Open access • Journal Article • DOI:10.1137/0730089

FFT-based preconditioners for Toeplitz-block least squares problems

Raymond H. Chan, James G. Nagy, Robert J. Plemmons

Published on: 01 Dec 1993 - SIAM Journal on Numerical Analysis (Society for Industrial and Applied Mathematics)

Topics: Toeplitz matrix, Least squares, Circulant matrix, Matrix (mathematics) and Fast Fourier transform

FFT-BASED PRECONDITIONERS FOR TOEPLITZ-BLOCK LEAST SQUARES PROBLEMS

By

Raymond H. Chan
James G. Nagy
and
Robert J. Plemmons

IMA Preprint Series # 957
April 1992

FFT–Based Preconditioners for Toeplitz-Block Least Squares Problems

Raymond H. Chan, * James G. Nagy † and Robert J. Plemmons ‡

March 27, 1992

Abstract

Discretized 2-D deconvolution problems arising, e.g., in image restoration and seismic tomography, can be formulated as *least squares computations*, $\min \|b - Tx\|_2$, where T is often a large-scale rectangular Toeplitz-block matrix. We consider solving such block least squares problems by the preconditioned conjugate gradient algorithm using square nonsingular circulant-block and related preconditioners, constructed from the blocks of the rectangular matrix T. Preconditioning with such matrices allows efficient implementation using the 1-D or 2-D Fast Fourier Transform (FFT). Two block preconditioners, related to ones proposed by T. Chan and J. Olkin for square nonsingular Toeplitz-block systems, are derived and analyzed. We show that, for important classes of T, the singular values of the preconditioned matrix are clustered around one. This extends our earlier work on preconditioners for Toeplitz least squares iterations for 1-D problems.

It is well known that the resolution of ill-posed deconvolution problems can be substantially improved by regularization to compensate for their ill-posed nature. We show that regularization can easily be incorporated into our preconditioners, and we report on numerical experiments on a Cray Y-MP. The experiments illustrate good convergence properties of these FFT–based preconditioned iterations.

Key Words. Least squares, Toeplitz-block matrix, circulant matrix, preconditioned conjugate gradients, regularization, deconvolution, image restoration, seismic tomography.

AMS(MOS) Subject Classifications. 65F10, 65F15, 43E10.

*Department of Mathematics, University of Hong Kong, Hong Kong. Research supported in part by ONR contract no. N00014-90-J-1695 and DOE grant no. DE-FG03-87ER25037.

†Institute for Mathematics and its Applications, University of Minnesota, Minneapolis, MN 55455.

‡Institute for Mathematics and its Applications, University of Minnesota, Minneapolis, MN 55455, and Department of Mathematics and Computer Science, Wake Forest University, P.O. Box 7388, Winston-Salem, NC 27109. Cray Y-MP time was provided by a grant from the North Carolina Supercomputing Center. Research supported by the US Air Force under grant no. AFOSR-91-0163.

1 Introduction

In this paper we investigate conjugate gradient preconditioners for the iterative solution of some large-scale structured least squares problems arising, for example, in 2-D deconvolution. Our aim is to provide computational algorithms, backed up by formal convergence analyses, and to report on preliminary numerical experience with applying the algorithms to some simulation problems in image restoration.

The preconditioned conjugate gradient (PCG) method is an iterative method of fundamental importance for solving systems of linear equations, *c.f.*, Golub and van Loan [13]. When T is a rectangular matrix with full column rank, one can still use the CG algorithm to find the solution to the least squares problem

$$\min \|b - Tx\|_2. \tag{1}$$

This can be done by applying the algorithm to the normal equations in factored form,

$$T^*(b - Tx) = 0, \tag{2}$$

which can be solved by conjugate gradients without explicitly forming the matrix T^*T.

The convergence rate of the conjugate gradient algorithm depends on the the singular values of the least squares data matrix T, see Axelsson and Lindskog [3]. If the singular values cluster around a fixed point, convergence will be rapid. Thus, to make the algorithm a useful iterative method, one usually *preconditions* the system. The PCG algorithm then solves (1) by transforming the problem with a preconditioner C, applying the conjugate gradient method to the transformed problem, and then transforming back. More precisely, one can use the conjugate gradient method to solve

$$\min \|b - TC^{-1}y\|_2,$$

with $Cx = y$. The version of the PCG algorithm we use is given in [5] and can be stated as follows:

Algorithm 1. PCG for Least Squares. *Let $x^{(0)}$ be an initial approximation to $Tx = b$, and let C be a given preconditioner. This algorithm computes the solution, x, to the least squares problem* (1).

$r^{(0)} = b - Tx^{(0)}$
$p^{(0)} = s^{(0)} = C^{-*}T^*r^{(0)}$
$\gamma_0 = \|s^{(0)}\|_2^2$
for $k = 0, 1, 2, \ldots$
  $q^{(k)} = TC^{-1}p^{(k)}$
  $\alpha_k = \gamma_k / \|q^{(k)}\|_2^2$
  $x^{(k+1)} = x^{(k)} + \alpha_k C^{-1}p^{(k)}$
  $r^{(k+1)} = r^{(k)} - \alpha_k q^{(k)}$
  $s^{(k+1)} = C^{-*}T^*r^{(k+1)}$
  $\gamma_{k+1} = \|s^{(k+1)}\|_2^2$
  $\beta_k = \gamma_{k+1}/\gamma_k$
  $p^{(k+1)} = s^{(k+1)} + \beta_k p^{(k)}$

This approach is generally called the Preconditioned CGNR method. A variation of CGNR is the LSQR method of Paige and Saunders [26]. When the problem is properly scaled, these two methods are mathematically equivalent, *e.g.*, [5].

In this paper we consider the least squares problem (1), where the data matrix is a rectangular matrix with square n-by-n Toeplitz blocks. Such matrices are said to have a Toeplitz-block (TB) structure. It is well-known that a matrix with a block-Toeplitz (BT) structure (*i.e.*, the blocks form a Toeplitz pattern) can always be permuted into a matrix that is Toeplitz-block. Thus we will concentrate on the TB form.

Toeplitz-block least squares problems occur in a variety of 2-D applications, especially in signal and image processing, *c.f.*, Andrews and Hunt [2], Jain [19] and Oppenheim and Schafer [25], and sometimes in computerized geophysical tomography, *c.f.*, Nolet [24] and van der Sluis and van der Vorst [29]. A comparison of the non-preconditioned CGNR algorithm with ART (Algebraic Reconstruction Techniques) for seismic tomography applications, where T is not necessarily Toeplitz, has been given by van der Sluis and van der Vorst [30]. They show that the conjugate gradient method CGNR performs much better than ART for these seismic applications. The general problems we are considering have the following characteristics:

- they are very large;
- they are structured;
- they may have singular values very close to 0;
- the right-hand side is generally perturbed by noise.

A square matrix $T = (t_{jk})$ is said to be *Toeplitz* if $t_{jk} = t_{j-k}$, *i.e.*, T is constant along its diagonals. An n-by-n matrix C is said to be *circulant* if it is Toeplitz and its diagonals c_j satisfy $c_{n-j} = c_{-j}$ for $0 < j \leq n-1$. Recently, we have shown how to precondition Toeplitz least squares computations using square circulant matrices C, and have applied the methods to solving 1-D deconvolution problems. These methods are based upon application of the *Fast Fourier Transform*, FFT, in finding the eigenvalues of C and in performing the CGNR iterations [8]. By applying the FFT, all matrix-vector multiplications involving T and T^* and solutions to linear systems involving C and C^* can be performed in at most $O(m \log n)$ operations, where T is m-by-n [8].

The idea of using the preconditioned conjugate gradient method with circulant preconditioners for solving square positive definite Toeplitz systems was first proposed by Strang [28]. The application of circulant approximations to Toeplitz matrices has been used for some time in image processing, *e.g.*, [4], but not as preconditioners for conjugate gradient iterations.

In this paper we consider the T. Chan [9] circulant approximation for a square Toeplitz matrix. The circulant approximation is optimal with respect to the Frobenius norm. The T. Chan preconditioner is defined for arbitrary square matrices A, and is constructed by averaging the entries along certain pairs of diagonals of A. In particular, for a given generic

n-by-n matrix A, let C be the n-by-n circulant approximation of A as defined in T. Chan [9], *i.e.*, C is the minimizer of $F(X) = ||A - X||_F$ over all circulant matrices X. For the special case where $A \equiv T$, where T is Toeplitz, we note that the (j, ℓ)th entry of C is given by the diagonal $c_{j-\ell}$ where

$$c_k = \begin{cases} \dfrac{(n-k)t_k + kt_{k-n}}{n} & 0 \le k < n, \\ c_{n+k} & 0 < -k < n. \end{cases} \tag{3}$$

Throughout this paper we will use the T. Chan approximations exclusively as a basic tool to construct our preconditioners for Toeplitz-block least squares problems. Other circulant preconditioners can perhaps be used in our work, by appropriately modifying the convergence analysis material in §3. We emphasize that the T. Chan circulant preconditioner can also be defined even when T is not Toeplitz, see [9].

Recently, the use of circulant preconditioners was considered for Toeplitz least squares problems by Nagy [22] and Nagy and Plemmons [23]. In R. Chan, Nagy and Plemmons[8] we establish formal convergence results and provide applications when T has a "1-D TB form". More precisely, we consider kn-by-n matrices T of the form

$$T = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_k \end{bmatrix}, \tag{4}$$

where each square block T_j is a Toeplitz matrix. Notice that by extending the Toeplitz structure of the matrix T and padding zeros to the bottom rows, we may assume without loss of generality that T is m-by-n with $m = kn$ for some positive integer k. This padding is only for convenience in constructing the preconditioner and does not alter the original least squares problem. If T itself is a rectangular Toeplitz matrix, then each block T_j is necessarily Toeplitz. For each block T_j, we construct a circulant approximation C_j. Then our preconditioner is defined as a square circulant matrix C, such that

$$C^*C = \sum_{j=1}^{k} C_j^* C_j.$$

Since each C_j is an n-by-n circulant matrix, it can be diagonalized by the Fourier matrix F, i.e.

$$C_j = F^* \Lambda_j F$$

where Λ_j is diagonal, see Davis [11]. In particular, $\Lambda_j \mathbf{1} = F\vec{c}_j$, where $\mathbf{1}$ is the vector of all ones and $\vec{c}_j$ is the first column of C_j.

Therefore the spectrum of C_j, $j = 1, \cdots k$, can be computed in $O(n \log n)$ operations by using the Fast Fourier Transform (FFT). Since

$$C^*C = F^* \sum_{j=1}^{k} (\Lambda_j^* \Lambda_j) F,$$

C^*C is also circulant and its spectrum can be computed in $O(kn \log n)$ operations, where T is kn-by-n. Here we choose, as in [22, 23], C to be the symmetric positive definite matrix defined by

$$C \equiv F^*(\sum_{j=1}^{k} \Lambda_j^* \Lambda_j)^{\frac{1}{2}} F. \tag{5}$$

Notice that to use the preconditioner C in Algorithm 1 we need only know its eigenvalues. That is, we simply use the right-hand side of (5) to solve linear systems involving C and C^*. The construction is summarized in the following algorithm:

Algorithm 2. Construction of the preconditioner for 1-D problems. *Let T be given in* (4), *and let F denote the n-by-n discrete Fourier matrix. This algorithm computes the eigenvalues of C given in* (5) *and stores them in the diagonal matrix Λ.*

Find the first column, $\vec{c}_j$, of C_j using (3), $\quad j = 1, \cdots, k.$

Find Λ_j from $\Lambda_j \mathbf{1} = F\vec{c}_j$, using the FFT, $\quad j = 1, \cdots, k.$

$\Lambda = (\sum_{j=1}^{k} \Lambda_j^* \Lambda_j)^{\frac{1}{2}}$.

We show in [8] that the cost per iteration in the preconditioned conjugate gradient method based on the use of Λ and F is of the order $O(m \log n)$. As already mentioned in the beginning, the convergence rate of the method depends on the distribution of the singular values of the matrix TC^{-1}, which are the same as the square roots of the eigenvalues of the matrix $(C^*C)^{-1}(T^*T)$. We also show in [8] that if the generating functions of the blocks T_j are 2π-periodic continuous functions and if one of these functions has no zeros, then the singular values of the preconditioned matrix TC^{-1} are clustered around 1. It turns out that the class of 2π-periodic continuous functions contains the Wiener class of functions which in turn contains the class of rational functions considered in Ku and Kuo [21].

By using a standard error analysis of the conjugate gradient method, we then show in [8] that if the condition number $\kappa(T)$ of T is of $O(n^\alpha)$, then the number of iterations required for convergence is at most $O(\alpha \log n + 1)$ where $\alpha > 0$. Since the number of operations per iteration in the conjugate gradient method is of $O(m \log n)$, the total complexity of the algorithm is therefore of $O(\alpha m \log^2 n + m \log n)$. In the case when $\alpha = 0$, *i.e.*, if one block of T is well-conditioned, the method converges in $O(1)$ steps. Hence the complexity is reduced to just $O(m \log n)$ operations for solving these Toeplitz least squares problems. In [8] we also report on numerical experiments and applications of the method to 1-D deconvolution computations in signal restoration.

The outline of this paper is as follows. In §2, we define the *Toeplitz-block least squares problem* and suggest two preconditioners which are based on using either the 1-D or the 2-D FFT in the computations. Our preconditioners are related to those suggested by T. Chan and J. Olkin [10] for the square nonsingular TB case. In §3, we provide a detailed convergence analysis for the resulting preconditioned iterations under mild assumptions on

the generating functions for the Toeplitz blocks. In §4, we discuss an application of our methods in the area of 2-D deconvolution computations arising in image restoration. Such problems generally lead to very ill-conditioned least squares computations, due to the ill-posed nature of associated inverse problems. In §5, we discuss the technique of regularization when the given rectangular Toeplitz-block matrix is ill-conditioned. Numerical results and some concluding remarks on future work are given in §6.

2 Construction of the 2-D Preconditioners

We now derive two preconditioners for Toeplitz-block, *i.e.*, 2-D, least squares problems. The first is called the Level-1 preconditioner, since it is based in part on circulant approximations only at the Toeplitz-block level. The second, Level-2, preconditioner is based on circulant approximations at the Toeplitz-block level and again at a second, block-Toeplitz level . Each preconditioner reduces to the one described in Algorithm 2 for the special case of 1-D Toeplitz least squares problems.

Let T be the kmn-by-mn matrix of the form

$$T = \begin{bmatrix} T^{(1)} \\ T^{(2)} \\ \vdots \\ T^{(k)} \end{bmatrix}, \tag{6}$$

where each block $T^{(i)}$, $i = 1, \cdots, k$, is a *Toeplitz-block* matrix. More precisely, $T^{(i)}$ can be partitioned as

$$T^{(i)} = \begin{bmatrix} T^{(i)}_{1,1} & T^{(i)}_{1,2} & \cdots & T^{(i)}_{1,m} \\ T^{(i)}_{2,1} & T^{(i)}_{2,2} & \cdots & T^{(i)}_{2,m} \\ \vdots & \ddots & \ddots & \vdots \\ T^{(i)}_{m,1} & T^{(i)}_{m,2} & \cdots & T^{(i)}_{m,m} \end{bmatrix}, \quad i = 1, \cdots, k \tag{7}$$

where each $T^{(i)}_{\alpha,\beta}$, $1 \le \alpha, \beta \le m$, is an n-by-n Toeplitz matrix. In particular, the (γ, δ)th entry of $T^{(i)}_{\alpha,\beta}$ depends only on $\gamma - \delta$, i.e.

$$[T^{(i)}_{\alpha,\beta}]_{\gamma,\delta} = [T^{(i)}_{\alpha,\beta}]_{\gamma-\delta}, \quad 1 \le \alpha, \beta \le m, \quad 1 \le \gamma, \delta \le n.$$

We denote $T^{(i)}$ given in (7) as a TB$(1, m, n)$ matrix and T given in (6) as a general TB (k, m, n) matrix. If, moreover, $T^{(i)}$ is also a *block–Toeplitz* (BT) matrix, i.e.

$$T^{(i)}_{\alpha,\beta} = T^{(i)}_{\alpha-\beta}, \quad 1 \le \alpha, \beta \le m,$$

then we call $T^{(i)}$ in (7) a *block–Toeplitz matrix with Toeplitz-blocks*, and denote it as a BTTB$(1, m, n)$ matrix. If each block $T^{(i)}$ in (7) is BTTB$(1, m, n)$, then T in (6) is BTTB

(k, m, n). Such a matrix has a column of k blocks $T^{(i)}$, each of which is a block-Toeplitz matrix with m blocks in each direction, and where each individual block is an n-by-n Toeplitz matrix. In particular, n-by-n Toeplitz matrices are denoted by BTTB$(1, 1, n)$ and the Toeplitz least squares matrix in (4) associated with 1-D problems is BTTB$(k, 1, n)$. Similar notation will be used for circulant-block and block-circulant matrices.

We first describe a preconditioner, to be called the Level-1 preconditioner, which is based upon an application of the 1-D FFT and the computation of a resulting approximate Cholesky factor of T^*T. We will show how to construct an mn-by-mn preconditioning matrix $R = (I_m \otimes F_n)^* \hat{R} (I_m \otimes F_n)$ for Algorithm 1, where $\hat{R}$ is block upper triangular with diagonal blocks, and F_n denotes the Fourier transform matrix of dimension n. Here, $\otimes$ denotes the usual Kronecker product. It turns out that the actual preconditioner R is block upper triangular with circulant blocks. Thus applying the preconditioner R in Algorithm 1 involves application of the 1-D FFT together with backsolves involving $\hat{R}$.

For an n-by-n Toeplitz block $T^{(i)}_{\alpha,\beta}$ of $T^{(i)}$, we let $c(T^{(i)}_{\alpha,\beta})$ denote the T. Chan [9] circulant preconditioner of $T_{\alpha,\beta}$, as defined in (3). Using this notation and the Fourier transform represented by the matrix F_n, we construct a first-level circulant block approximation $c_1(T^{(i)})$ to $T^{(i)}$ given by

$$c_1(T^{(i)}) \equiv (c(T^{(i)}_{\alpha,\beta})) = \begin{bmatrix} c(T^{(i)}_0) & c(T^{(i)}_{-1}) & \cdots & c(T^{(i)}_{1-m}) \\ c(T^{(i)}_1) & c(T^{(i)}_0) & \cdots & c(T^{(i)}_{2-m}) \\ \vdots & \vdots & \ddots & \vdots \\ c(T^{(i)}_{m-1}) & c(T^{(i)}_{m-2}) & \cdots & c(T^{(i)}_0) \end{bmatrix} \tag{8}$$

Notice that we can express $c_1(T^{(i)})$ as

$$c_1(T^{(i)}) = (I_m \otimes F_n)^* \Lambda^{(i)} (I_m \otimes F_n), \quad i = 1, \cdots, k,$$

where the subscripts on the matrices indicate the dimensions and $\otimes$ indicates the Kronecker product. (We will drop the subscripts on these matrices in cases where their dimensions are apparent.) Here $\Lambda^{(i)}$ is a the block matrix whose (α, β) block entry is given by the diagonal matrix

$$\Lambda^{(i)}_{\alpha,\beta} = F c(T^{(i)}_{\alpha,\beta}) F^*.$$

Then the preconditioner R that we seek satisfies

$$R^* R = c_1(T^{(1)})^* c_1(T^{(1)}) + \cdots + c_1(T^{(k)})^* c_1(T^{(k)}) = (I \otimes F)^* B (I \otimes F),$$

where

$$B \equiv (B_{\alpha,\beta}) = \sum_{i=1}^{k} (\Lambda^{(i)})^* \Lambda^{(i)}.$$

Here, each $B_{\alpha,\beta}$ is a diagonal matrix. Observe that the matrix B is symmetric positive definite.

Let $[\vec{v}^{(1)}, \cdots, \vec{v}^{(m)}]^T$ be an mn-vector, where $\vec{v}^{(i)} = [v_1^{(i)}, \cdots, v_n^{(i)}]^T$. Define a permutation matrix P by

$$P[\vec{v}^{(1)}, \cdots, \vec{v}^{(m)}]^T = [v_1^{(1)}, \cdots, v_1^{(m)}, \cdots, v_n^{(1)}, \cdots, v_n^{(m)}]^T.$$

That is, P groups together all the first elements of each block vector, followed by all second elements, etc. Then it is easy to see that P satisfies

$$P^T B P = diag(B_1, \cdots, B_n),$$

where each B_i is an m-by-m symmetric positive definite matrix. Thus we can factor

$$P^T B P = \tilde{R}^T \tilde{R},$$

where $\tilde{R}$ is the Cholesky factor of $P^T B P$, which can be obtained by block factorization of the m-by-m symmetric positive definite blocks B_i. Then we define the *Level-1 preconditioner* R by

$$R \equiv (I \otimes F)^* P \tilde{R} P^T (I \otimes F). \tag{9}$$

Notice that $\hat{R} \equiv P \tilde{R} P^T$ is block upper triangular with diagonal blocks, and hence R is block upper triangular with circulant blocks. We also note that when $T^{(i)}$ is block-Toeplitz with Toeplitz blocks, BTTB$(1, m, n)$, then each B_i is Toeplitz. Thus a fast Toeplitz algorithm [14] can be used to construct $\hat{R}$. The permutation matrix P in (9) is for notational convenience only, and no permutations need to be used in the actual implementation. The Level-1 preconditioner is thus formed by using the 1-D Fourier transform and the Cholesky algorithm. It is essentially a transform based approximate Cholesky factor of T^*T.

The construction of our Level-1 preconditioner for the TB(k, m, n) matrix T is summarized in the following algorithm:

Algorithm 3. Construction of the Level-1 preconditioner for 2-D problems. *Let T be the TB(k, m, n) matrix given as in* (6) *with the $T^{(i)}$ given as in* (7), *and let F denote the n-by-n discrete Fourier matrix. This algorithm computes the Level-1 preconditioner R for T as an approximate factor of T^*T.*

Find the first column, $c(T^{(i)}_{\alpha,\beta})e_1$, of $c(T^{(i)}_{\alpha,\beta})$ using (3), *for all α, β and $i = 1, \cdots, k$,*

Find $\Lambda^{(i)}_{\alpha,\beta}$ from $\Lambda^{(i)}_{\alpha,\beta}\mathbf{1} = Fc(T^{(i)}_{\alpha,\beta})e_1$, using the FFT, for all α, β and $i = 1, \cdots, k$.

Set $\Lambda^{(i)} = (\Lambda^{(i)}_{\alpha,\beta})$ and form $B = \sum_{i=1}^{k} (\Lambda^{(i)})^* \Lambda^{(i)}$.

Factor $B = \hat{R}^* \hat{R}$, $\hat{R}$ block upper triangular with diagonal blocks.

Set $R = (I \otimes F)^* \hat{R} (I \otimes F)$.

In the following table, we write down the cost in the different stages of forming the Level-1 preconditioner R, for the cases where T is either block-Toeplitz (BT) or block-Toeplitz with Toeplitz blocks (BTTB). Of course R is kept in the factored form $R = (I \otimes F)^* \hat{R} (I \otimes F)$. Recall that T is kmn-by-mn.

Computations (for $i = 1, \cdots, k$)	Order of Operations	
	TB: T	BTTB: T
$c(T^{(i)}_{\alpha,\beta})e_1$	km^2n	kmn
$\Lambda^{(i)}$	$km^2n \log n$	$kmn \log n$
B	km^3n	km^2n
$\hat{R}$	m^3n	$nm \log^2 m$

Table 1. Order of operations for constructing the preconditioner R in Algorithm 3.

Observe that for any vector y, $R^{-1}y$ can be computed by the formula $R^{-1}y = (I \otimes F)^* z$, using the FFT, where $z = \hat{R}^{-1}(I \otimes F)y$ can computed using the FFT and block back substitution with diagonal blocks. The computation of $R^{-1}y$ is parallelizable and involves $O(mn^2)$ operations.

For the second, Level-2, preconditioner, each matrix $T^{(i)}$ is approximated by the second-level circulant preconditioner $c_2(T^{(i)})$ related to the one proposed by T. Chan and Oklin [10] for square Toeplitz-block matrices. Then we stack the $c_2(T^{(i)})$ together to get our circulant approximation to T:

$$c_2(T) = \begin{bmatrix} c_2(T^{(1)}) \\ c_2(T^{(2)}) \\ \vdots \\ c_2(T^{(k)}) \end{bmatrix}.$$

Since the matrix $c_2(T)$ is of the type $\mathrm{CB}(k, m, n)$ and is not square, it cannot be used as a preconditioner in Algorithm 1 applied to the block Toeplitz least squares problem.

We now construct a square matrix C, the Level-2 preconditioner, of size mn-by-mn such that

$$C^*C = c_2(T)^* c_2(T) = \sum_{i=1}^{k} c_2(T^{(i)})^* c_2(T^{(i)}). \tag{10}$$

From Chan and Jin [6, Theorem 3], it follows that the matrix $c_2(T^{(i)})$ we seek satisfies

$$c_2(T^{(i)}) = (F_m \otimes F_n)^* \delta[(F_m \otimes F_n) T^{(i)} (F_m \otimes F_n)^*](F_m \otimes F_n) \quad i = 1, \cdots, k, \tag{11}$$

where $\delta[A]$ denotes the diagonal matrix whose diagonal is equal to the diagonal of the matrix A, and F_ℓ denotes the Fourier matrix of size ℓ. (Again, we will drop the subscript for F_ℓ in the following discussion.)

Clearly, the matrix $\delta[(F \otimes F)T^{(i)}(F \otimes F)^*]$ gives the eigenvalues of $c_2(T^{(i)})$. Adopting the procedure used to construct our previous preconditioners, it follows from (10), that we can define the *Level-2 preconditioner* C to be

$$C = (F \otimes F)^* \Phi (F \otimes F), \tag{12}$$

where

$$\Phi = \left\{ \sum_{i=1}^{k} \delta[(F \otimes F)T^{(i)}(F \otimes F)^*]^* \delta[(F \otimes F)T^{(i)}(F \otimes F)^*] \right\}^{1/2} \tag{13}$$

is a square diagonal matrix.

We next introduce an efficient way of generating the diagonal matrix

$$\delta[(F \otimes F)T^{(i)}(F \otimes F)^*].$$

For any $T^{(i)}$, let $c_1(T^{(i)})$ be the first-level circulant approximation to $T^{(i)}$ given by (8), where $c(T^{(i)}_{\alpha,\beta})$ is the T. Chan circulant preconditioner for $T^{(i)}_{\alpha,\beta}$, see (3). In this notation, the γth entry of the first column of $c(T^{(i)}_{\alpha,\beta})$ is given by

$$[c(T^{(i)}_{\alpha,\beta})]_{\gamma,0} = \frac{1}{n} \sum_{\mu-\nu=\gamma(mod n)} [T^{(i)}_{\alpha,\beta}]_{\mu,\nu} \quad 0 \le \gamma \le n-1. \tag{14}$$

Now we apply another first-level circulant approximation $\tilde{c}_1$ to $c_1(T^{(i)})$ to get the second-level circulant approximation $c_2(T^{(i)})$ that we want. The matrix $\tilde{c}_1\{c_1(T^{(i)})\}$ is obtained by treating each block $c(T^{(i)}_{\alpha,\beta})$ in $c_1(T^{(i)})$ as an entry of a matrix and then applying formula (3) to this "point" matrix. More precisely, we form the block-circulant matrix

$$\tilde{c}_1\{c_1(T^{(i)})\} = \begin{bmatrix} \tilde{C}^{(i)}_0 & \tilde{C}^{(i)}_{-1} & \cdots & \tilde{C}^{(i)}_{-m+1} \\ \tilde{C}^{(i)}_1 & \tilde{C}^{(i)}_0 & \cdots & \tilde{C}^{(i)}_{-m+2} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{C}^{(i)}_{m-1} & \tilde{C}^{(i)}_{m-2} & \cdots & \tilde{C}^{(i)}_0 \end{bmatrix}$$

where in the notation (14),

$$\tilde{C}^{(i)}_\gamma = \frac{1}{m} \sum_{\alpha-\beta=\gamma(mod m)} c(T^{(i)}_{\alpha,\beta}), \quad 0 \le \gamma \le m-1. \tag{15}$$

Since each $c(T^{(i)}_{\alpha,\beta})$ is circulant, we see that the sums $\tilde{C}^{(i)}_\gamma$ are circulant matrices. Moreover, by their definition in (15), we see that

$$\tilde{C}^{(i)}_\gamma = \tilde{C}^{(i)}_{\gamma-m}, \quad 0 < \gamma < m.$$

In particular, we see that $\tilde{c}_1\{c_1(T^{(i)})\}$ is then a block-circulant matrix with circulant blocks of the form BCCB$(1, m, n)$. We note further that

$$c_2(T^{(i)}) = \tilde{c}_1\{c_1(T^{(i)})\},$$

see Chan and Jin [6, Theorem 3]. Thus the entries of $c_2(T^{(i)})$ are given by formulas (14) and (15). For our purposes, we need only find the first column of $c_2(T^{(i)})$.

By (11), we see that the eigenvalues of $c_2(T^{(i)})$ are given by the vector

$$\delta[(F \otimes F)T^{(i)}(F \otimes F)^*]\mathbf{1} = (F \otimes F)c_2(T^{(i)})(F \otimes F)^*\mathbf{1},$$

where $\mathbf{1}$ is the vector of all ones. Since $(F \otimes F)^* \mathbf{1} = e_1$, the first unit vector, we see that

$$\delta[(F \otimes F) T^{(i)} (F \otimes F)^*] \mathbf{1} = (F \otimes F) c_2(T^{(i)}) e_1.$$

Thus the eigenvalues of $c_2(T^{(i)})$ can be obtained by taking the 2-D FFT of the first column of $c_2(T^{(i)})$, which is given by (14) and (15). Once we have the eigenvalues of each $c_2(T^{(i)})$, then Φ can be computed easily by (13).

The construction of our Level-2 BCCB preconditioner (*i.e.*, the eigenvalues of C) for the matrix T is summarized in the following algorithm:

Algorithm 4. Construction of the Level-2 preconditioner for 2-D problems. *Let T be the $TB(k,m,n)$ matrix given as in* (6) *with the $T^{(i)}$ given as in* (7) *and let F denote the n-by-n discrete Fourier matrix. This algorithm computes the eigenvalues of the BCCB preconditioner C given in* (12), *using the 2-D FFT, and stores them in the diagonal matrix Φ.*

Find the first column, $\vec{c}_1^{(i)}$, of $c_1(T^{(i)})$ using (14), $\quad i = 1, \cdots, k.$

Find the first column, $\vec{c}_2^{(i)}$, of $c_2(T^{(i)})$ using (15), $\quad i = 1, \cdots, k.$

Find Λ_i from $\Lambda_i \mathbf{1} = (F \otimes F) \vec{c}_2^{(i)}$, using the 2-D FFT, $\quad i = 1, \cdots, k.$

$\Phi = (\sum_{i=1}^{k} \Lambda_i^* \Lambda_i)^{\frac{1}{2}}$.

In the following table, we write down the cost in the different stages of forming the Level-2 BCCB preconditioner C, for the cases where T is either block-Toeplitz (BT) or block-Toeplitz with Toeplitz blocks (BTTB). For implementation in Algorithm 1, we need only the eigenvalues of C together with the 2-D FFT. Recall that T is kmn-by-mn.

Computations (for $i = 1, \cdots, k$)	Order of Operations	
	TB: T	BTTB: T
$\vec{c}_1^{(i)}$	km^2n	kmn
$\vec{c}_2^{(i)}$	km^2n	kmn
Λ_i	$kmn(\log n + \log m)$	$kmn(\log n + \log m)$
Φ	kmn	kmn

Table 2. Order of operations for constructing the eigenvalues of C in Algorithm 4.

We note that for any vector y, by (12), $C^{-1}y$ can be computed by the formula

$$C^{-1}y = (F \otimes F)^* \Phi^{-1} (F \otimes F) y$$

in $O(kmn(\log n + \log m))$ operations by using the 2-D FFT.

3 Convergence Analysis

In this section, we analyze the convergence rates of the methods in the case where all the blocks $T^{(i)}$ in the matrix T are BTTB. For simplicity, we write

$$T^{(i)} = \begin{bmatrix} T_0^{(i)} & T_{-1}^{(i)} & \cdots & T_{2-m}^{(i)} & T_{1-m}^{(i)} \\ T_1^{(i)} & T_0^{(i)} & \ddots & & T_{2-m}^{(i)} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ T_{m-2}^{(i)} & & \ddots & \ddots & T_{-1}^{(i)} \\ T_{m-1}^{(i)} & T_{m-2}^{(i)} & \cdots & T_1^{(i)} & T_0^{(i)} \end{bmatrix}, \quad i = 1, \cdots, k,$$

where each $T_\mu^{(i)}$, $|\mu| < m$, is a Toeplitz matrix by itself. We denote the entries of $T_\mu^{(i)}$ by

$$[T_\mu^{(i)}]_{\gamma,\delta} = t_{\mu,\gamma-\delta}^{(i)}, \quad 0 \le \gamma, \delta < n.$$

We assume that the double-index sequence formed by $t_{\mu,\nu}^{(i)}$ is absolutely summable for each i, i.e.

$$\sum_{\mu=-\infty}^{\infty} \sum_{\nu=-\infty}^{\infty} |t_{\mu,\nu}^{(i)}| \le K_i < \infty \quad i = 1, \cdots, k. \tag{16}$$

Thus we immediately have

$$||T^{(i)}||_2^2 \le ||T^{(i)}||_1 ||T^{(i)}||_\infty \le K_i^2 \quad i = 1, \cdots, k. \tag{17}$$

We will analyze both the Level-1 and Level-2 preconditioners in the following. However, since the proof for the Level-2 preconditioners basically covers most of that for the Level-1 preconditioners, we begin with the Level-2 preconditioners first. The results for Level-1 preconditioners will come as corollaries.

3.1 Level-2 Precondtioners

The following Lemma is proved in Chan and Jin [6, Theorem 4] for the case $T^{(i)}$ is real symmetric BTTB. Its generalization to general complex BTTB matrices is straightforward and therefore we omit its proof here.

Lemma 1 *Let $T^{(i)}$ satisfy (16). Then for all $\epsilon > 0$, there exists $N_i > 0$, such that for all $n > N_i$ and $m > 0$,*

$$T^{(i)} - c_1(T^{(i)}) = U_1^{(i)} + V_1^{(i)}$$

where

$$\text{rank } U_1^{(i)} = O(m)$$

and

$$||V_1^{(i)}||_2 < \epsilon.$$

Thus $T^{(i)}$ and $c_1(T^{(i)})$ are close, apart from a rank $O(m)$ perturbation. Next we will show that $c_1(T^{(i)})$ and $c_2(T^{(i)})$ are also close. We use this fact for the purpose of analyzing the convergence rate for the Level-2 preconditioner.

Lemma 2 *Let $T^{(i)}$ satisfy* (16). *Then for all $\epsilon > 0$, there exists $M_i > 0$, such that for all $m > M_i$ and $n > 0$,*

$$c_1(T^{(i)}) - c_2(T^{(i)}) = U_2^{(i)} + V_2^{(i)}$$

where

$$\text{rank } U_2^{(i)} = O(n)$$

and

$$||V_2^{(i)}||_2 < \epsilon.$$

Proof: We first recall that the first-level circulant preconditioner to $T^{(i)}$ is given by

$$c_1(T^{(i)}) = \begin{bmatrix} c(T_0^{(i)}) & c(T_{-1}^{(i)}) & \cdots & c(T_{1-m}^{(i)}) \\ c(T_1^{(i)}) & c(T_0^{(i)}) & \cdots & c(T_{2-m}^{(i)}) \\ \vdots & \vdots & \ddots & \vdots \\ c(T_{m-1}^{(i)}) & c(T_{m-2}^{(i)}) & \cdots & c(T_0^{(i)}) \end{bmatrix},$$

and that each block $c(T_\mu^{(i)})$ can be diagonalized by the Fourier matrix F. Thus

$$(I \otimes F)c_1(T^{(i)})(I \otimes F)^* = \begin{bmatrix} \Lambda_0^{(i)} & \Lambda_{-1}^{(i)} & \cdots & \Lambda_{1-m}^{(i)} \\ \Lambda_1^{(i)} & \Lambda_0^{(i)} & \cdots & \Lambda_{2-m}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ \Lambda_{m-1}^{(i)} & \Lambda_{m-2}^{(i)} & \cdots & \Lambda_0^{(i)} \end{bmatrix}, \tag{18}$$

where $\Lambda_\mu^{(i)}$ are diagonal matrices with their αth diagonal entries given by

$$[\Lambda_\mu^{(i)}]_{\alpha,\alpha} = \frac{1}{n} \sum_{\nu=-(n-1)}^{n-1} (n - |\nu|) t_{\mu,\nu}^{(i)} e^{-2\pi i \nu \alpha / n}, \quad 0 \le \alpha < n, \tag{19}$$

see for instance Chan and Yeung [7]. Clearly, we have

$$|[\Lambda_\mu^{(i)}]_{\alpha,\alpha}| \le \sum_{\nu=-(n-1)}^{n-1} |t_{\mu,\nu}^{(i)}|, \quad 0 \le \alpha < n. \tag{20}$$

Next we recall that $c_2(T^{(i)}) = \tilde{c}_1[c_1(T^{(i)})]$ is a BCCB matrix and hence it is easy to show that (cf. Chan and Jin [6, Lemma 4])

$$\begin{aligned} (I \otimes F)c_2(T^{(i)})(I \otimes F)^* &= (I \otimes F)\tilde{c}_1[c_1(T^{(i)})](I \otimes F)^* \\ &= \tilde{c}_1[(I \otimes F)c_1(T^{(i)})(I \otimes F)^*]. \end{aligned}$$

By applying formula (15) to the matrix in (18), we get

$$\tilde{c}_1[(I\otimes F)c_1(T^{(i)})(I\otimes F)^*]$$

$$= \begin{bmatrix} \Lambda_0^{(i)} & \frac{m-1}{m}\Lambda_{-1}^{(i)}+\frac{1}{m}\Lambda_{m-1}^{(i)} & \cdots & \frac{1}{m}\Lambda_{1-m}^{(i)}+\frac{m-1}{m}\Lambda_1^{(i)} \\ \frac{m-1}{m}\Lambda_1^{(i)}+\frac{1}{m}\Lambda_{1-m}^{(i)} & \Lambda_0^{(i)} & \cdots & \frac{2}{m}\Lambda_{2-m}^{(i)}+\frac{m-2}{m}\Lambda_2^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{m-1}{m}\Lambda_{-1}^{(i)}+\frac{1}{m}\Lambda_{m-1}^{(i)} & \frac{m-2}{m}\Lambda_{-2}^{(i)}+\frac{2}{m}\Lambda_{m-2}^{(i)} & \cdots & \Lambda_0^{(i)} \end{bmatrix}.$$

Thus

$$\begin{aligned} & (I\otimes F)c_1(T^{(i)})(I\otimes F)^* - (I\otimes F)c_2(T^{(i)})(I\otimes F)^* \\ = \; & (I\otimes F)c_1(T^{(i)})(I\otimes F)^* - \tilde{c}_1[(I\otimes F)c_1(T^{(i)})(I\otimes F)^*] \end{aligned}$$

$$= \frac{1}{m}\begin{bmatrix} 0 & \Lambda_{-1}^{(i)}-\Lambda_{m-1}^{(i)} & \cdots & \Lambda_{2-m}^{(i)}-\Lambda_2^{(i)} & \Lambda_{1-m}^{(i)}-\Lambda_1^{(i)} \\ \Lambda_1^{(i)}-\Lambda_{1-m}^{(i)} & 0 & \ddots & & \Lambda_{2-m}^{(i)}-\Lambda_2^{(i)} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \Lambda_{-2}^{(i)}-\Lambda_{m-2}^{(i)} & & \ddots & \ddots & \Lambda_{-1}^{(i)}-\Lambda_{m-1}^{(i)} \\ \Lambda_{-1}^{(i)}-\Lambda_{m-1}^{(i)} & \Lambda_{-2}^{(i)}-\Lambda_{m-2}^{(i)} & \cdots & \Lambda_1^{(i)}-\Lambda_{1-m}^{(i)} & 0 \end{bmatrix}.$$

For all $\epsilon > 0$, since the sequence $\{t_{\mu,\nu}^{(i)}\}$ is absolutely summable, there exist $\tilde{M}_i > 0$ and $M_i > 2\tilde{M}_i$ such that

$$\sum_{\tilde{M}_i<|\mu|}\sum_{|\nu|<\infty}|t_{\mu,\nu}^{(i)}| < \epsilon \tag{21}$$

and

$$\frac{1}{M_i}\sum_{|\mu|<\tilde{M}_i}|\mu|\sum_{|\nu|<\infty}|t_{\mu,\nu}^{(i)}| < \epsilon. \tag{22}$$

With this given M_i, we write

$$(I\otimes F)c_1(T^{(i)})(I\otimes F)^* - (I\otimes F)c_2(T^{(i)})(I\otimes F)^* = U_2^{(i)} + V_2^{(i)},$$

where $U_2^{(i)}$ is obtained by retaining the last nM_i rows and columns of the matrix

$$(I\otimes F)c_1(T^{(i)})(I\otimes F)^* - (I\otimes F)c_2(T^{(i)})(I\otimes F)^* \tag{23}$$

and setting all other entries to zeros. Thus $V_2^{(i)}$ contains the $[n(m-M_i)]$-by-$[n(m-M_i)]$ principal submatrix of (23) with the other entries being zeros.

Clearly, $\text{rank}U_2^{(i)} = nM_i = O(n)$. It remains to show that $||V_2^{(i)}||_2 < \epsilon$. Since the principal submatrix of $V_2^{(i)}$ is still a block-Toeplitz matrix, it is easy to check that

$$||V_2^{(i)}||_1 \le \sum_{|\mu|<M_i}\frac{|\mu|}{m}||\Lambda_\mu^{(i)}||_1 + \sum_{M_i<|\mu|<m}\frac{m-|\mu|}{m}||\Lambda_\mu^{(i)}||_1.$$

By using (20), (21) and (22), we therefore have

$$||V_2^{(i)}||_1 \leq \sum_{|\mu|<M_i} \frac{|\mu|}{m} \sum_{|\nu|<n} |t_{\mu,\nu}^{(i)}| + \sum_{M_i<|\mu|<m} \frac{m-|\mu|}{m} \sum_{|\nu|<n} |t_{\mu,\nu}^{(i)}| < 2\epsilon,$$

for all $m > M_i$ and $n > 0$.

Similarly, we can show that $||V_2^{(i)}||_\infty < 2\epsilon$. Therefore

$$||V_2^{(i)}||_2^2 < ||V_2^{(i)}||_1 ||V_2^{(i)}||_\infty < 4\epsilon^2. \quad \square$$

Combining Lemmas 1 and 2, we immediately get

Corollary 1 *Let $T^{(i)}$ satisfy* (16). *Then for all $\epsilon > 0$, there exist $M_i, N_i > 0$, such that for all $m > M_i$ and $n > N_i$,*

$$T^{(i)} - c_2(T^{(i)}) = \tilde{U}^{(i)} + \tilde{V}^{(i)}$$

where

$$\text{rank } \tilde{U}^{(i)} = O(m) + O(n)$$

and

$$||\tilde{V}^{(i)}||_2 < \epsilon.$$

Using the Corollary, we are able to show that the singular values of $T^{(i)}$ and $c_2(T^{(i)})$ are close.

Theorem 1 *Let $T^{(i)}$ satisfy* (16). *Then for all $\epsilon > 0$, there exist $N_i, M_i > 0$, such that for all $m > M_i$ and $n > N_i$,*

$$T^{(i)*}T^{(i)} - c_2(T^{(i)})^* c_2(T^{(i)}) = U^{(i)} + V^{(i)}$$

where $U^{(i)}$ and $V^{(i)}$ are Hermitian matrices with

$$\text{rank } U^{(i)} = O(m) + O(n)$$

and

$$||V^{(i)}||_2 < \epsilon.$$

Proof: By Corollary 1, we have

$$\begin{aligned}
& T^{(i)*}T^{(i)} - c_2(T^{(i)})^* c_2(T^{(i)}) \\
= \; & T^{(i)*}(T^{(i)} - c_2(T^{(i)})) + (T^{(i)} - c_2(T^{(i)}))^* c_2(T^{(i)}) \\
= \; & T^{(i)*}(T^{(i)} - c_2(T^{(i)})) - (T^{(i)} - c_2(T^{(i)}))^*(T^{(i)} - c_2(T^{(i)})) + (T^{(i)} - c_2(T^{(i)}))^* T^{(i)} \\
= \; & T^{(i)*}(\tilde{U}^{(i)} + \tilde{V}^{(i)}) - (\tilde{U}^{(i)} + \tilde{V}^{(i)})^*(\tilde{U}^{(i)} + \tilde{V}^{(i)}) + (\tilde{U}^{(i)} + \tilde{V}^{(i)})^* T^{(i)} \\
\equiv \; & U^{(i)} + V^{(i)}.
\end{aligned}$$

Here

$$\begin{aligned} U^{(i)} &= T^{(i)*}\tilde{U}^{(i)} + \tilde{U}^{(i)*}T^{(i)} - \tilde{U}^{(i)*}\tilde{U}^{(i)} - \tilde{U}^{(i)*}\tilde{V}^{(i)} - \tilde{V}^{(i)*}\tilde{U}^{(i)} \\ &= \tilde{U}^{(i)*}(T^{(i)} - \tilde{U}^{(i)} - \tilde{V}^{(i)}) + (T^{(i)} - \tilde{V}^{(i)})^*\tilde{U}^{(i)} \end{aligned}$$

and

$$V^{(i)} = \tilde{V}^{(i)*}T^{(i)} + T^{(i)*}\tilde{V}^{(i)} - \tilde{V}^{(i)*}\tilde{V}^{(i)}.$$

It is clear that both $U^{(i)}$ and $V^{(i)}$ are Hermitian matrices. Moreover we have, by Corollary 1 again, rank $U^{(i)} = O(m) + O(n)$ and

$$||V^{(i)}||_2 < 2\epsilon||T^{(i)}||_2 + \epsilon^2.$$

By (17), we then have

$$||V^{(i)}||_2 < 2\epsilon K_i + \epsilon^2. \quad \square$$

Using the facts that for the Level-2 preconditioner C,

$$T^*T - C^*C = \sum_{i=1}^{k}\{T^{(i)*}T^{(i)} - c_2(T^{(i)})^*c_2(T^{(i)})\}$$

and that k is independent of m and n, we immediately have

Corollary 2 *Let $T^{(i)}$ satisfy (16) for $i = 1, \cdots, k$. Then for all $\epsilon > 0$, there exist $M, N > 0$, such that for all $m > M$ and $n > N$,*

$$T^*T - C^*C = U + V$$

where U and V are Hermitian matrices with

$$\text{rank } U = O(m) + O(n) \tag{24}$$

and

$$||V||_2 < \epsilon. \tag{25}$$

Our next task is to show that C^*C is uniformly invertible for large m and n. Again let $T^{(i)}$ satisfy (16). Define

$$f^{(i)}(x,y) = \sum_{\mu=-\infty}^{\infty}\sum_{\nu=-\infty}^{\infty} t_{\mu,\nu}^{(i)} e^{-i\mu x}e^{-i\nu y}, \quad \forall x, y \in [0, 2\pi]. \tag{26}$$

Notice that since $\{t_{\mu,\nu}^{(i)}\}$ is absolutely summable, the function $f^{(i)}(x,y)$ is well-defined and indeed continuous on $[0,2\pi]^2$ and 2π-periodic in each direction.

Lemma 3 *Let $T^{(i)}$ satisfy* (16). *Define the corresponding function $f^{(i)}(x,y)$ as in* (26). *If*

$$\min_{x,y\in[0,2\pi]} |f^{(i)}(x,y)| \geq \delta > 0, \tag{27}$$

then for sufficiently large m and n, $c_2(T^{(i)})$ is invertible with

$$||c_2(T^{(i)})^{-1}||_2 \leq 2/\delta.$$

Proof: We first recall that $\tilde{c}_1[(I\otimes F)c_1(T^{(i)})(I\otimes F)^*]$ is a block-circulant matrix. Hence it can be block-diagonalized by $(F\otimes I)$. More precisely, we have

$$\begin{aligned}(F\otimes F)c_2(T^{(i)})(F\otimes F)^* &= (F\otimes I)\tilde{c}_1[(I\otimes F)c_1(T^{(i)})(I\otimes F)^*](F\otimes I)^* \\ &= \begin{bmatrix} \Theta_0^{(i)} & & & 0 \\ & \Theta_1^{(i)} & & \\ & & \ddots & \\ 0 & & & \Theta_{m-1}^{(i)} \end{bmatrix},\end{aligned}$$

where (cf (19))

$$\Theta_\beta^{(i)} = \frac{1}{m}\sum_{\mu=-(m-1)}^{m-1}(m-|\mu|)e^{-2\pi i\mu\beta/m}\Lambda_\mu^{(i)}, \quad 0\leq\beta<m.$$

Since $\Theta_\beta^{(i)}$ are diagonal matrices, the eigenvalues of $(F\otimes F)c_2(T^{(i)})(F\otimes F)^*$ (and hence of $c_2(T^{(i)})$) are given by

$$\begin{aligned}&\lambda_{\alpha,\beta}\{(F\otimes F)c_2(T^{(i)})(F\otimes F)^*)\} \\ =\ & [\Theta_\beta^{(i)}]_{\alpha,\alpha} \\ =\ & \frac{1}{m}\sum_{\mu=-(m-1)}^{m-1}(m-|\mu|)[\Lambda_\mu^{(i)}]_{\alpha,\alpha}e^{-2\pi i\mu\beta/m} \\ =\ & \frac{1}{m}\frac{1}{n}\sum_{\mu=-(m-1)}^{m-1}\sum_{\nu=-(n-1)}^{n-1} t_{\mu,\nu}^{(i)}e^{-2\pi i\mu\beta/m}e^{-2\pi i\nu\alpha/n}.\end{aligned}$$

Notice that the last expression can be written in terms of the Fejér sum of $f^{(i)}(x,y)$ evaluated at the point $(2\pi\beta/m, 2\pi\alpha/n)$, see Chan and Yeung [7]. More precisely, we have

$$\lambda_{\alpha,\beta}(c_2(T^{(i)})) = \sigma_{m-1,n-1}[f^{(i)}(x,y)](\frac{2\pi\beta}{m},\frac{2\pi\alpha}{n}), \quad 0\leq\beta<m, 0\leq\alpha<n,$$

where $\sigma_{m-1,n-1}[f^{(i)}](x,y)$ denotes the $(m-1,n-1)$ Fejér sum, (also called the $(C,1)$ sum), evaluated at the point (x,y), see Zygmund [32, p.302]. Since $f^{(i)}(x,y)$ is continuous on

$[0, 2\pi]^2$, the Fejér sum converges uniformly to $f^{(i)}(x, y)$ on $[0, 2\pi]^2$, see Zygmund [32, p.304]. In particular, by (27)

$$|\lambda_{\alpha,\beta}(c_2(T^{(i)}))| = |\sigma_{m-1,n-1}[f^{(i)}](x, y)| \geq \delta/2 > 0$$

for m and n sufficiently large. Since $c_2(T^{(i)})$ can be diagonalized by the unitary matrix $(F \otimes F)$, it is clear that

$$\lambda_{\alpha,\beta}\{c_2(T^{(i)})^* c_2(T^{(i)})\} = |\lambda_{\alpha,\beta}(c_2(T^{(i)}))|^2 \geq (\delta/2)^2 > 0$$

for m and n sufficiently large. □

Corollary 3 *Let $T^{(i)}$ satisfy (16) for $i = 1, \cdots, k$. Define the corresponding functions $f^{(i)}(x, y)$ as in (26). If one of the $f^{(i)}(x, y)$ satisfies (27), then for sufficiently large m and n, C^*C is inverible with*

$$||(C^*C)^{-1}|| \leq K, \tag{28}$$

for some constant K independent of m and n.

Proof: Let $f^{(\ell)}(x, y)$ be the function that satisfies (27). Then by (10) and Lemma 3, we have

$$||(C^*C)^{-1}|| \leq ||[c_2(T^{(\ell)})^* c_2(T^{(\ell)})]^{-1}|| = ||c_2(T^{(\ell)})^{-1}||^2 \leq (2/\delta)^2. \quad \square$$

Now we are ready to prove the main Theorem for the Level-2 preconditioner C.

Theorem 2 *Let $T^{(i)}$ satisfy (16) for $i = 1, \cdots, k$. If one of the $f^{(i)}(x, y)$ as defined in (26) satisfies (27), then for all $\epsilon > 0$, there exist $M, N > 0$, such that for all $m > M$ and $n > N$, at most $O(m) + O(n)$ eigenvalues of the matrix*

$$(C^*C)^{-1}(T^*T) - I$$

have absolute values larger than ϵ. Thus at most $O(m) + O(n)$ of the singular values of the preconditioned matrix

$$TC^{-1}$$

lie outside the interval $(1 - \epsilon, 1 + \epsilon)$.

Proof: By Corollary 2, we have

$$(C^*C)^{-1}(T^*T) - I = (C^*C)^{-1}(T^*T - C^*C) = (C^*C)^{-1}(U + V).$$

Therefore the spectra of the matrices

$$(C^*C)^{-1}(T^*T) - I \quad \text{and} \quad (C^*C)^{-1/2}(U + V)(C^*C)^{-1/2}$$

are the same. However, by (24), we have

$$\text{rank } \left\{(C^*C)^{-1/2}U(C^*C)^{-1/2}\right\} = O(m) + O(n)$$

and by (25), (10) and (28), we have

$$||(C^*C)^{-1/2}V(C^*C)^{-1/2}||_2 \leq ||V||_2||(C^*C)^{-1}||_2 \leq K\epsilon.$$

Thus by applying Cauchy's interlace theorem (see Wilkinson [31]) to the Hermitian matrix

$$(C^*C)^{-1/2}U(C^*C)^{-1/2} + (C^*C)^{-1/2}V(C^*C)^{-1/2},$$

we see that its eigenvalues are clustered around zero except for at most $O(m)+O(n)$ outlying ones. □

Using standard error analysis of conjugate gradient method, we see that the method will converge in at most $O(m) + O(n)$ steps for n and m sufficiently large.

3.2 Level-1 Preconditioners

By substituting Lemma 1 for Corollary 1 in the proof of Theorem 1 and using the fact that

$$T^*T - R^*R = \sum_{i=1}^{k}\{T^{(i)*}T^{(i)} - c_1(T^{(i)})^*c_1(T^{(i)})\},$$

we get, instead of Corollary 2, the following result.

Lemma 4 *Let $T^{(i)}$ satisfy (16) for $i = 1, \cdots, k$. Then for all $\epsilon > 0$, there exist $M, N > 0$, such that for all $m > M$ and $n > N$,*

$$T^*T - R^*R = U + V$$

where U and V are Hermitian matrices with rank $U = O(m)$ and $||V||_2 < \epsilon$.

However to get similar results as in Theorem 2 for the Level-1 preconditioner, we need to bound the smallest singular value of R away from zero as we did in (28). Unfortunately, the conditions on $f^{(i)}(x, y)$ in Theorem 2 are not sufficient for getting such a bound. For a counter-example, consider T with only one block $T^{(1)}$ where $T_1^{(1)} = I$, the identity matrix and $T_j^{(1)} = 0$, the zero matrix, for $j \neq 1$, with $m > 1$. Then the smallest singular value of R^*R is zero whereas C^*C is still well-conditioned. One remedy to this situation is to restrict our class of matrices $T^{(i)}$ from general complex matrices to Hermitian matrices. Suppose there exists an ℓ, $1 \leq \ell \leq k$, such that $f^{(\ell)}(x, y)$ is real-valued. Then clearly $T^{(\ell)}$ is Hermitian and we can show that

$$f_{\min}^{(\ell)} \leq \lambda_{\alpha,\beta}(T^{(\ell)}) \leq f_{\max}^{(\ell)}, \quad 0 \leq \alpha < m, 0 \leq \beta < n,$$

where $f_{\min}^{(\ell)}$ and $f_{\max}^{(\ell)}$ denote the minimum and maximum values of $f^{(\ell)}$, see for instance Jin [20]. We remark that this result is a 2-dimensional generalization of a similar result found in Grenander and Szegö [14].

Using Theorem 1 in Chan and Jin [6], we then have

$$\lambda_{\alpha,\beta}(c_1(T^{(\ell)})) \geq f_{\min}^{(\ell)}, \quad 0 \leq \alpha < m, 0 \leq \beta < n.$$

Thus if additionally $f^{(\ell)}$ is a positive function, i.e. $f_{\min}^{(\ell)} \geq \delta > 0$, then using arguments similar to those used in Corollary 3, we have

$$||(R^*R)^{-1}|| \leq K$$

for some constant K independent of m and n. Using this result instead of (28) in the proof of Theorem 2, we then have our main Theorem for the Level-1 preconditioner.

Theorem 3 *Let $T^{(i)}$ satisfy* (16) *for $i = 1, \cdots, k$. If one of the $f^{(i)}(x,y)$ as defined in* (26) *is a positive function, then for all $\epsilon > 0$, there exist $M, N > 0$, such that for all $m > M$ and $n > N$, at most $O(m)$ eigenvalues of the matrix*

$$(R^*R)^{-1}(T^*T) - I$$

have absolute values larger than ϵ. Thus at most $O(m)$ of the singular values of the preconditioned matrix TR^{-1} lie outside the interval $(1-\epsilon, 1+\epsilon)$.

Using a standard error analysis of the conjugate gradient method, we see that the PCG method, with the Level-1 preconditioner, will converge in at most $O(m)$ steps for n and m sufficiently large.

4 Image Restoration Computations

In this section we consider possible applications of our methods to ill-conditioned inverse problems arising in image restoration. Image restoration refers to the removal or minimization of degradations (or blur) in an image using a priori knowledge about the degradation phenomena. It is important to recover the information in a blurred image: for example, (i) in remote sensing, details about the photographed terrain should be clarified; (ii) in medical imaging, the diagnosis is based on the clarity of the x-ray radiographs taken; and (iii) in space activities, images transmitted to earth by unmanned or manned spacecraft need to be analyzed. When the quality of the recorded images is degraded by blurring and noise, important information remains hidden and cannot be directly interpreted without numerical processing.

Here we will apply our preconditioned Toeplitz iterative least squares schemes to image restoration (deblurring) computations. We remark that the presentation of the image restoration problem given here is brief. The interested reader can find more thorough discussions on this topic in Andrews and Hunt [2] and Jain [19].

We begin with a mathematical model of the image restoration problem. The image of an object can be modeled as

$$g(\xi,\delta) = s\left\{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(\xi,\delta;\alpha,\beta)f(\alpha,\beta)d\alpha d\beta\right\} + \eta(\xi,\delta)$$

where $g(\xi,\delta)$ is the recorded (or degraded) image, $f(\alpha,\beta)$ is the ideal (or original) image, the vector $\eta(\xi,\delta)$ represents additive noise and $s\{\cdot\}$ represents the nonlinear characteristics of the device which senses and records the image. The function $h(\xi,\delta;\alpha,\beta)$ is called the *point spread function* (PSF) and represents the degradation of the image. Typically the nonlinearity $s\{\cdot\}$ is either neglected, or linearized [2], resulting in a model given by

$$g(\xi,\delta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(\xi,\delta;\alpha,\beta)f(\alpha,\beta)d\alpha d\beta + \eta(\xi,\delta). \tag{29}$$

In the digital implementation of (29), the integral is discretized using some quadrature rule, to obtain the discrete scalar model

$$g(i,j) = \sum_{k=1}^{N}\sum_{l=1}^{N} h(i,j;k,l)f(k,l) + \eta(i,j).$$

Writing this in matrix-vector notation, we obtain the linear algebraic form of the image restoration problem,

$$g = Hf + \eta, \tag{30}$$

where g, η and f are N^2–vectors and H is an $N^2 \times N^2$ matrix. This is the square image formulation. Often the discretization is chosen so that g is much larger than f. In this case, H is a rectangular M^2-by-N^2 matrix with $M >> N$.

The image processing problem can be stated as follows. Given the observed image g, the matrix H (which represents the degradation) and, possibly, the statistics of the noise vector η, compute an approximation to the original signal f.

Some remarks on the linear algebraic model (30) are needed. Firstly, notice that the addition of the nonzero noise vector η, which is modeled as a random process, may imply that there is no unique solution to (30). Secondly, the matrix H is usually ill-conditioned and, hence, extremely sensitive to noise [2]. Thus one cannot consider the noise vector η insignificant and simply solve $Hf = g$. As we will see in §5, one method to overcome this ill-conditioning is to use a method of regularization. This has been shown (*e.g.*, [4]) to be an effective method for solving signal and image restoration problems.

Finally, observe that the sizes of the arrays in (30) can become quite large. For example, in low resolution imagery, $N = 256$. Thus the matrix H is $65,536 \times 65,536$. In high resolution imagery, N is typically on the order of 10^3, which implies the matrix H has dimensions on the order of $10^6 \times 10^6$ [2]. The sizes of typical image restoration problems make the use of iterative methods of solution almost essential [4]. These remarks indicate that if H has no special structure then the signal restoration problem can be nontrivial and extremely complicated to solve. But in many cases, fortunately, H has a TB or a BTTB structure, allowing for more efficient methods.

Writing the PSF as $h(\xi, \delta; \alpha, \beta)$ provides the most general description of the imaging system. This representation allows the PSF to vary with position in both the image and object planes. In this case the PSF is said to be *spatially variant.* If we assume the PSF is spatially variant, then the matrix H in (30) has no special structure. Thus computing a solution to (30), in this case, can be very expensive.

In many cases, though, the PSF acts uniformly across the image and object planes. That is, the PSF is independent of position and, hence, becomes a function of only $\xi - \alpha$ and $\delta - \beta$. In this case the PSF is said to be *spatially invariant*, and is written as

$$h(\xi, \delta; \alpha, \beta) = h(\xi - \alpha, \delta - \beta).$$

Thus the image model (29) is written as

$$g(\xi, \delta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(\xi - \alpha, \delta - \beta) f(\alpha, \beta) d\alpha d\beta + \eta(\xi, \delta), \tag{31}$$

where the integral in (31) is a 2–dimensional convolution. The inverse problem of recovering f is thus a 2-D *deconvolution problem.* In the discrete implementation, (31) becomes

$$g = Hf + \eta$$

where the matrix $T \equiv H$ is now a block-Toeplitz matrix with Toeplitz blocks *i.e.,* BTTB. The image restoration problem, with a spatially invariant PSF, can be reduced to solving an ill-conditioned BTTB system using a regularization method to be discussed in §5. Applying the preconditioned conjugate gradient algorithm with the Level-1 or Level-2 preconditioner derived in §2, a solution to the space invariant signal restoration problem may be efficiently computed.

5 Preconditioned Regularized Least Squares

In this section we consider solving least squares problems (1), where the rectangular matrix T comes from an ill-posed inverse problem , such as 2-D deconvolution, and is thus very ill-conditioned. Such problems arise in many applications, such as signal and image restoration, as discussed in §4. Often, the ill-conditioned nature of T results from discretization of ill-posed problems in partial differential and integral equations [15]. Here for example, the problem of estimating an original image from a blurred and noisy observed image is an important case of an *inverse problem*, and was first studied by Hadamard [16] in the inversion of certain integral equations. Because of the ill-conditioning of T, naively solving $Tx = b$ will lead to extreme instability with respect to perturbations in b. The method of *regularization* can be used to achieve stability for these problems [5].

In classical *Tikhonov regularization* [15], stability is attained by introducing a stabilizing operator (called a regularization operator) which restricts the set of admissible solutions. Since this causes the regularized solution to be biased, a scalar (called a regularization

parameter) is introduced to control the degree of bias. More specifically, the regularized solution is computed as

$$\min \| \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} T \\ \mu L \end{bmatrix} x(\mu) \|_2, \tag{32}$$

where μ is the regularization parameter and the $p \times n$ matrix L is the regularization operator.

The standard least squares solution to (1), given by $x = T^{\dagger} b$, is useless for these problems because it is dominated by rapid oscillations due to the errors. Hence in (32), one adds a term $\|Lx\|^2$ in order to *smooth* the solution x. Choosing L as a kth difference operator matrix forces the solution to have a small kth derivative. For example, the 2nd difference operator L has the banded rectangular Toeplitz form:

$$L = \begin{bmatrix} 1 & & & & & & \\ -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & \\ & & & & & 1 & \end{bmatrix}.$$

The regularization parameter μ controls the degree of smoothness (*i.e.*, degree of bias) of the solution, and is usually small. Choosing μ is not a trivial problem. In some cases a priori information about the signal and the degree of perturbations in b can be used to choose μ [1], or generalized cross-validation techniques may be used, *e.g.,* [5]. If no *a priori* information is known, then it may be necessary to solve (32) for several values of μ [12]. Recent analytical methods for choosing an optimal parameter μ are discussed by Reeves and Mersereau [27]. Other regularization schemes for ill-posed problems are discussed by Hanke [17], and by Hanke, Nagy and Plemmons [18].

Based on the discussion above for Tikhonov regularization, the regularization operator L is usually chosen to be the identity matrix or some discretization of a differential operator [4, 12]. In problems arising in 2-D applications, the matrix T will have a block structure. In this case the regularization operator L is often chosen to be the identity matrix, the Laplacian, or a discretization of a 2-D differentiation operator [2]. Hence, if T has the Toeplitz-block form (4), then the matrix

$$\tilde{T} = \begin{bmatrix} T \\ \mu L \end{bmatrix}$$

retains this structure, with the addition of one block (or two blocks if L has more rows than columns). Since $\tilde{T}$ has the block structure (4), we can form Level-1 and Level-2 preconditioners for $\tilde{T}$ and use Algorithm 1 for least squares problems to solve (32). Notice that if L is chosen to be the identity matrix, then the Level-2 preconditioner for $\tilde{T}$ can be constructed by simply adding μ to each of the eigenvalues of the preconditioner C for T.

In addition, the last block in $\tilde{T}$ (*i.e.*, μI) has singular values μ. Due to the remarks at the end of §3, if each block in T has an appropriate generating function, and if μ is not too small, then $\kappa(T) = O(1)$. It follows then, based on the analytical results for these problems given is §3, that (32) may be solved efficiently by our methods.

6 Numerical Tests and Final Comments

In this section we report on some numerical experiments using both the Level-1 and Level-2 preconditioners in the conjugate gradient algorithm for solving Toeplitz least squares problems of the form (6). The good performance of our preconditioners is illustrated using test generating sequences from [6], as well as an ill-conditioned problem arising in image restoration simulations. All numerical examples were performed on the Cray Y-MP at the North Carolina Supercomputing Center.

Example 1: In this example we consider matrices of the form

$$T = \begin{bmatrix} T^{(1)} \\ T^{(2)} \\ T^{(3)} \\ T^{(4)} \end{bmatrix},$$

where each $T^{(i)}$, $i = 1, 2, 3, 4$, is a BTTB$(1, m, n)$ matrix. The diagonals of the $T^{(i)}$ are given by the generating sequences

$$\begin{aligned}
&(1)\quad t_k^{(j)} = \frac{1}{(j+1)(|k|+1)^{1+0.1\times(j+1)}}, && j \geq 0, \quad k = 0, \pm 1, \pm 2, \cdots, \\
&(2)\quad t_k^{(j)} = \frac{1}{(j+1)^{1.1}(|k|+1)^{1+0.1\times(j+1)}}, && j \geq 0, \quad k = 0, \pm 1, \pm 2, \cdots, \\
&(3)\quad t_k^{(j)} = \frac{1}{(j+1)^{1.1}+(|k|+2)^{1.1}}, && j \geq 0, \quad k = 0, \pm 1, \pm 2, \cdots, \\
&(4)\quad t_k^{(j)} = \frac{1}{(j+1)^{2.1}+(|k|+1)^{2.1}}, && j \geq 0, \quad k = 0, \pm 1, \pm 2, \cdots.
\end{aligned}$$

These generating functions come from Chan and Jin [6]. They note that the sequences (2) and (4) are absolutely summable, while (1) and (3) are not. The right hand side vector b is chosen to be the vector of all ones and the zero vector is the initial guess. The stopping criteria we use is $\|s^{(j)}\|_2 / \|s^{(0)}\|_2 < 10^{-7}$, where $s^{(j)}$ is the (normal equations) residual after j iterations.

The convergence results for this example are given in Table 3, which shows the numbers of iterations required for the PCG algorithm to converge using no preconditioner and our Level-1 and Level-2 preconditioners, for several values of m and n. We can see from Table 3 that, for this problem, our preconditioners accelerate the convergence rate of the PCG algorithm quite significantly. Moreover, as m and n increase, the number of iterations for the Level-1 preconditioner remains essentially constant, while the number of iterations increases very slowly for the Level-2 preconditioner. But if no preconditioner is used, then the number of iterations increases dramatically with m and n.

$m = n$	kmn	mn	no prec.	Level-1	Level-2
8	256	64	24	8	11
16	1024	256	80	9	13
32	4096	1024	220	9	15
64	16384	4096	> 220	9	16

Table 3. Number of iterations for Example 1.

In Tables 4, 5 and 6 we show the cpu timings on the Cray Y-MP using four processors, for the PCG method with no preconditioner, the Level-1 preconditioner and the Level-2 preconditioner, respectively. The timings per iteration were done by taking an average over 5 iterations. We remark that the code for performing the preconditioner linear system solves, for both the Level-1 and Level-2 preconditioners, may not be optimal. Thus, it may not be fair to compare the timings for the Level-1 and Level-2 preconditioned systems with each other. But these tables do show, for this problem, that we gain excellent timing speed-ups using our preconditioners. Finally, in Table 7 we summarize the numerical results for Example 1, and in Figure 1 we plot the convergence history of the PCG using no preconditioner and both the Level-1 and Level-2 preconditioners for the case 4096×1024 size problem.

$kmn \times mn$	initial. time	time/iter.	total time
256×64	3.48×10^{-2}	6.96×10^{-2}	1.68×10^{0}
1024×256	1.96×10^{-1}	3.93×10^{-1}	3.18×10^{1}
4096×1024	1.09×10^{0}	2.18×10^{0}	4.81×10^{2}
16384×4096	6.84×10^{0}	1.37×10^{1}	*

Table 4. CPU timings (in seconds) for Example 1, using no preconditioner.

$kmn \times mn$	initial. time	time/iter.	total time
256×64	5.26×10^{-2}	6.97×10^{-2}	$6.11 \times^{-1}$
1024×256	3.00×10^{-1}	3.97×10^{-1}	3.87×10^{0}
4096×1024	1.75×10^{0}	2.19×10^{0}	2.15×10^{1}
16384×4096	1.22×10^{1}	1.38×10^{1}	1.36×10^{2}

Table 5. CPU timings (in seconds) for Example 1, using the Level-1 preconditioner.

$kmn \times mn$	initial. time	time/iter.	total time
256×64	3.97×10^{-2}	7.12×10^{-2}	8.13×10^{-1}
1024×256	4.62×10^{-1}	8.40×10^{-1}	1.13×10^{1}
4096×1024	1.12×10^{0}	2.19×10^{0}	3.38×10^{1}
16384×4096	7.07×10^{0}	1.37×10^{1}	2.27×10^{2}

Table 6. CPU timings (in seconds) for Example 1, using the Level-2 preconditioner.

	no prec.		Level-1		Level-2	
$kmn \times mn$	its.	time	its.	time	its.	time
256×64	24	1.68×10^{0}	8	6.11×10^{-1}	11	8.13×10^{-1}
1024×256	80	3.18×10^{1}	9	3.87×10^{0}	13	1.13×10^{1}
4096×1024	220	4.81×10^{2}	9	2.15×10^{1}	15	3.38×10^{1}
16384×4096	> 220	*	9	1.36×10^{2}	16	2.27×10^{2}

Table 7. Summary of numerical results for example 1.

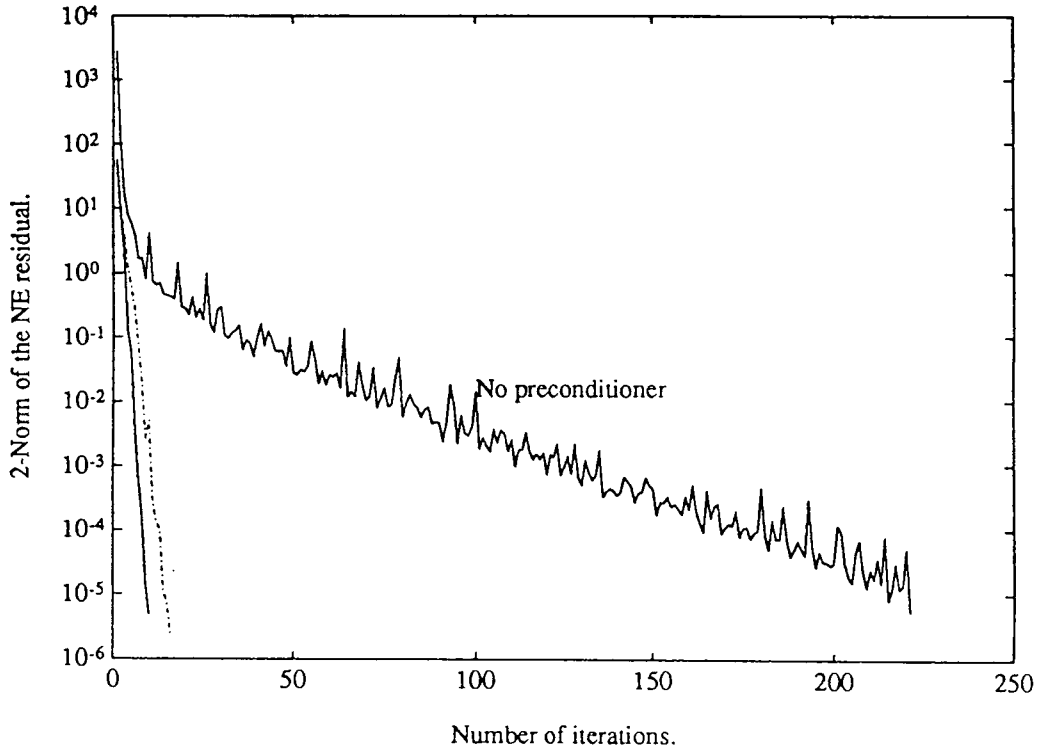

Figure 1: Convergence history for the 4096×1024 problem in Example 1.

Example 2: In this example we apply our PCG methods to a simulated image restoration problem with a spatially invariant point spread function. Recall that, in this case, the image restoration problem can be modeled as the discrete problem

$$g = Hf + \eta,$$

where H is a BTTB matrix, f represents the true image, g represents the blurred, noisy image, and η represents the noise.

The 2-D simulation provided in this example is constructed as follows. First, we generate the known 64×64 image shown in Figure 2, and consider the spatially invariant discretized

PSF matrix H defined by

$$h_{i-j,k-l} = \begin{cases} \exp\{-0.1((i-j)^2+(k-l)^2)\} & -8 \le i-j, k-l \le 8, \\ 0 & \text{otherwise} \end{cases} . \tag{33}$$

In the image processing community this is known as a Gaussian PSF, and can be used to model aberrations in a lens with finite aperture [2, 19]. The matrix H for this example is 4096×4096.

We construct the observed image by forming the vector $g = Hf + \eta$, where H is the 4096×4096 BTTB matrix defined by equation (33), f is a vector formed by row ordering (see [19]) the original image in Figure 2, and η is a vector with random entries chosen from a normal distribution with a mean of 0.0 and a variance of 1.0. The condition number of H in this example is $O(10^7)$. The noise vector η is scaled so that the noise to signal ratio, $\|\eta\|_2/\|Hf\|_2$, was 10^{-3}. By unstacking the vector g (again using row ordering), we obtain the blurred noisy image, shown also in Figure 2.

Our goal is, given g and H, to recover an approximation to the original image f. Because of the ill-conditioning of H, we use the method of Tikhonov regularization described in §5 to stabilize the problem and to obtain a reasonable solution. Here we use the identity matrix as the regularization operator, and $\mu = 0.1$ as the regularization parameter. The parameter μ was chosen based on several runs.

To test the effectiveness of our methods, we used the PCG with our preconditioners, and with no preconditioner, and used the relative normal equations residual norm of 10^{-3} as a stopping tolerance. The results for this simulated image restoration problem are summarized in Table 8, and Figures 3-6.

	no prec.	Level-1	Level-2
iterations	108	32	42
CPU time (sec.)	729.87	214.22	278.19

Table 8. Convergence results for the simulated image restoration in Example 2.

From these experiments, we notice that the restorations after just a few iterations of the preconditioned computations are fairly good, except for some edge artifacts. In contrast, the restoration obtained when no preconditioner is used are not very good until several iterations are completed. It is also interesting to note that after just a few iterations, the restorations obtained from the Level-1 preconditioned system are very good, except for some edge artifacts which appear only in one direction (*i.e.,* on two edges). Moreover, the Level-2 preconditioned system also produces good restorations after only a few iterations, but with edge artifacts in two directions (*i.e.,* on all four edges). This is due to the fact that the Level-1 preconditioner approximates T at *one level* or *one direction* only, whereas the Level-2 preconditioner approximates T at *two levels* or *both directions*, see Chan and Jin [6].

It should be pointed out that only preliminary experiments with applying the preconditioners have been reported in this paper. The results thus far look promising, and we intend to test our methods on actual image restoration problem in the next phase of our work.

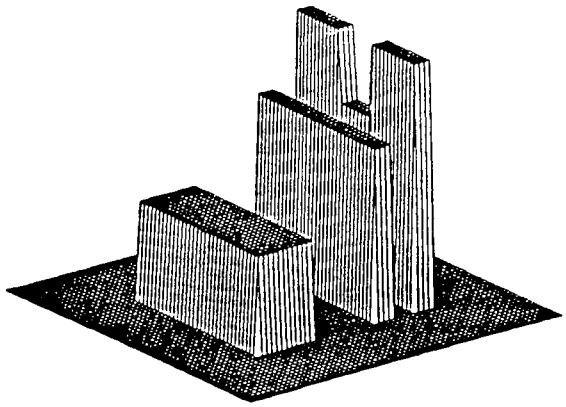

Figure 2a: Original simulated image.

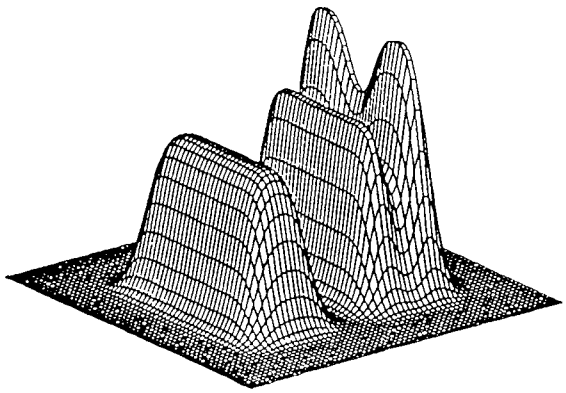

Figure 2a: Blurred, noisy image.

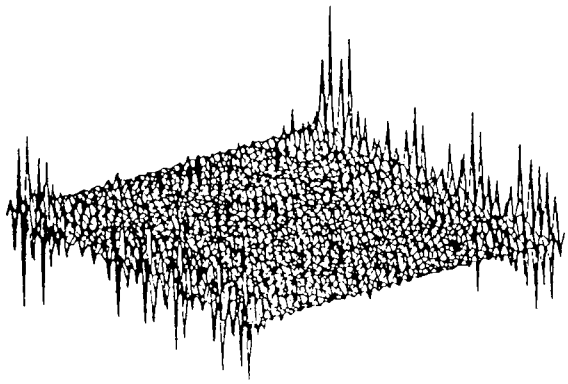

Figure 3: Restoration after 2 iterations with Level-1 preconditioner and no regularization.

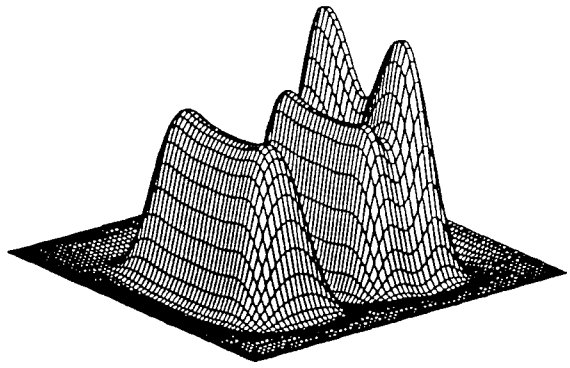

Figure 4a: Restoration after 2 iterations using no preconditioner.

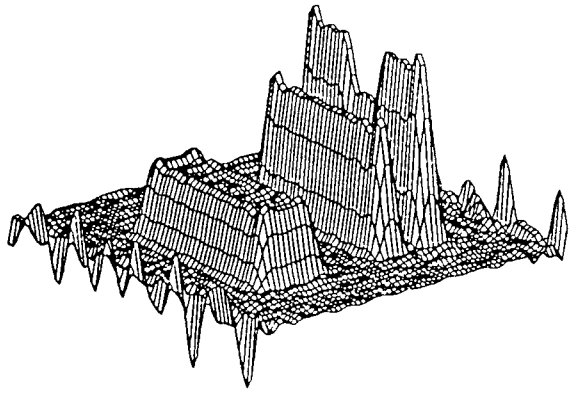

Figure 4b: Restoration after 2 iterations using the Level-1 preconditioner.

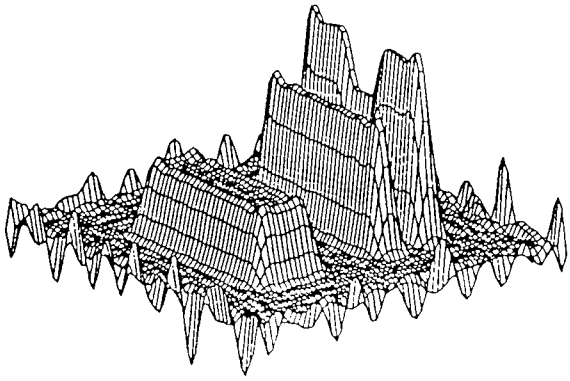

Figure 4c: Restoration after 2 iterations using the Level-2 preconditioner.

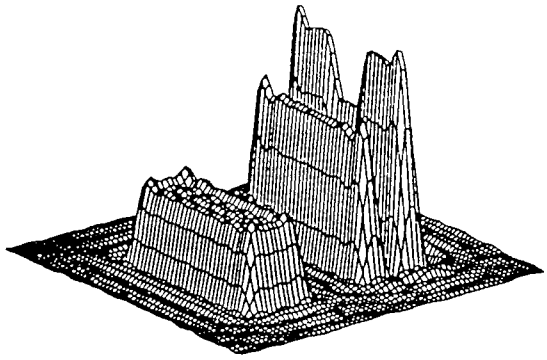

Figure 5a: Restoration using no preconditioner (108 iterations).

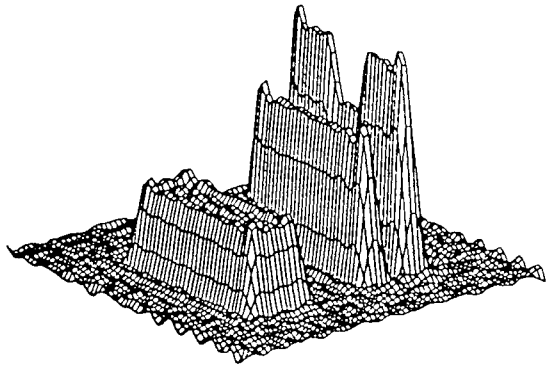

Figure 5b: Restoration using the Level-1 preconditioner (32 iterations).

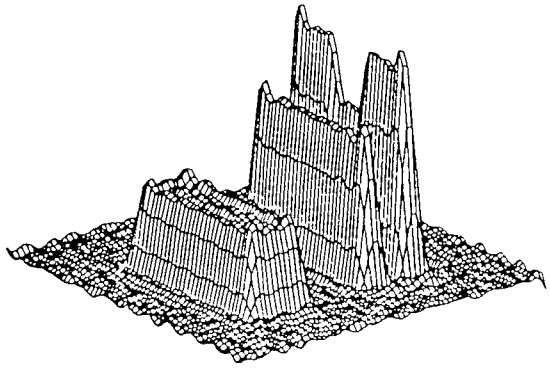

Figure 5c: Restoration using the Level-2 preconditioner (42 iterations).