}_1^* \\ \hat{V}_2^* \\ \hat{V}_3^* \end{bmatrix} \end{aligned}$$

be singular value decompositions with $V_3, \hat{V}_3 \in \mathcal{C}^{N\times k}$ and $V_1, \hat{V}_1 \in \mathcal{C}^{N\times m^*}$. Then

$$\|V_1^*\hat{V}_3\|_2 \le \frac{\hat{\sigma}_{N-k+1}}{\sigma_{m^*}}(\max\{1, \max_i \|K_{ii}^{(1)}\|_2\}).$$

Proof: Using the decompositions we have

$$\begin{aligned} V_1^*\hat{V}_3 &= (V_1^*C^{-1})M(M^{-1}C\hat{V}_3) \\ &= \Sigma_1^{-1}Q_1^*M\hat{Q}_3\hat{\Sigma}_3. \end{aligned}$$

Since Q_1^* has orthonormal columns, as does $\hat{Q}_3$, it follows that

$$\|V_1^*\hat{V}_3\|_2 \le \frac{\hat{\sigma}_{N-k+1}}{\sigma_{m^*}}\|M\|_2 = \frac{\hat{\sigma}_{N-k+1}}{\sigma_{m^*}}(\max\{1, \max_i \|K_{ii}^{(1)}\|_2\}). \ \square$$

We note that if the preconditioner developed in [13] for their right preconditioning scheme is applied to the left rather than the right, a similar result can be obtained.

Next we show that $\hat{\sigma}_j \approx \sigma_j$ for σ_j corresponding to the last $N - m^*$ singular values, and thus $\hat{\sigma}_{N-k+1}$ is small. Hence, if M is well-conditioned, we are guaranteed that the upper and lower subspaces remain unmixed.

For given values of m_i, we first rewrite C in block form:

$$C = \begin{bmatrix} C_{11}^{(1)} & C_{11}^{(2)} & C_{12}^{(1)} & C_{12}^{(2)} & \dots & C_{1n}^{(2)} \\ C_{11}^{(3)} & C_{11}^{(4)} & C_{12}^{(3)} & C_{12}^{(4)} & \dots & C_{1n}^{(4)} \\ C_{21}^{(1)} & C_{21}^{(2)} & C_{22}^{(1)} & C_{22}^{(2)} & \dots & C_{2n}^{(2)} \\ C_{21}^{(3)} & C_{21}^{(4)} & C_{22}^{(3)} & C_{22}^{(4)} & \dots & C_{2n}^{(4)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ C_{n1}^{(1)} & C_{n1}^{(2)} & C_{n2}^{(1)} & C_{n2}^{(2)} & \dots & C_{nn}^{(2)} \\ C_{n1}^{(3)} & C_{n1}^{(4)} & C_{n2}^{(3)} & C_{n2}^{(4)} & \dots & C_{nn}^{(4)} \end{bmatrix}$$

where $C_{ii}^{(1)}$ is $m_i \times m_i$ and $C_{ii}^{(4)}$ is $n - m_i \times n - m_i$. Likewise, we rewrite $M^{-1}C$

$$M^{-1}C = \begin{bmatrix} K_{11}^{(1)-1}C_{11}^{(1)} & K_{11}^{(1)-1}C_{11}^{(2)} & K_{11}^{(1)-1}C_{12}^{(1)} & K_{11}^{(1)-1}C_{12}^{(2)} & \dots & K_{11}^{(1)-1}C_{1n}^{(2)} \\ C_{11}^{(3)} & C_{11}^{(4)} & C_{12}^{(3)} & C_{12}^{(4)} & \dots & C_{1n}^{(4)} \\ K_{22}^{(1)-1}C_{12}^{(1)} & K_{22}^{(1)-1}C_{12}^{(2)} & K_{22}^{(1)-1}C_{22}^{(1)} & K_{22}^{(1)-1}C_{12}^{(2)} & \dots & K_{22}^{(1)-1}C_{2n}^{(2)} \\ C_{21}^{(3)} & C_{21}^{(4)} & C_{22}^{(3)} & C_{22}^{(4)} & \dots & C_{2n}^{(4)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ K_{nn}^{(1)-1}C_{n1}^{(1)} & K_{nn}^{(1)-1}C_{n1}^{(2)} & K_{nn}^{(1)-1}C_{n2}^{(1)} & K_{nn}^{(1)-1}C_{n2}^{(2)} & \dots & K_{nn}^{(1)-1}C_{nn}^{(2)} \\ C_{n1}^{(3)} & C_{n1}^{(4)} & C_{n2}^{(3)} & C_{n2}^{(4)} & \dots & C_{nn}^{(4)} \end{bmatrix}.$$

Let the rank m^* matrices E_{M_1}, E_{C_1} be defined from the odd row-blocks of C and $M^{-1}C$:

$$E_{C_1} = \begin{bmatrix} C_{11}^{(1)} & C_{11}^{(2)} & C_{12}^{(1)} & C_{12}^{(2)} & \dots & C_{1n}^{(2)} \\ C_{21}^{(1)} & C_{21}^{(2)} & C_{22}^{(1)} & C_{22}^{(2)} & \dots & C_{2n}^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ C_{n1}^{(1)} & C_{n1}^{(2)} & C_{n2}^{(1)} & C_{n2}^{(2)} & \dots & C_{nn}^{(2)} \end{bmatrix},$$

$$E_{M_1} = \begin{bmatrix} K_{11}^{(1)-1}C_{11}^{(1)} & K_{11}^{(1)-1}C_{11}^{(2)} & K_{11}^{(1)-1}C_{12}^{(1)} & K_{11}^{(1)-1}C_{12}^{(2)} & \dots & K_{11}^{(1)-1}C_{1n}^{(2)} \\ K_{22}^{(1)-1}C_{12}^{(1)} & K_{22}^{(1)-1}C_{12}^{(2)} & K_{22}^{(1)-1}C_{22}^{(1)} & K_{22}^{(1)-1}C_{12}^{(2)} & \dots & K_{22}^{(1)-1}C_{2n}^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ K_{nn}^{(1)-1}C_{n1}^{(1)} & K_{nn}^{(1)-1}C_{n1}^{(2)} & K_{nn}^{(1)-1}C_{n2}^{(1)} & K_{nn}^{(1)-1}C_{n2}^{(2)} & \dots & K_{nn}^{(1)-1}C_{nn}^{(2)} \end{bmatrix},$$

and let E_1 be the rank $N - m^*$ matrix defined from the even row-blocks of C and $M^{-1}C$:

$$E_1 = \begin{bmatrix} C_{11}^{(3)} & C_{11}^{(4)} & C_{12}^{(3)} & C_{12}^{(4)} & \dots & C_{1n}^{(4)} \\ C_{21}^{(3)} & C_{21}^{(4)} & C_{22}^{(3)} & C_{22}^{(4)} & \dots & C_{2n}^{(4)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ C_{n1}^{(3)} & C_{n1}^{(4)} & C_{n2}^{(3)} & C_{n2}^{(4)} & \dots & C_{nn}^{(4)} \end{bmatrix}.$$

Under this partitioning, it easy to verify the relations

$$\begin{aligned} (M^{-1}C)^*(M^{-1}C) &= E_M + E \\ C^*C &= E_C + E \end{aligned} \tag{16}$$

where $E_M = E_{M_1}^* E_{M_1}$, $E_C = E_{C_1}^* E_{C_1}$, and $E = E_1^* E_1$. Consequently, we obtain the following:

THEOREM 4.7. *The $(m^* + i)$th singular value of each of the matrices C and $M^{-1}C$ lies in the interval $[0, \sigma_i(E)]$, for $i = 1, \dots, N - m^*$.*

Proof: Since the matrices in (16) are all Hermitian, we may apply Corollary IV.4.9 and problem 4, page 211, of [24] to obtain

$$\lambda_N(E) + \lambda_{m^*+i}(E_M) \le \lambda_{m^*+i}((M^{-1}C)^*(M^{-1}C)) \le \lambda_{m^*+1}(E_M) + \lambda_i(E)$$

and

$$\lambda_N(E) + \lambda_{m^*+i}(E_C) \le \lambda_{m^*+i}(C^*C) \le \lambda_{m^*+1}(E_C) + \lambda_i(E).$$

However, $\lambda_N(E) = 0$ and $\lambda_{m^*+1}(E_M) = 0 = \lambda_{m^*+1}(E_C)$, and thus

$$0 \le \lambda_{m^*+i}((M^{-1}C)^*(M^{-1}C)) \le \lambda_{m^*+1}(E_M) + \lambda_i(E) = \lambda_i(E)$$

and

$$0 \le \lambda_{m^*+i}(C^*C) \le \lambda_{m^*+1}(E_C) + \lambda_i(E) = \lambda_i(E).$$

The proof is completed by taking square roots. □

4.3. Properties of the factorization. The theorems in §4.1 and §4.2 show that the preconditioner will be effective under two restrictions. First M, and hence each $K_{ii}^{(1)}$, must be well conditioned. Second, the entries in E and E_S are required to be small. We now discuss to what extent these conditions hold for integral equation discretizations. We begin by showing how the entries of $\tilde{K}$ are computed from the elements of H.

Since the entries of $\tilde{K}$ can be written in terms of the generators of each block, it is necessary to discuss how these generators are obtained. Because each block of H is circulant, the non-zero entries of $(I \otimes F)H(I \otimes F^*)$, which lie on the diagonals of each of its blocks, are the eigenvalues of each block of H. Let $\lambda_k^{(l)}$ for $1 \le k \le p$, $1 - n \le l \le n - 1$ denote the kth eigenvalue of block H_l. It is well known that the eigenvalues of a $p \times p$ circulant matrix can be computed by means of an FFT in $O(p \lg p)$ operations. Since there are at most $2n$ distinct H_l, all the $\lambda_k^{(l)}$ can be computed in at most $O(np \lg p)$ operations. The matrix $\hat{P}$ permutes these eigenvalues so that $\tilde{H} = \hat{P}(I \otimes F)H(I \otimes F^*)\hat{P}^T$ is a block diagonal matrix with p, $n \times n$ Toeplitz matrices $\tilde{H}_i$ on its diagonal. The diagonals of $\tilde{H}_i$ are given by $\tilde{t}_j^{(i)} = \lambda_i^j$, $1 - n \le j \le n - 1$, $i = 1, \ldots, p$.

Since $\tilde{H}_i$ is Toeplitz, it satisfies (7) with $A = A_i$ and $B = B_i$. Examination of Equation (7) shows that the entries of the $n \times l$ matrices A_i and B_i are easily determined. The first column of A_i is the first unit vector, and the second column is given by

$$(17) \quad [0, \tilde{t}_{n-1}^{(i)}, \tilde{t}_{n-2}^{(i)}, \ldots, \tilde{t}_{p-1}^{(i)}, \ldots, \tilde{t}_1^{(i)}]^T + [\tilde{t}_0^{(i)}, \tilde{t}_{-1}^{(i)}, \tilde{t}_{-2}^{(i)}, \ldots, \tilde{t}_{-(q-1)}^{(i)}, \ldots, \tilde{t}_{-(n-1)}^{(i)}]^T$$

The first column of B_i is

$$(18) [\tilde{t}_{-(n-1)}^{(i)}, \tilde{t}_{-(n-2)}^{(i)}, \ldots, \tilde{t}_{-(q-1)}^{(i)}, \ldots, \tilde{t}_{-1}^{(i)}, \tilde{t}_0^{(i)}]^T - [\tilde{t}_1^{(i)}, \tilde{t}_2^{(i)}, \ldots, \tilde{t}_{p-1}^{(i)}, \ldots, \tilde{t}_{n-1}^{(i)}, 0]^T$$

and the second column is the last unit vector. The generators for $\tilde{K}_{ii}$ are then $\tilde{A}_i \equiv FA_i$ and $\tilde{B}_i \equiv conj(FS_0)B_i$, where $conj(\cdot)$ denotes complex conjugation, with F and S_0 as described in Property 3. Since $\tilde{A}_i$ and $\tilde{B}_i$ can be computed by means of the inverse fast Fourier transform of size n, computing all p generator pairs requires a total of $O(np \lg n)$ operations.

Now the absolute value of the (k, j) entry of $\tilde{K}_{ll}$ is given by

$$|\tilde{K}_{ll}|_{kj} = \frac{|\tilde{a}_k^{(l)T} \tilde{b}_j^{(l)}|}{|\omega_k - \theta_j|}$$

where ω_k and θ_j are the kth and jth diagonal entries of S_1 and S_{-1} defined in Property 3, respectively, and $\tilde{a}_k^{(l)T}$ and $\tilde{b}_j^{(l)}$ denote the kth row of $\tilde{A}_l$ and the jth row of $\tilde{B}_l$, respectively. Following the discussion in [22],

$$\frac{|\tilde{a}_k^{(l)T}\tilde{b}_j^{(l)}|}{|\omega_k - \theta_j|} < 10|\tilde{a}_k^{(l)T}\tilde{b}_j^{(l)}|$$

away from the corners and the diagonal of $\tilde{K}_{ll}$.

By direct computation it can be shown that

$$|\tilde{a}_k^{(l)T}\tilde{b}_j^{(l)}| = \frac{1}{\sqrt{n}}|conj(\xi_j^{(l)}) + e^{\frac{n-1}{n}\pi i(1-2j)}\nu_k^{(l)}|, \quad i = \sqrt{-1}$$

where $\nu_k^{(l)}$ is the kth entry in the second column of $\tilde{A}_l$ and $\xi_j^{(l)}$ is the jth entry in the first column of $FS_0conj(B_l)$. Therefore it is the normalized inverse Fourier coefficients of the second column of A_l and the first column of $S_0conj(B_l)$ that determine the magnitude of $|\tilde{a}_k^{(l)T}\tilde{b}_j^{(l)}|$.

However, since H_j is the T. Chan circulant approximant of T_j, we have [5]

$$\lambda_l^{(j)} = \sum_{s=-n+1}^{n-1} t_s^{(j)}(1 - \frac{|s|}{n})e^{\frac{2\pi i}{n}sl}, \quad i = \sqrt{-1}.$$

Since $\tilde{t}_j^{(l)} = \lambda_l^{(j)}$ we use the above substitution in Equations (17) and (18). Defining $t_s^{(n)} = 0$, we obtain

$$|\nu_k^{(l)}| \leq \frac{1}{\sqrt{n}} \sum_{s=-n+1}^{n-1} |(1 - \frac{|s|}{n})||v_k^{(s)}|, \text{ where}$$

$$v_k^{(s)} = \sum_{j=0}^{n-1}(t_s^{(-j)} + t_s^{(n-j)})e^{\frac{2\pi i}{n}jk}$$

and

$$|\xi_k^{(l)}| \leq \frac{1}{\sqrt{n}} \sum_{s=-n+1}^{n-1} |(1 - \frac{|s|}{n})||u_k^{(s)}|, \text{ where}$$

$$u_k^{(s)} = \sum_{j=0}^{n-1} e^{\frac{\pi i j}{n}}(t_s^{(-j)} + t_s^{(n-j)})e^{\frac{2\pi i}{n}jk}.$$

But $v_k^{(s)}$ is just the kth inverse Fourier coefficient of a vector having entries $(t_s^{(-j)} + t_s^{(n-j)})$, $j = 0, \ldots, n-1$. Likewise, $u_k^{(s)}$ is just the kth inverse Fourier coefficient of a vector having entries $e^{\frac{\pi i j}{n}}(t_s^{(-j)} + t_s^{(n-j)})$, $j = 0, \ldots, n-1$.

Therefore, if the kernel t in the integral equation is smooth, for every s $v_k^{(s)}$ and $u_k^{(s)}$ will be large only for small indices k. Hence,

$$\frac{|\tilde{a}_k^{(l)T}\tilde{b}_j^{(l)}|}{|\omega_k - \theta_j|} < \frac{10}{\sqrt{n}}(|\nu_k| + |\xi_j|) \ll 1$$

away from the corners of $\tilde{K}_{ll}$.

Thus, by relating the entries of $\tilde{K}_{ii}$ back to the entries in T as shown above, we discover the following property:

PROPERTY 4. *Suppose T is the BTTB matrix which results from the discretization of a smooth kernel t, normalized so the maximum element is one. For $l = 1, \ldots, p$, let $\tilde{K}_{ll} = F\tilde{H}_l S_0^* F^*$ where $\tilde{H}_l$ is a $n \times n$ Toeplitz matrix from the block diagonal of the matrix $\tilde{H}$ defined in §3.2. Then for n sufficiently large, there exists an $\epsilon \ll 1$ and $m_l \ll n$ such that all the elements of $\tilde{K}_{ll}$ are less than ϵ in magnitude except for those located in the four corner blocks of total dimension $m_l \times m_l$.*

Consequently, each Cauchy-like block of $\tilde{K}$ can be permuted to contain the large elements in its upper left block. We have observed that when Gu's algorithm is applied to $\tilde{K}_{ll}$ having the structure described in Property 4, his pivoting strategy is such that $K_{ll}^{(1)}$ will contain the four corner blocks. In fact, any pivoting strategy that yields this type of permutation will give a reasonable preconditioner for our scheme. We refer the interested reader to [11] for details of Gu's modified complete pivoting strategy. The key fact is that his algorithm makes its pivoting decision based on the size of the elements in the generator corresponding to the block that remains to be factored.

Thus, the components of the matrix E_S in §4.1 are small, and therefore our preconditioner has the property that the largest singular values of the preconditioned matrix are clustered, provided that the singular values of $K - C$ are small. But if the singular values of $K - C$ are small, then the matrix E in §4.2 necessarily has small singular values. Hence, the invariant subspace corresponding to the small singular values of C is not much perturbed by preconditioning. We therefore expect that the initial iterations of CGLS applied to the preconditioned system will produce a solution that is a good approximation to the noise free solution $\hat{f}$.

5. Algorithmic issues.

Our algorithm is as follows:

Algorithm 1: Solving $Tf = g$

1. Compute the generators for each submatrix K_{ii} (see §5.3).
2. For each i, determine the size m_i of the partial factorization and factor $K_{ii} = P_i L_{ii} D_{ii} U_{ii} Q_i$.
3. Set $P = diag(P_1, \ldots, P_p)$, $Q = diag(Q_1, \ldots, Q_p)$, $z = P^T F \tilde{g}$.
4. For $i = 1 \ldots n$, determine the generators of the $m_i \times m_i$ leading principal submatrix, $K_{ii}^{(1)}$ of K_{ii} and let $M_i = \begin{bmatrix} K_{ii}^{(1)} & 0 \\ 0 & I \end{bmatrix}$. (See §4.3.)
5. Set $M = diag(M_1, \ldots, M_p)$ and compute M^{-1} as in §5.3.
6. Compute an approximate solution $\tilde{y}$ to $M^{-1}Cy = M^{-1}z$ using a few steps of CGLS where matrix vector products involving $C = P^T \tilde{C} Q^T$ are formed without forming C itself (see §5.2).
7. The approximate solution in the original coordinate system is $f = S_0^* F^* Q^T y$.

A few comments about the algorithm are in order. First, the submatrices $K_{ii}^{(1)}$ and the matrix M are never actually formed; with only the easily determined generators of $K_{ii}^{(1)}$ and its factors, we can compute matrix-vector products with M^{-1} in $O(pn \lg n)$ operations (see §5.4). Second, when to stop the CGLS iteration in order to get the best approximate solution is a well-studied but open question (for instance, see [16] and the references therein). We do not solve this problem, but we consider other algorithmic issues in the following subsections.

5.1. Determining the size of the $K_{ii}^{(1)}$. As shown in §4, the choices of the parameters m_i determine the number of clustered singular values in the preconditioned system. Since p partial factorizations of size m_i need to be computed, they influence the amount of work per iteration. Most importantly, as Theorem 4.6 indicates, $m^* = \sum_{i=1}^{p} m_i$ influences the mixing of upper and lower subspaces. We use a simple heuristic in our numerical experiments. Given the noisy right hand side vector g, let G be the $n \times p$ matrix with entries given by $G_{kj} = g_{(k-1)n+j}$ for $1 \leq k \leq n$, $1 \leq j \leq p$, and let $\tilde{G}$ be its two-dimensional, normalized, inverse discrete Fourier transform. Then it is easy to show that the right hand side z defined in §3.2 results from stacking $\tilde{G}$ by columns. We sort the absolute values of z and determine m^* to be the index of the value, *ctol*, for which the Fourier coefficients start to level off. This is presumed to be the noise level. Since $\tilde{G}$ requires $O(np(\lg p + \lg n))$ operations to compute, the cost involved in determining m^* is also $O(np(\lg p + \lg n))$ operations.

We choose the values m_i using two slightly different methods, which we now describe, and we compare the results in §6. In the first approach, each value m_i is defined as the number of elements in the ith column of $\hat{G}$ which are larger than *ctol*. We call this method of computing the values m_i the Fourier coefficient method. In the second approach, a full factorization is performed on each block $\tilde{K}_{ii}$ so that all the entries of each diagonal matrix D_{ii} are known. We set d to be the N length vector comprised of the diagonals of the D_{ii}, sort the elements in decreasing magnitude, and set *dtol* to be the m^*th largest magnitude element. The value m_i is then defined to be the number of diagonal entries in D_{ii} which have magnitude greater than *dtol*. This is the d-selection method for computing the values m_i.

The latter approach appears the more expensive of the two, requiring $O(pn^2)$ operations to compute all values m_i. However, we found that the entries of D_{ii} decay nearly monotonically so that the values m_i could be similarly obtained by performing the steps of the factorization of each block in parallel. That is, the first step of the factorization is performed on all the blocks sequentially, then the second step on all the blocks, and so forth, up to step j. Block $\tilde{K}_{ii}$ ceases to be factored after step j when $|(D_{ii})_{jj}|$ is determined to be too small. Hence, with careful administration, all values m_i can be computed from the diagonal entries in D_{ii} as they accumulate in $O(\sum_{i=1}^{p} m_i^2)$ operations, where the values m_i are almost identical to the values obtained in our second approach.

5.2. Matrix vector products with C. Recall that C is related to the original BTTB matrix T through a sequence of fast orthogonal transforms and permutation matrices described in §3.2. As was mentioned in the introduction, matrix vector products involving T can be computed in $O(np(\lg n + \lg p))$ operations. It follows that matrix vector products involving C can also be computed in $O(np(\lg n + \lg p))$ operations without ever having to compute the entries of C.

5.3. Computing the preconditioner. By Property 3, $\tilde{K}_{ii}$ satisfies the displacement equation (4) with with $\Omega = S_1$ and $\Theta = S_{-1}$. Therefore using Property 1 $K_{ii}^{(1)}$ satisfies

$$\Omega_1 K_{ii}^{(1)} - K_{ii}^{(1)} \Theta_1 = A_i^{(1)} B_i^{(1)T},$$

where Ω_1 and Θ_1 are the leading principal submatrices of $P_i^T \Omega P_i$ and $Q_i \Theta Q_i^T$ respectively, and $A_i^{(1)}$ and $B_i^{(1)}$ contain the first m_i rows of $P_i^T \tilde{A}_i$ and $Q_i^T \tilde{B}_i$ respectively (refer to §4.3).

From Property 5, the expression for the entries of $K_{ii}^{(1)-1}$ is

$$(19) \qquad K_{ii}^{(1)-1} = -\left(\frac{x_j^{(i)T} w_k^{(i)}}{\tilde{\theta}_j - \tilde{\omega}_k}\right)_{1 \le j,k \le n},$$

where $\tilde{\theta}_j$ and $\tilde{\omega}_k$ are the elements of Θ and Ω that appear in Θ_1 and Ω_1 respectively and, from (6), the vectors $x_j^{(i)}$ and $w_k^{(i)}$ are rows of $X_i^{(1)}$ and $W_i^{(1)}$ defined as

$$K_{ii}^{(1)} X_i^{(1)} = A_i^{(1)}, \;\; W_i^{(1)T} K_{ii}^{(1)} = B_i^{(1)T}.$$

Computing $X_i^{(1)}$ and $W_i^{(1)}$ costs $O(m_i^2)$ operations, given the factorization of $K_{ii}^{(1)}$ and the matrices $A_i^{(1)}$ and $B_i^{(1)}$. Since M^{-1} is a block-diagonal matrix given by $M^{-1} = diag(M_1^{-1}, \ldots, M_p^{-1})$, it takes $O(\sum_{i=1}^{p} m_i^2)$ operations to precompute given the matrices $\tilde{A}_i, \tilde{B}_i$ for $i = 1, \ldots, p$. Since these generator matrices require $O(b_w p \lg p + pn \lg n)$ operations to precompute (refer to the beginning of §4.3), the total cost for precomputing M^{-1} is $O(b_w p \lg p + pn \lg n + \sum_{i=1}^{p} m_i^2)$ operations.

5.4. Applying the preconditioner. Since M^{-1} is block diagonal, to compute $M^{-1}r$ requires the p computations $K_{ii}^{(1)-1} r_i$ where r_i is the length m_i subvector of r beginning at index $ip + 1$. Using Algorithm 2 of [22], we compute each $K_{ii}^{(1)-1} r_i$ stably in $O(n \lg n)$ operations. Thus each application of the preconditioner costs at most a total of $O(pn \lg n)$ operations. (If $p_m < p$ of the values m_i are nonzero, as we often found in practice, the cost reduces to $O(p_m n \lg n)$ operations.) Since matrix vector products involving the BTTB matrix T can be computed in $O(np(\lg p + \lg n))$ operations, each iteration of CGLS costs $O(np(\lg p + \lg n))$ operations.

6. Numerical results. In this section we summarize results of our algorithm on two test problems using Matlab and IEEE floating point double precision arithmetic. Our measure of success in filtering noise is the *relative error*, the 2-norm of the difference between the computed estimate f and the vector $\hat{f}$ corresponding to zero noise, divided by the 2-norm of $\hat{f}$. In each case, we apply the CGLS iteration with block Cauchy-like preconditioner with $m^* = \sum_{i=1}^{p} m_i$. The value $m^* = 0$ corresponds to no preconditioning.

6.1. Example 1. As mentioned in the introduction, BTTB matrices often arise in two dimensional image processing problems. For our first example, we began by generating the 64×64 image shown in Figure 1. The tower in the image is composed of 3 concentric circles, centered at row 46, column 38, of radii 6,4, and 2, with values 2,4, and 8, respectively. The 3×3 spike in the image has its upper left corner at row 39, column 29 and has height 10. The vector $\hat{f}$ was then generated by stacking the image by columns. We consider the BTTB matrix T with entries given by

$$t_{i-j}^{(k-l)} = \begin{cases} ce^{-.1((i-j)^2 + (k-l)^2)}, & -5 \le i-j, k-l \le 5 \\ 0 & otherwise, \end{cases}$$

where c is a normalization constant. This matrix is the one used in [13] and has a condition number of about 10^{11}. Since this is a $BTTB$ matrix, only the first row and first column of the $2n$ blocks $T_i, \;\; -n < i < n$, need to be generated and stored.

We next computed $\hat{g} = T\hat{f}$, and used the Matlab randn function to generate a vector e of length N. We then scaled e such that the noise level, $\frac{\|e\|_2}{\|\hat{g}\|_2}$, was equal

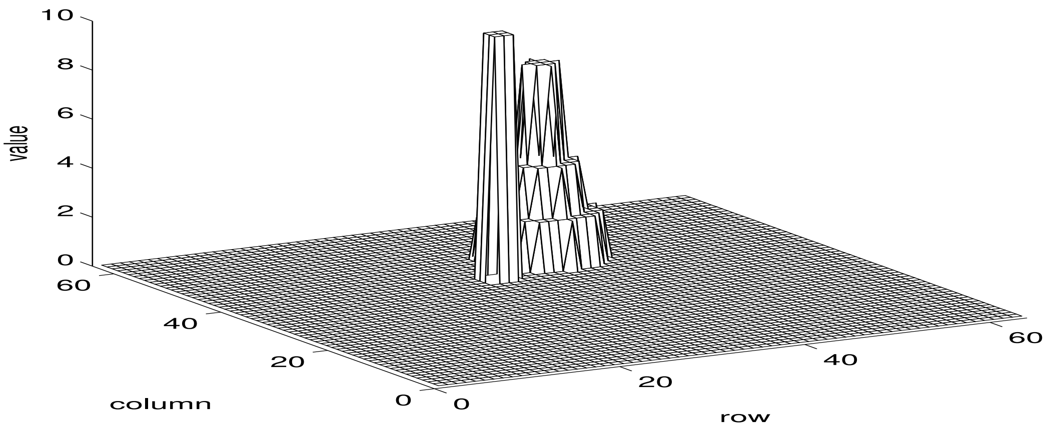

FIG. 1. *Original image, Example 1.*

to 10^{-2}. Finally, we set the noisy data g to $g = \hat{g} + e$. The blurred, noisy image, whose columns are the p length consecutive subvectors of g, is shown in Figure 2, and its sorted absolute 2-D Fourier coefficients are displayed, along with the vector d of §5.1, in Figure 3. An appropriate cutoff m^* and the values m_i were determined as explained in §5.1.

We conducted experiments for different values of m^* using both of the methods for choosing the m_i. The solid line in Figure 4 shows the convergence of CGLS in relative error at each iteration. When $m^* = 0$ (that is, no preconditioning is used), a minimum relative error of 3.41×10^{-1} is achieved at 90 iterations. The dashed line in Figure 4 shows the convergence behavior for a preconditioner determined using $m^* = 711$ and using the d-method of determining m_i. After 13 iterations, a relative error value of 3.53×10^{-1} was reached. The dotted line shows the convergence using a preconditioner which was determined by setting $m^* = 583$ and using the Fourier coefficient method of determining m_i. This was the best preconditioner that could be determined using this selection method; after 13 iterations a relative error of 3.86×10^{-1} was reached. For comparison, the dashed-dot line illustrates the optimal convergence behavior of right-preconditioned scheme in [13], where the cutoff was determined to be 725 eigenvalues. This method achieves a minimum relative error value of 3.49×10^{-1} in 9 iterations.

6.2. Example 2. As a second image processing example, we consider the BTTB matrix $T = T_0 \otimes T_0$ where T_0 is the 32×32 Toeplitz matrix with diagonals (see [20])

$$h_k = \begin{cases} c\dfrac{sin(\frac{k}{B_L})}{\frac{k}{B_L}}^2 & 0 \le |k| \le B_w \\ 0 & \text{otherwise} \end{cases}$$

where c is a normalization constant, $B_L = 2$, and B_w, the bandwidth of T_0, is set to 5. The condition number of T is approximately 1.6×10^8.

We then generate $\hat{f}$ by forming the image shown in Figure 5 and stacking it by columns. The image itself was created by truncating to radius 8 a 2-D Gaussian with standard deviation 30 centered at row 20, column 19, and multiplying the values by 40. A 3×3 spike of height 40 with upper left corner at row 13, column 10, was also

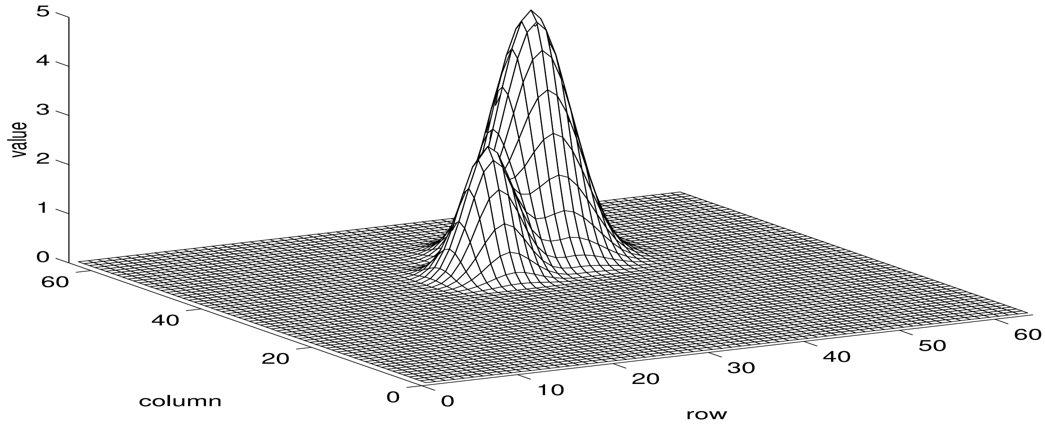

FIG. 2. *Blurred, noisy image, Example 1.*

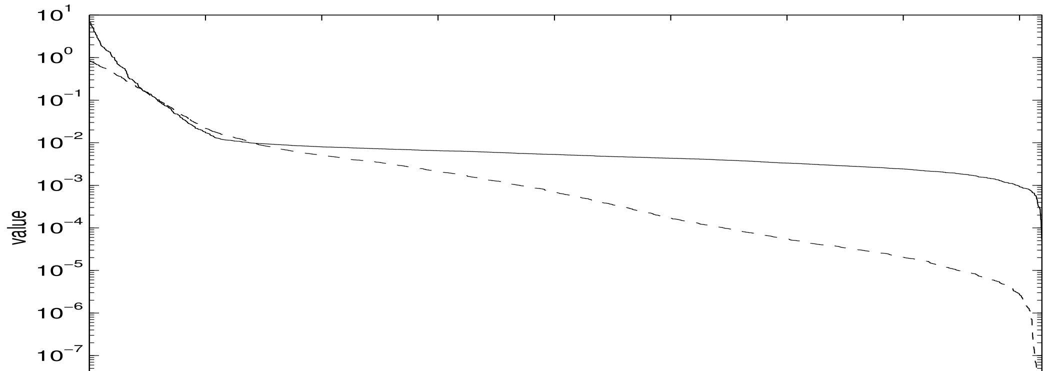

FIG. 3. *Solid line shows 2-D Fourier coefficients of the noisy data sorted in order of decreasing magnitude. Dashed line shows diagonal entries obtained during factorization, sorted in order of decreasing magnitude, Example 1.*

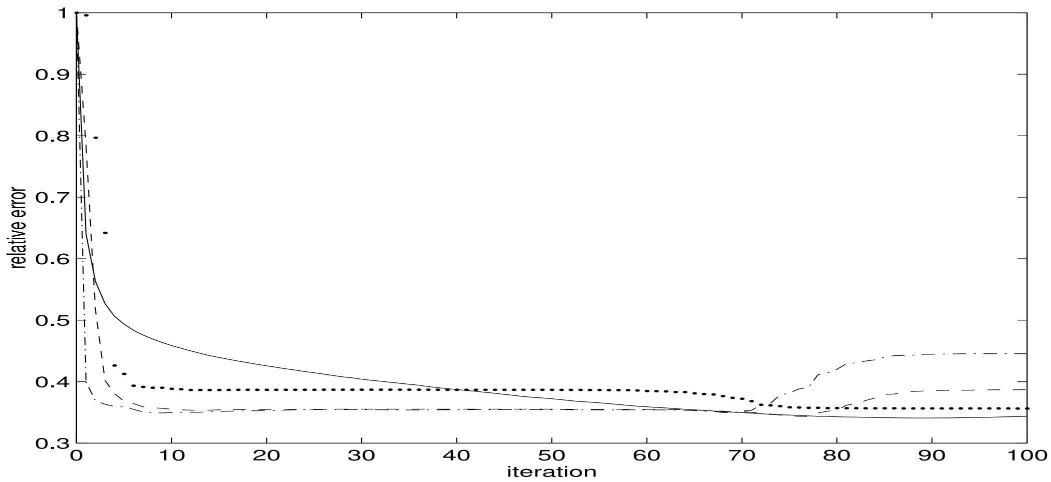

FIG. 4. *Relative error in computed solutions for Example 1. Solid line shows convergence when $m^* = 0$; dashed line shows convergence for our preconditioner with $m^* = 711$ using the d-selection method; dotted line shows convergence when $m^* = 583$ using the Fourier coefficient selection method; dash-dotted line shows the convergence behavior for the preconditioning scheme in [13] with the cutoff at 725 eigenvalues.*

added. Next, we set $g = T\hat{f} + e$, where e is a normally distributed random vector, generated with the Matlab randn function, scaled so that the noise level was 10^{-2}. The blurred noisy image, whose columns are the consecutive p length subvectors of g, is displayed in Figure 6 and its sorted absolute 2-D Fourier coefficients together with the vector d are shown in Figure 7.

The solid line in Figure 8 shows the convergence of CGLS in relative error for Example 2. With no preconditioning (i.e. $m^* = 0$) CGLS required 49 iterations to achieve its minimum relative error value of 2.54×10^{-1}. The dashed line in Figure 8 depicts the convergence of CGLS on the left preconditioned system using our preconditioner with $m^* = 122$ and where the d-method for selecting the m_i is used. After 7 iterations, a value of 2.59×10^{-1} was achieved. The dotted line in the figure shows the convergence behavior on the left preconditioned system when $m^* = 109$ using the Fourier coefficient method of determining m_i. After 6 iterations, the minimum relative error of 2.66×10^{-1} was reached. In comparison, the dashed-dot line illustrates the optimal convergence behavior of the right-preconditioned scheme in [13],where the cutoff was determined to be 116 eigenvalues. This method achieves a minimum relative error value of 2.53×10^{-1} in 7 iterations.

In fact, we note that the matrices in Examples 1 and 2 are special examples of BTTB matrices since they arise from tensor products of Toeplitz matrices. In the case where $T = T_1 \otimes T_2$, $T_1 \neq T_2$, the matrix $\tilde{K}$ can be written as a tensor product of a $p \times p$ matrix times an $n \times n$ Cauchy-like matrix. Thus, only the single $n \times n$ Cauchy-like matrix needs to be factored to obtain the preconditioner. Therefore it is easy to show that the cost of precomputing our preconditioner is reduced to $O(m_1^2 + p \lg p + n \lg n + N)$ operations for this special case. Likewise, the cost of precomputing the preconditioner in [13] reduces to $O(p \lg p + n \lg n + N)$ operations.

6.3. Example 3. In both Examples 1 and 2, the matrix T was symmetric and its block bandwidth was small relative to the number p of blocks. Since unsymmetry

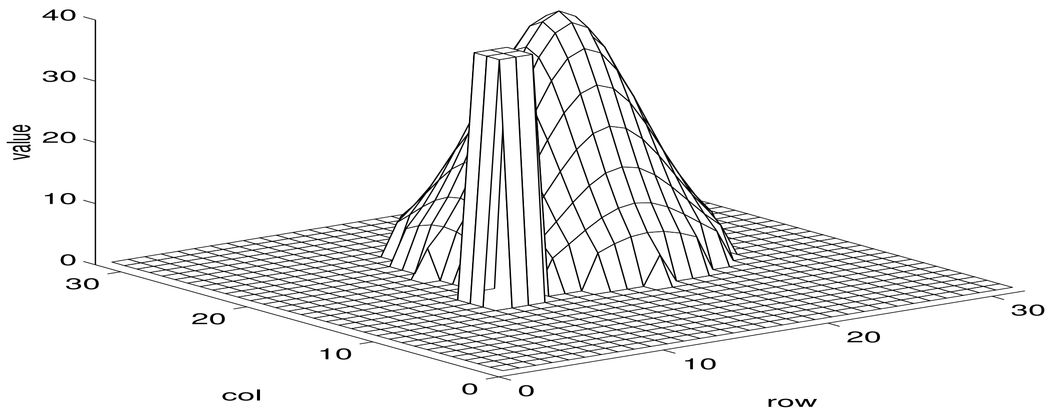

FIG. 5. *Original image, Example 2.*

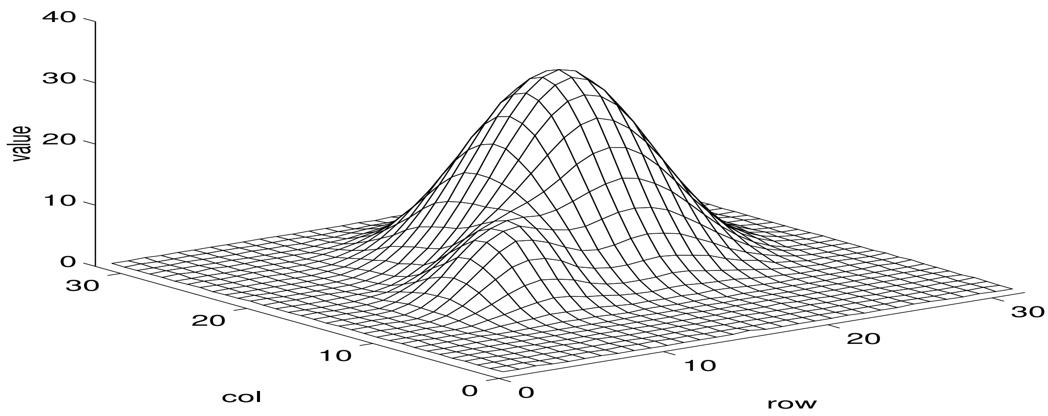

FIG. 6. *Blurred, noisy image, Example 2.*

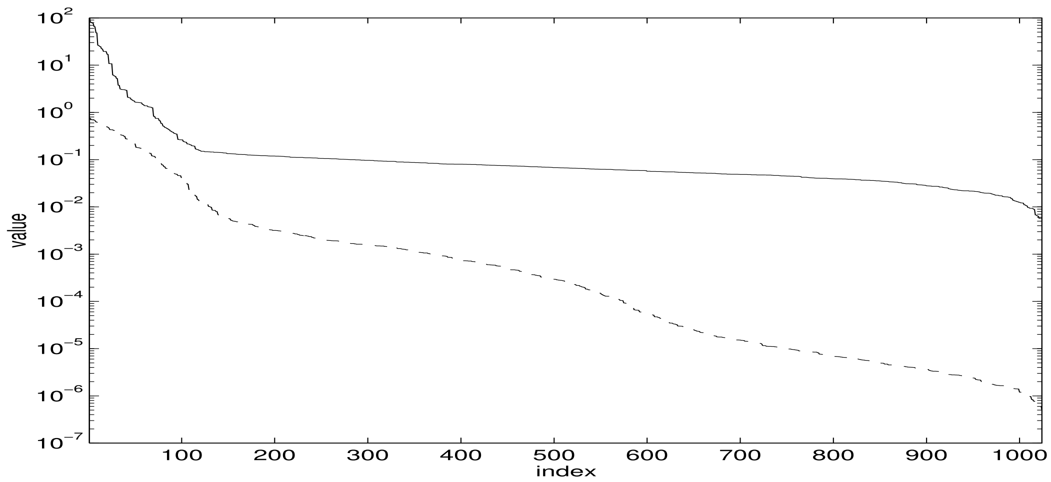

FIG. 7. *Solid line shows 2-D Fourier coefficients of the noisy data sorted in order of decreasing magnitude. Dashed line shows diagonal entries obtained during factorization, sorted in order of decreasing magnitude, Example 2.*

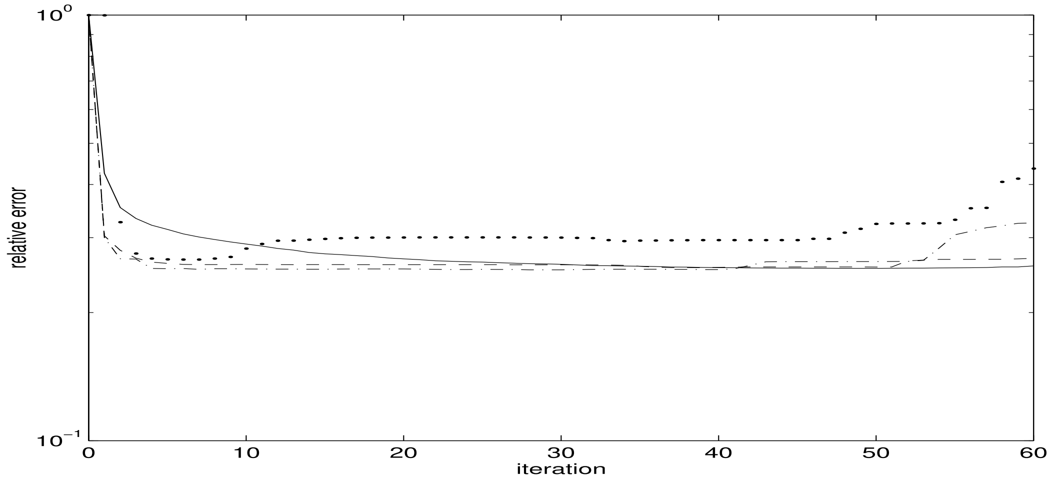

FIG. 8. *Relative error in computed solutions for Example 2. Solid line shows convergence when* $m^* = 0$*; dashed line shows convergence for our preconditioner with* $m^* = 122$ *using the d-selection method; dotted line shows convergence when* $m^* = 109$ *using the Fourier coefficient selection method; dash-dotted line shows the convergence behavior for the preconditioning scheme in [13] with the cutoff at 116 eigenvalues.*

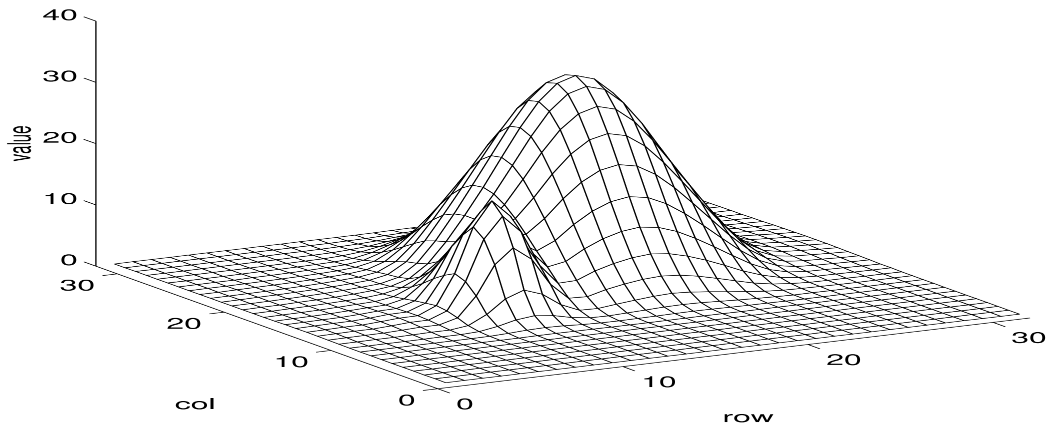

FIG. 9. *Blurred, noisy image, Example 3.*

and larger bandwidth can be encountered in practice, we consider the non-symmetric matrix with a larger block bandwidth as follows. We set $t_j^{(1)}$ and $t_j^{(2)}$ to be the length 32 vectors with entries given by

$$t_j^{(1)} = c_1 e^{(-.1(1-j)^2)} \text{ if } j \leq 6 \text{ and } 0 \text{ otherwise,}$$

$$t_j^{(2)} = c_2 e^{(-.2(1-j)^2)} \text{ if } j \leq 11 \text{ and } 0 \text{ otherwise,}$$

where c_1 and c_2 were normalization constants, and we set $\bar{T}$ to be the Toeplitz matrix with first column t_1 and first row t_2 using the Matlab command $\bar{T} = toep(t^{(1)}, t^{(2)})$. We then generate a matrix H as shown in Example 2 with $B_L = 1$ and $B_w = 12$. Finally, we form T by tensor products:

$$T = H \otimes \bar{T}.$$

The condition number of T is 1×10^9.

For this example we took the exact solution $\hat{f}$ to be same as in Example 2. Next, we set $g = T\hat{f} + e$, where e is a normally distributed random vector scaled so that the noise level was 10^{-3}. The blurred, noisy image, whose columns are the consecutive p length subvectors of g, is displayed in Figure 9. The sorted absolute 2-D Fourier coefficients of g together with the vector d are shown in Figure 10.

The relative error plot in Figure 11 shows that with no preconditioning, CGLS reaches its minimum relative error value of 1.05×10^{-1} at 121 iterations. However, using our preconditioner with $m^* = 576$ and the d-selection method for determining the m_i, a relative error value of 1.06×10^{-1} was reached in only 8 iterations. When $m^* = 435$ and the Fourier coefficient selection method is used to determine our preconditioner, a relative error value of 1.44×10^{-1} was reached in 9 iterations; after 30 iterations, the relative error was improved to 1.18×10^{-1}. In contrast, the preconditioned iterative scheme in [13] could do no better than 1.24×10^{-1} after 17 iterations (the preconditioner which achieved this value was constructed using a cutoff of 574 eigenvalues).

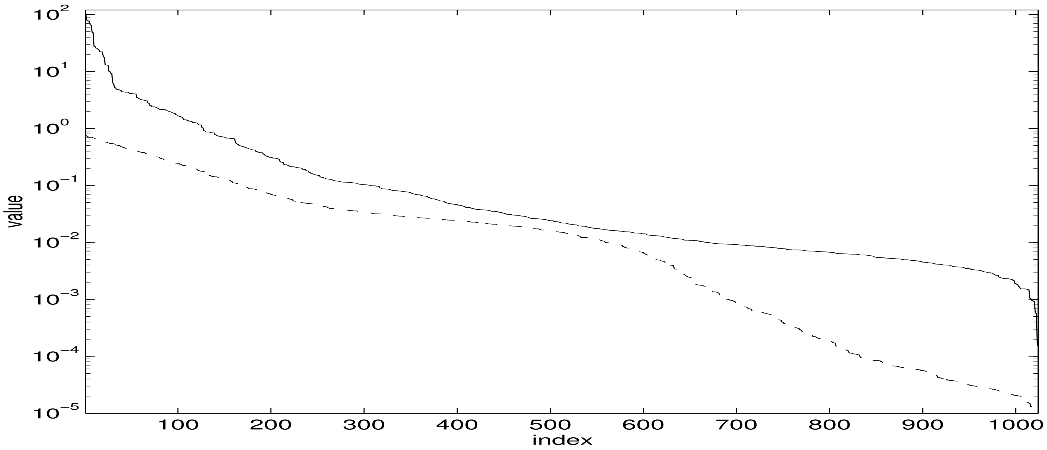

FIG. 10. *Solid line shows 2-D Fourier coefficients of the noisy data sorted in order of decreasing magnitude. Dashed line shows diagonal entries obtained during factorization, sorted in order of decreasing magnitude, Example 3.*

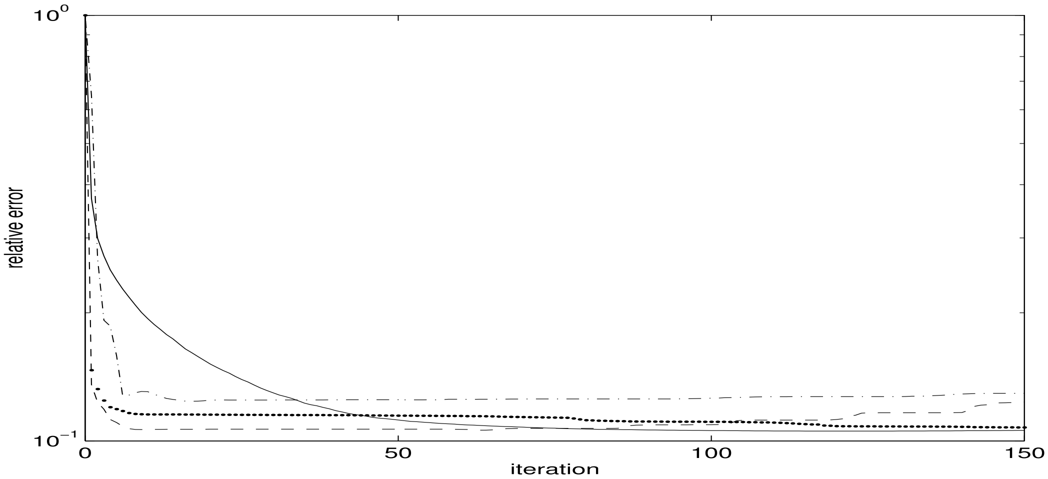

FIG. 11. *Relative error in computed solutions for Example 3. Solid line shows convergence when $m^* = 0$; dashed line shows convergence for our preconditioner with $m^* = 576$ using the d-selection method; dotted line shows convergence when $m^* = 435$ using the Fourier coefficient selection method; dash-dotted line shows the convergence behavior for the preconditioning scheme in [13] with the cutoff at 574 eigenvalues.*

6.4. Results Summary. We conducted several other experiments comparing the effectiveness of our preconditioner with the effectiveness of the preconditioner found in [13]. The experiments, which we now summarize, were conducted using matrices of different sizes and structure, different original images, and various noise levels. First, we found that in cases where the dimension of the transition subspace was large relative to the dimension of the problem, while both preconditioners could be successful in speeding convergence in the first few iterations (i.e. the cutoffs could chosen to cluster the largest singular values) it was unlikely that either preconditioned scheme could, within fewer iterations, produce solutions whose relative errors were comparable to those generated by unpreconditioned CGLS. We attribute this phenomena to the fact that both preconditioners mixed too much noise into early iterates by clustering too many singular values without being able to reconstruct some important components of the solution lying in the transition space.

As the ratio of block bandwidth to block size was increased, we found the BTCB approximation to T did a much better job than the BCCB approximation to T of approximating the mid-range and small singular values of the matrix. We also found this to be true when the matrix was blockwise unsymmetric. Consequently, our preconditioner can show significant improvement for these types of problems over the preconditioner in [13] when the cutoff, determined by the noise level, is large enough to include some mid-range singular values, as evidenced in Example 3. For larger noise levels, there was no consistent or significant advantage to using one precondition over the other. We therefore particularly recommend our preconditioner when T is block unsymmetric, has a ratio of block bandwidth to block size larger than say 1/8, and in other such cases when we expect that the T. Chan BCCB matrix approximation to T will fail to approximate T well on the block level.

Examples 1 and 2 show that when the block bandwidth is small relative to block size, the matrix is symmetric, and the dimension of the upper subspace is small relative to N, the optimal preconditioner in [13] can produce solutions with slightly smaller relative error in somewhat fewer iterations than our optimal preconditioner. It is important to remember that both preconditioners were sensitive to the choice of cutoff so finding the optimal preconditioner is difficult in practice. Also, the cost to initialize our preconditioner in Examples 1 and 2 is of the same order of magnitude as the initialization cost of the preconditioner in [13].

In short, our preconditioner never performs much worse than the preconditioner in [13] and can perform much better in some cases.

7. Conclusions. We have developed an efficient algorithm for computing regularized solutions to discrete ill-posed problems involving BTTB matrices. Our algorithm uses an orthogonal transform to transform the BTTB matrix and its BTCB approximant to Cauchy-like matrices whose blocks are Cauchy-like. It then iterates using the CGLS algorithm on the left preconditioned system, where the preconditioner was determined using size m_i partial factorizations with pivoting on each of the p blocks of the transformed BTCB matrix. By exploiting properties of the transformation, we showed each iteration of CGLS costs $O(np(\lg n + \lg p))$ operations for a Cauchy-like system with p blocks of size n.

The theory developed in §4 predicts that for many types of BTTB matrices, the preconditioner determined in the course of Gu's fast, modified, complete pivoting algorithm can be expected to cluster the largest singular values around one and to keep the small singular values small while leaving the upper and lower subspaces unmixed. Thus, CGLS produces a good approximation to the noise free solution within a small number of iterations. Our results indicate that the algorithm is both efficient and practical with the truncation parameters m_i chosen using our second heuristic. Finally, the results indicate that our preconditioned method is competitive with the preconditioned method of [13] in terms of both the number of iterations required to reach a reasonable regularized solution and the amount of work per iteration.

We note that this preconditioner can also be applied in situations where the matrix T is only block Toeplitz and the blocks are not necessarily Toeplitz. In this case the diagonals of block H_i of the BTCB matrix H will be the T. Chan approximation to block T_i given by the formula (see [5], for instance)

$$h_l^{(i)} = \frac{1}{p} \sum_{j-k=l(\bmod n)} T_{jk}^{(i)} \quad l = 0, \ldots, p-1.$$

Acknowledgments. I would like to thank Professor Dianne P. O'Leary for her helpful comments and guidance during the preparation of this paper. My thanks, also, to Dr. Jim Nagy for providing suggestions on numerical examples.

REFERENCES

[1] R. Chan, J. Nagy, and R. Plemmons, *FFT-based preconditioners for Toeplitz-block least squares problems*, SIAM J. Numer. Anal., 30 (1993), pp. 1740–1768.

[2] ———, *Circulant preconditioned Toeplitz least squares iterations*, SIAM J. Matrix Anal. and Appl., 15 (1994), p. 80.

[3] R. Chan and M. Yeung, *Circulant preconditioners for Toeplitz matrices with positive continuous generating functions*, Math. Comput., (1992), pp. 233–240. Cited by [2].

[4] R. H. Chan and X.-Q. Jin, *A family of block preconditioners for block systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 1218–1235.

[5] R. H. Chan and M. K. Ng, *Conjugate gradient methods for Toeplitz systems*, SIAM Review, 38 (1996), pp. 427–482.

[6] T. Chan, *An optimal circulant preconditioner for Toeplitz systems*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 766–771.

[7] T. F. Chan and P. C. Hansen, *A lookahead Levinson algorithm for general Toeplitz systems*, IEEE Proc. Signal Processing, 40 (1992), pp. 1079–1090. Cited by [11].

[8] H. E. Fleming, *Equivalence of regularization and truncated iteration in the solution of ill-posed problems*, Linear Algebra Appl., 130 (1990), pp. 133–150.

[9] I. Gohberg, T. Kailath, and V. Olshevsky, *Fast Gaussian elimination with partial pivoting of matrices with displacement structure*, Math. Comput., 64 (1995), pp. 1557–1576.

[10] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley Publishing Company, Inc., 1992.

[11] M. Gu, *Stable and efficient algorithms for structured systems of linear equations*, Tech. Report LBL-37690, Lawrence Berkeley Laboratory, 1995.

[12] M. H. Gutknecht and M. Hochbruck, *Look-ahead Levinson and Schur algorithms for non-Hermitian Toeplitz systems*, Tech. Report IPS 93-11, IPS Supercomputing, ETH-Zurich, 1993. Cited by [11].

[13] M. Hanke, J. Nagy, and R. Plemmons, *Preconditioned iterative regularization for ill-posed problems*, Numerical Linear Algebra and Sci. Computing, (1993). L. Reichel, A. Ruttan and R.S. Varga, editors.

[14] P. C. Hansen, *The discrete Picard condition for discrete ill-posed problems*, BIT, 30 (1990), pp. 658–672.

[15] ———, *Rank Deficient and Discrete Ill-Posed Problems*, PhD thesis, Technical University of Denmark, July, 1995. UNIC Report UNIC-95-07.

[16] P. C. Hansen and D. P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, SIAM J. Sci. Comput., 14 (1993), pp. 1487–1503.

[17] G. Heinig, *Inversion of generalized cauchy matrices and other classes of structured matrices*, Linear Algebra in Signal Processing, IMA Volumes in Math. and Appl., 69 (1994), pp. 95–114.

[18] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Bur. Standards, 49 (1952), pp. 409–436.
[19] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, 1991.
[20] A. K. Jain, *An operator factorization method for restoration of blurred images*, IEEE Trans. on Computers, C-26 (1977), pp. 1061–1071.
[21] T. Kailath, S. Kung, and M. Morf, *Displacement ranks of matrices and linear equations*, Journal of Math. Anal. and Appl., 78 (1979), pp. 395–407.
[22] M. E. Kilmer and D. P. O'Leary, *Pivoted Cauchy-like preconditioners for regularized solution of ill-posed problems*, Tech. Report CS-TR-3682, University of Maryland, College Park, 1996.
[23] T. Ku and C.-C. J. Kuo, *On the spectrum of a family of preconditioned block Toeplitz matrices*, SIAM J. Sci. Stat. Comput., 13 (July, 1992), pp. 948–966.
[24] G. W. Stewart and J. Guang Sun, *Matrix Perturbation Theory*, Academic Press, 1990.
[25] D. R. Sweet, *The use of pivoting to improve the numerical performance of Toeplitz matrix algorithms*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 468–493.
[26] A. van der Sluis and H. van der Vorst, *The rate of convergence of conjugate gradients*, Numer. Math, 48 (1986), pp. 543–560.
[27] J. M. Varah, *Pitfalls in the numerical solution of linear ill-posed problems*, SIAM J. Sci. Stat. Comput., 4 (1983), pp. 164–176.