In this paper, we consider solving the least squares problem $\min_{x} \| b - T x \|_2$ by using preconditioned conjugate gradient (PCG) methods, where $T$ is a large rectangular matrix which consists of several square block-Toeplitz–Toeplitz-block (BTTB) matrices and $b$ is a column vector. We propose a BTTB preconditioner to speed up the PCG method and prove that the BTTB preconditioner is a good preconditioner. We then discuss the construction of the BTTB preconditioner. Numerical examples, including image restoration problems, are given to illustrate the efficiency of our BTTB preconditioner. Numerical results show that our BTTB preconditioner is more efficient than the well-known Level-1 and Level-2 circulant preconditioners.

© 2009 Elsevier Inc. All rights reserved.
The idea of using circulant matrix to precondition Toeplitz systems $T_n x = b$ was first proposed by Strang [34] and Olkin [27] independently. Recall that an $n \times n$ circulant matrix is of the form $C_n = [c_{i-j}]_{n \times n}$ with $c_{i-j} = c_{n-j}$ for $j = 1, 2, \ldots, n - 1$, and all circulant matrices can be diagonalized by the Fourier matrix, see [13]. Several kinds of circulant preconditioners have been proposed and proved to be good preconditioners: for any given $\rho > 0$, all eigenvalues of $C_n^{-1}T_n$ lie in the interval $(1 - \varepsilon, 1 + \varepsilon)$ except at most $\rho$ outliers, where $\rho > 0$ is an integer independent of $n$, see for instance [4,6,8,10,11,34]. Therefore, the PCG methods with circulant preconditions converge very fast when they are used to solve Toeplitz systems.

Block circulant preconditioners for block systems have been intensively studied, see for instance [2,3,17,18], Chan and Olkin [12] and Holmgren and Otto [15], in solving noise reduction problems and hyperbolic differential equations, independently proposed using block circulant matrices to precondition BTTB matrices. Replacing each Toeplitz block by its optimal circulant preconditioner [11], we obtain a Level-1 circulant preconditioner $C_1(T_{mn})$, which is a block Toeplitz matrix with circulant blocks (BTCB). We can further approximate $C_1(T_{mn})$ by a block circulant matrix with circulant blocks (BCCB) to get a Level-2 circulant preconditioner $C_2(T_{mn})$. In [5], Chan et al. extended block circulant preconditioners to least squares problems. However, Serra Capizzano and Tyrtyshnikov first and subsequently Noutsos, Serra Capizzano, and Vassalos proved that it is impossible to obtain a strong clustering (the number of outliers is an unbounded function of the partial sizes $n$ and $m$) and it is impossible in general to get spectrally equivalent or essentially spectrally equivalent BCCB preconditioners, when the original BTTB sequence is asymptotically ill conditioned, see [31–33,23,24].

As alternatives to circulant matrices, Toeplitz matrices have also been proposed and analyzed as preconditioners for Toeplitz systems, see for instance [1,6,9,16,19,21,22,25,26,28]. Band Toeplitz preconditioner was first proposed and analyzed by Chan [1] and band BTTB preconditioners was first proposed and analyzed by Serra Capizzano [28]. In [6,16,19,20], inverse Toeplitz preconditioners have been studied.

In this paper, we study an inverse BTTB preconditioner for the BTTB least squares problem

$$\min_{x} \| b - T x \|_2,$$  \hspace{1cm} (2)

where $b$ is a $kmn \times 1$ vector and

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

with each $T^{(i)}$ being a BTTB($m, n$) matrix, $i = 1, \ldots, k$. Such kind of problems appear in image processing and time series analysis, etc., see for instance [7,14]. The least squares problem (2) is equivalent to the following normal equation

$$T^* T x = T^* b.$$  \hspace{1cm} (3)

Since BTTB least squares problems are usually very large and very ill-conditioned, one often solves such kind of problems by preconditioned iterative methods such as the preconditioned conjugate gradient (PCG) methods. It has been proved that certain block preconditioners with circulant structure are efficient ones, see for instance [5,29,30].

As in literature, we assume in this paper that the $i$th BTTB($m, n$) block $T^{(i)}$ of the least squares problem (2) is generated by a function $f^{(i)}(x, y) \in C_{2\pi \times 2\pi}$ (the set of all bivariate $2\pi$-periodic continuous functions), i.e., $T^{(i)} = T_{mn}(f^{(i)})$, $i = 1, 2, \ldots, k$. Let $g(x, y) = \sum_{i=1}^{k} |f^{(i)}(x, y)|^2$. We will use $T_{mn}(1/g)$ to precondition (3) and prove that $T_{mn}(1/g)$ is a good preconditioner: for any given $\varepsilon > 0$, the preconditioned matrix $T_{mn}(1/g)(T^* T)$ has at most $O(m) + O(n)$ eigenvalues lying outside the interval $(1 - \varepsilon, 1 + \varepsilon)$.

The outline of the paper is as follows. In Section 2, we propose using $T_{mn}(1/g)$ to precondition the normal equation (3) and study the spectral property of the preconditioned matrix $T_{mn}(1/g)(T^* T)$. In Section 3, we discuss how to approximate $T_{mn}(1/g)$ efficiently. Numerical examples and concluding remarks are given in Sections 4 and 5, respectively.
2. BTTB preconditioner

Toeplitz preconditioners have been successfully used to precondition Toeplitz matrices, see [6,19]. It has been proved that if \( f(x) \) is a \( 2\pi \)-periodic continuous function and \( f(x) > 0 \), then \( T_n[1/f] \) is a good preconditioner for \( T_n[f] \). That is, for any given \( \varepsilon > 0 \), there exists an integer \( \rho > 0 \) such that there are at most \( \rho \) eigenvalues of the preconditioned matrix \( T_n[1/f]T_n[f] \) not being in the interval \((1 - \varepsilon, 1 + \varepsilon)\) for all \( n \). Recently, the above result was extended to BTTB systems [20]: If \( f(x,y) \in C_{2\pi \times 2\pi} \) and \( f(x,y) > 0 \), then for any given \( \varepsilon > 0 \), there are at most \( O(m) + O(n) \) eigenvalues of the preconditioned matrix \( T_m[f]T_n[f] \) that are not in the interval \((1 - \varepsilon, 1 + \varepsilon)\) for all \( m,n \).

Now consider the BTTB least squares problem (2). Let the \( i \)th BTTB block \( T^{(i)} \) is generated by \( f^{(i)}(x,y) \in C_{2\pi \times 2\pi}, i = 1, 2, \ldots, k \), and let
\[
g(x,y) = \sum_{i=1}^{k} |f^{(i)}(x,y)|^2.
\]
Assume that \( g(x,y) > 0 \), we use the BTTB matrix \( T_{m,n}[1/g] \) to precondition the coefficient matrix of the normal equation (3), i.e.,
\[
\sum_{i=1}^{k} (T^{(i)})^* \cdot T^{(i)} = \sum_{i=1}^{k} T_{m,n}^*[f^{(i)}] T_{m,n}[f^{(i)}].
\]
In the following, we analyze the spectrum of the preconditioned matrix
\[
T_{m,n}[1/g] \sum_{i=1}^{k} T_{m,n}^*[f^{(i)}] T_{m,n}[f^{(i)}].
\]
We require the following lemmas.

**Lemma 2.1** [29]. Let \( f(x,y) \in C_{2\pi \times 2\pi} \). For any given \( \varepsilon > 0 \), there exist integers \( \rho, \sigma > 0 \), such that for \( m > \rho \) and \( n > \sigma \), we have
\[
T_{m,n}[f] T_{m,n}[f] = T_{m,n}[f]^2 + S_{m,n} + G_{m,n},
\]
where \( S_{m,n} \) and \( G_{m,n} \) are Hermitian, and
\[
\|S_{m,n}\|_2 < \varepsilon, \quad \text{rank}(G_{m,n}) \leq \rho m + \sigma n = O(m) + O(n).
\]

**Lemma 2.2** [20]. Let \( f(x,y) \in C_{2\pi \times 2\pi} \) be real-valued and \( f(x,y) > 0 \). For any given \( \varepsilon > 0 \), there exist integers \( \rho, \sigma > 0 \), such that for all \( n > \rho, m > \sigma \),
\[
T_{m,n}^{-1}[1/f] - T_{m,n}[f] = G_{m,n} + H_{m,n},
\]
where \( G_{m,n} \) and \( H_{m,n} \) are Hermitian, and
\[
\text{rank}(G_{m,n}) = O(m) + O(n), \quad \|H_{m,n}\|_2 < \varepsilon.
\]

The following Theorem 2.1 is the main result of this paper.

**Theorem 2.1.** Suppose \( f^{(i)}(x,y) \in C_{2\pi \times 2\pi}, i = 1, 2, \ldots, k \), and the function
\[
g(x,y) = \sum_{i=1}^{k} |f^{(i)}(x,y)|^2
\]
has no zeros. Then for any given \( \varepsilon > 0 \), there exist integers \( \rho, \sigma > 0 \) such that for all \( m > \rho, n > \sigma \), the matrix
\[
T_{m,n}[1/g] \sum_{i=1}^{k} T_{m,n}^*[f^{(i)}] T_{m,n}[f^{(i)}]
\]
has at most \( O(m) + O(n) \) eigenvalues lying outside the interval \((1 - \varepsilon, 1 + \varepsilon)\).
Proof. Obviously \( g(x, y) \in C_{2\pi \times 2\pi} \) and
\[
\forall x \in [0, 2\pi] \times [0, 2\pi], \quad g(x, y) > 0,
\]
Therefore, \( T_{m,n}[1/g] \) is Hermitian and positive definite. From Lemmas 2.2 and 2.1, it follows that for any given \( \epsilon > 0 \), there exist positive integers \( \rho \) and \( \sigma \) such that for all \( n > \rho, m > \sigma \):

(i) There exist Hermitian matrices \( G_{m,n} \) and \( H_{m,n} \) satisfying
\[
\text{rank}(G_{m,n}) = O(m) + O(n), \quad \|H_{m,n}\|_2 < \epsilon g_{\min}/2,
\]
such that
\[
T_{m,n}^{-1}[1/g] - T_{m,n}[g] = G_{m,n} + H_{m,n}.
\]
(ii) There exist Hermitian matrices \( U_{m,n} \) and \( S_{m,n} \) satisfying
\[
\text{rank}(U_{m,n}) = O(m) + O(n), \quad \|S_{m,n}\|_2 < \epsilon g_{\min}/2,
\]
such that
\[
T_{m,n}[g] = \sum_{i=1}^{k} T_{m,n}^*[f(\cdot)]^i T_{m,n}[f(\cdot)] + U_{m,n} + S_{m,n}.
\]
Therefore
\[
T_{m,n}^{-1}[1/g] = \sum_{i=1}^{k} T_{m,n}^*[f(\cdot)]^i T_{m,n}[f(\cdot)] = (G_{m,n} + U_{m,n}) + (H_{m,n} + S_{m,n}).
\]
Equivalently
\[
l_{mn} = T_{m,n}^{-1/2}[1/g] \left( \sum_{i=1}^{k} T_{m,n}^*[f(\cdot)]^i T_{m,n}[f(\cdot)] \right) T_{m,n}^{1/2}[1/g]
\]
\[
= T_{m,n}^{-1/2}[1/g](G_{m,n} + U_{m,n}) T_{m,n}^{1/2}[1/g] + T_{m,n}^{-1/2}[1/g](H_{m,n} + S_{m,n}) T_{m,n}^{1/2}[1/g]. \tag{5}
\]
Notice that
\[
\text{rank} \left( T_{m,n}^{-1/2}[1/g](G_{m,n} + U_{m,n}) T_{m,n}^{1/2}[1/g] \right) = \text{rank}(G_{m,n} + U_{m,n}) = O(m) + O(n)
\]
and
\[
\left\| T_{m,n}^{-1/2}[1/g](H_{m,n} + S_{m,n}) T_{m,n}^{1/2}[1/g] \right\|_2 \\
\leq \left\| T_{m,n}^{-1/2}[1/g] \right\|_2 \left( \|H_{m,n}\|_2 + \|S_{m,n}\|_2 \right) < 1/g_{\min} \cdot 2 \cdot \epsilon g_{\min}/2 = \epsilon.
\]
By using Weyl's Theorem (see for instance [3, p. 5]), it follows from (5) that there are at most \( O(m) + O(n) \) eigenvalues of \( T_{m,n}^{-1/2}[1/g](\sum_{i=1}^{k} T_{m,n}^*[f(\cdot)]^i T_{m,n}[f(\cdot)]) T_{m,n}^{1/2}[1/g] \) lying outside the interval \((1 - \epsilon, 1 + \epsilon)\). Finally, from
\[
T_{m,n}^{-1/2}[1/g] \left( \sum_{i=1}^{k} T_{m,n}^*[f(\cdot)]^i T_{m,n}[f(\cdot)] \right) T_{m,n}^{1/2}[1/g]
\]
\[
= T_{m,n}^{-1/2}[1/g] \left[ T_{m,n}[1/g] \left( \sum_{i=1}^{k} T_{m,n}^*[f(\cdot)]^i T_{m,n}[f(\cdot)] \right) \right] T_{m,n}^{1/2}[1/g].
\]
the result (4) follows. \( \square \)
From Theorem 2.1, we see that when the PCG method with the BTBB preconditioner \( T_{mn}[1/g] \) is used to solve BTBB least squares problems, the method converges very fast. Therefore, \( T_{mn}[1/g] \) is an efficient preconditioner.

3. The approximation of the BTBB preconditioner

In Section 2, we proved that \( T_{mn}[1/g] \) is an efficient preconditioner for the normal equation (3), where \( g(x,y) = \sum_{i=1}^{k} [f^{(i)}(x,y)]^2 \) with \( f^{(i)}(x,y) \) being the generating function of the BTBB block \( T^{(i)} \). However, in general it is difficult to obtain the exact Fourier coefficients of \( 1/g(x,y) \)

\[
q_{jl} = \frac{1}{4\pi^2} \int_0^{2\pi} \int_0^{2\pi} \frac{1}{g(x,y)} e^{-i(jx+ly)} \, dx \, dy, \quad j, l = 0, \pm 1, \pm 2, \ldots \tag{6}
\]

A simple way to compute approximate Fourier coefficients is applying the rectangular rule to the above integration

\[
q_{jl} \approx p_{jl} = \frac{1}{mn} \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} \frac{1}{g(2u\pi/m, 2v\pi/n)} e^{-2\pi i(ju/m - 2\pi i/v/n)} , \quad j, l = 0, \pm 1, \pm 2, \ldots \tag{7}
\]

Let \( P = [p_{jl}]_{j=0,1,...,n-1} \) and \( G = [1/g(2u\pi/m, 2v\pi/n)]_{u=0,1,...,m-1} \). It is easily seen that

\[
P = \left( \frac{1}{\sqrt{m}} \mathbf{F} \right) \mathbf{G} \left( \frac{1}{\sqrt{n}} \mathbf{F}^\dagger \right), \tag{8}
\]

where \( \mathbf{F} \) is the Fourier matrix of order \( p \). Therefore, we can obtain \( P \), which contains the required approximate Fourier coefficients of \( 1/g(x,y) \), in \( O(mn \log(mn)) \) operations by using 2-D FFT to \( G \). About the accuracy of the approximate coefficients \( p_{jl} \), we have the following result, see [20].

**Theorem 3.1.** Let \( 1/g(x,y) \) be in the Wiener class, that is, the Fourier coefficients of \( 1/g(x,y) \) satisfy \( \sum_{j=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} |q_{jl}| < \infty \), where \( q_{jl} \) are given by (6). Let

\[
p_{m/2-1,n/2-1}(x,y) = \sum_{j=-(m/2-1)}^{m/2-1} \sum_{l=-(n/2-1)}^{n/2-1} p_{jl} e^{ijx+ily}
\]

(assume that \( m \) and \( n \) are even) where \( p_{jl} \) are given by (7). Then

\[
\left| p_{m/2-1,n/2-1}(x,y) - \frac{1}{g(x,y)} \right| \leq 2 \sum_{|j| > m/2} \sum_{|l| > n/2} |q_{jl}|.
\]

That is, the sequence of trigonometric polynomials \( p_{m/2,n/2}(x,y) \) converges to \( 1/g(x,y) \) uniformly as \( m \) and \( n \) tend to infinity.

We would like to note since the Wiener class is an algebra we can simplify the assumption. That is, instead of asking that \( 1/g \) is in the Wiener class they can ask that \( g \) belongs to the Wiener class. The latter is much simpler to check. Moreover, Theorem 3.1 can be extended to simply continuous positive symbols following the Korovkin approach [29] and using the same coefficients.

In general, we do not have an explicit formula for \( g(x,y) \) since we do not have explicit formulas for \( f^{(i)}(x,y) \), \( i = 1, 2, \ldots, k \). We use the Cesáro sum to approximate \( f^{(i)}(x,y) \):

\[
f^{(i)}(x,y) \approx f^{(i)}_{m-1,n-1}(x,y) = \sum_{j=-(m-1)}^{m-1} \sum_{l=-(n-1)}^{n-1} w_{jl}^{(i)} e^{ijx+ily},
\]

where

\[
w_{jl}^{(i)} = \left( 1 - \frac{|j|}{m} \right) \left( 1 - \frac{|l|}{n} \right) f_{jl}^{(i)}.
\]

Then we approximate $g(x, y)$ by

$$g_{m-1,n-1}(x, y) = \frac{1}{k} \sum_{i=1}^{k} |f_{m-1,n-1}(x, y)|^2.$$  

To compute $\tilde{G} = \{1 / g_{m-1,n-1}(2m\pi / m, 2m\pi / n)\}_{m=0,1,...,m-1}$, efficiently, we only require to compute $F(i) = \{f_{m-1,n-1}(2m\pi / m, 2m\pi / n)\}_{m=0,1,...,m-1}$. It is easily seen that

$$f_{m-1,n-1}(2m\pi / m, 2m\pi / n) = \sum_{j=1}^{m-1} \sum_{l=1}^{n-1} w_{ij} e^{2\pi i (aj + vl) / n},$$

where

$$w_{ij} = \begin{cases} w_{0,0}, & j = 0, l = 0, \\ w_{0,j} + w_{j,m,0}, & j = 1, 2, \ldots, m - 1, l = 0, \\ w_{0,j} + w_{0,j-n}, & j = 0, l = 1, 2, \ldots, n - 1, \\ w_{j,j} + w_{j,m} + w_{j,l-n}, & j > 1, l > 1. \\ \end{cases}$$  

Let $\tilde{W}(i) = \{\tilde{w}_{ij} = 0, ..., n-1\}$, then $F(i) = (\sqrt{\tilde{m}e_n}) \tilde{W}(i)(\sqrt{\tilde{m}e_n})$. It follows that we can obtain $F(i)$ by applying 2-D inverse FFT to $\tilde{W}(i)$ in $O(mn \log(mn))$ operations.

Here we make a note on the approximation $g_{m,n}(x, y)$ of $g(x, y)$. Assume that each $f_{m,n}(x, y)$ is in Wiener class, $i = 1, 2, \ldots, k$, then

$$\lim_{m,n \to \infty} \|f_{m,n}^{(i)} - f^{(i)}\| = 0, \quad i = 1, 2, \ldots, k,$$

where $\|f(x, y)\| = \max_{x,y \in [0,2\pi] \times [0,2\pi]} |f(x, y)|$. It follows that

$$\lim_{m,n \to \infty} \|g_{m,n} - g\| = 0.$$  

Therefore, for all given $\epsilon > 0$, there exist integers $M, N > 0$ such that $\|g_{M,N} - g\| < \epsilon$ and $\min_{x,y \in [0,2\pi] \times [0,2\pi]} g_{M,N}(x, y) > 0$. Since $g_{M,N}(x, y)$ is a trigonometric polynomial, it is very smooth. It follows that $1 / g_{M,N}(x, y)$ is also a very smooth function, and hence $1 / g_{M,N}(x, y)$ is in Wiener class. Thus, by virtue of Theorems 3.1 and 2.1, $T_{m,n}[1 / g_{M,N}]$ is a good preconditioner.

To end this section, we give the costs per iteration of the PCG methods with the BTTB preconditioner $T_{m,n}[P_{m/2-1,n/2-1}]$ (PCG_{BTTB}), with the Level-1 circulant preconditioner (PCG_{L1}), and with the Level-2 circulant preconditioner (PCG_{L2}), respectively.

1. The matrix-vector multiplication $T_{m,n}[P_{m/2-1,n/2-1}]y$ can be done in $c \cdot 9mn / 4 \cdot \log(9mn / 4)$ (c is a constant related to the cost of FFT) operations by using FFTs by first embedding $T_{m,n}[P_{m/2-1,n/2-1}]$ into a $3m/2$ by $3m/2$ block circulant matrix with $3n/2$ by $3n/2$ circulant blocks. Similarly, the computation of $\sum_{k=1}^{m} (T^{(i)})^* (T^{(i)}) y$ requires $c \cdot 8km \log(4mn)$ operations. Thus, the cost of computing $T_{m,n}[P_{m/2-1,n/2-1}]y$ is about $9(32k)$ times that of computing $\sum_{k=1}^{m} (T^{(i)})^* (T^{(i)}) y$. Therefore, the cost per iteration of PCG_{BTTB} is about $(1 + 9/(32k))$ times that of the CG method.

2. The Level-1 circulant preconditioner $\sum_{k=1}^{m} (c_1(T^{(i)}))^* c_1(T^{(i)})$ can be decomposed as
The number of iterations and the CPU timings for Example 1 (zero initial guess).

Table 1

<table>
<thead>
<tr>
<th>m = n</th>
<th>Number of iterations</th>
<th>CPU timings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CG</td>
<td>PCGL₁</td>
</tr>
<tr>
<td>32</td>
<td>209</td>
<td>9</td>
</tr>
<tr>
<td>64</td>
<td>395</td>
<td>9</td>
</tr>
<tr>
<td>128</td>
<td>676</td>
<td>10</td>
</tr>
<tr>
<td>256</td>
<td>&gt;1000</td>
<td>14</td>
</tr>
</tbody>
</table>

\[(l_m \otimes F_n)^* \left( \sum_{i=1}^{k} (A(i)^* A(i)) \right) (l_m \otimes F_n),\]

where \(\otimes\) denotes the Kronecker tensor product and \(A(i), i = 1, 2, \ldots, n,\) are block matrices with diagonal blocks. To speed up the preconditioning step, R. Chan, Nagy, and Plemmons proposed applying Cholesky factorization to \(\sum_{i=1}^{k} (A(i)^* A(i))\) [5]. Thus the preconditioning step requires \((c \cdot mn \log m + nnm)\) operations.

3. Since the Level-2 circulant preconditioner \(\sum_{i=1}^{k} (\gamma_2(T(i)))^* \gamma_2(T(i))\) is a BCCB matrix, it can be easily checked that for PCGL₂, the cost per iteration is about \((1 + 1/(8k))\) times that of the CG method.

Therefore, the cost per iteration of PCG_BTTB is about \((32k + 9)/(32k + 4)\) times (about 1.14 for \(k = 1\) and 1.07 for \(k = 2\)) that of PCGL₂, and much smaller than that of PCGL₁.

4. Numerical examples

In this section, the performance of four CG type methods are compared: the CG method (CG), the PCG method with Level-1 circulant preconditioner (PCGL₁), the PCG method with Level-2 circulant preconditioner (PCGL₂), and the PCG method with our BTTB preconditioner (PCG_BTTB).

Three examples are tested. Examples 1 and 2 are from [5], with some extension. Example 3 solves Example 2 by using another regularization operator. All numerical tests were run using Matlab on a PC with Pentium 4/3.00 GHz of CPU and 448 MB of memory.

Example 1 [5]. Consider rectangular matrices consisting of four BTTB matrices whose diagonals are given by the following generating sequences:

\[
\begin{align*}
    i_{j}^{(1)} &= \frac{1}{(|l| + 1)(|j| + 1)^{1+0.1(|l|+1)}}, \\
    i_{j}^{(2)} &= \frac{1}{(|l| + 1)^{1.1}(|j| + 1)^{1+0.1(|l|+1)}}, \\
    l_{j}^{(3)} &= \frac{1}{(|l| + 1)^{1.1} + (|j| + 1)^{1.1}}, \\
    l_{j}^{(4)} &= \frac{1}{(|l| + 1)^{2.1} + (|j| + 1)^{2.1}}.
\end{align*}
\]

\(j, l = 0, \pm 1, \pm 2, \ldots.\) The right hand side vector \(T^* b\) of the normal equation is the vector of all ones.

For each case, two initial guesses were tested: the zero vector and a random vector (the same for all methods). The stopping criterion is \(\|r_q\|_2/\|r_0\|_2 < 10^{-7}\), where \(r_q\) is the (normal equation) residual vector at the \(q\)th iteration.

When the zero initial guess is used, the PCG method PCGL₁ converges the fastest in the sense of number of iterations, see Table 1. The PCG methods PCGL₂ and PCG_BTTB require a few more iterations, and the CG method requires a lot of iterations. However, we also see that PCG_BTTB is the fastest among the four methods. Moreover, as the size of the problem increases, the advantage of PCG_BTTB becomes more significant. This observation consists with the fact that the cost per iteration of PCG_BTTB is much smaller than that of PCGL₁ and is only a bit larger than that of PCGL₂.
Table 2
The number of iterations for Example 1 (random initial guess).

<table>
<thead>
<tr>
<th>m = n</th>
<th>Number of iterations</th>
<th>CPU timings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CG</td>
<td>PCGL₁</td>
</tr>
<tr>
<td>32</td>
<td>201</td>
<td>15</td>
</tr>
<tr>
<td>64</td>
<td>324</td>
<td>16</td>
</tr>
<tr>
<td>128</td>
<td>504</td>
<td>17</td>
</tr>
<tr>
<td>256</td>
<td>733</td>
<td>20</td>
</tr>
</tbody>
</table>

Fig. 1. Original simulated images.

Table 2 shows the number of iterations and the CPU timings of the four methods with the same random initial guess (we use the Matlab command “rand(’seed’, 37’)”). In this settings, the methods PCGL₁ and PCGBTTB require about the same number of iterations, and PCGL₂ requires much more iterations. Again, the method PCGBTTB is the fastest among the four methods.

Example 2 [5]. Consider the image restoration problem

\[ \mathbf{g} = T \mathbf{f} + \eta \]  

where \( T \) is the BTTB \((m, n)\) matrix with diagonals being given by

\[ t_{ij} = \begin{cases} \exp(-0.1j^2 - 0.1l^2), & -8 \leq j, l \leq 8, \\ 0, & \text{otherwise}, \end{cases} \]  

\( \mathbf{f} \) is the \( mn \times 1 \) vector obtained from the true image by row ordering, \( \mathbf{g} \) presents the blurred, noisy image, and \( \eta \) presents the noise. We tested simulation images of sizes: \( 64 \times 64, 128 \times 128, 256 \times 256 \), see Fig. 1(a)–(c), respectively. The observed image \( \mathbf{g} \) is constructed by \( \mathbf{g} = T \mathbf{f} + \eta \), where the noise vector \( \eta \) is a vector with random entries chosen from a normal distribution with the mean 0 (\( \eta \) is scaled so that \( \| \eta \|_2 / \| T \|_2 = 10^{-3} \)).

As in [5], we solve the following regularized normal equation:

\[ (\mu^2 L^*L + T^*T)\mathbf{f} = T^* \mathbf{g}. \]  

where \( L = \mathbf{l}_m \) and \( \mu = 0.1 \). In the numerical tests, the initial guess is set to the zero vector and the stopping criterion is set to \( \| \mathbf{r}_q \|_2 / \| \mathbf{r}_0 \|_2 < \varepsilon = 10^{-7} \), which is different from the one proposed in [5] where \( \varepsilon = 10^{-3} \) (our numerical tests show that the latter cannot guarantee the optimal accuracy of computed solutions).

The number of iterations and the CPU timings of the four methods are listed in Table 3. From Table 3, we see that the method PCGBTTB is much faster than other three methods except the case \( m = n = 64 \). Moreover, except the case \( m = n = 64 \), the method PCGBTTB requires the minimal iterations. The restored images are shown in Fig. 2(a)–(c), respectively. We see that the restored images are close to the original simulated images.
Table 3
Number of iterations and CPU timings for Example 2.

<table>
<thead>
<tr>
<th>m = n</th>
<th>Number of iterations</th>
<th>CPU timings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CG</td>
<td>PCGL₁</td>
</tr>
<tr>
<td>64</td>
<td>605</td>
<td>160</td>
</tr>
<tr>
<td>128</td>
<td>853</td>
<td>265</td>
</tr>
<tr>
<td>256</td>
<td>913</td>
<td>342</td>
</tr>
</tbody>
</table>

Fig. 2. Restored images: \( L \) is the identity matrix and \( \mu = 0.1 \).

Table 4
Number of iterations and CPU timings for Example 3.

<table>
<thead>
<tr>
<th>m = n</th>
<th>Number of iterations</th>
<th>CPU timings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CG</td>
<td>PCGL₁</td>
</tr>
<tr>
<td>64</td>
<td>518</td>
<td>143</td>
</tr>
<tr>
<td>128</td>
<td>706</td>
<td>167</td>
</tr>
<tr>
<td>256</td>
<td>704</td>
<td>170</td>
</tr>
</tbody>
</table>

Fig. 3. Restored images: \( L \) is given by (14) and \( \mu = 0.09 \).

Example 3. Consider solving Example 2 by using another regularization operator. Let \( F = [f_{ij}]_{i=1,...,n} \) be the image to be restored. We define \( L \) as

\[
\|LF\|^2 = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} (u_{i+1,j} - u_{ij})^2 + (v_{i+1,j} - v_{ij})^2 + \sum_{j=1}^{n-1} (u_{m,j+1} - u_{mj})^2 + \sum_{i=1}^{m-1} (v_{i,n+1} - v_{ni})^2.
\]

(14)

The regularization parameter is set to \( \mu = 0.09 \), which was chosen based on several tests.
In this case, $L$ is not a BTTB($m, n$) matrix. In fact, $L^*L = \tilde{L} - D$, where $\tilde{L}$ is symmetric and positive definite BTTB matrix and $D$ is a matrix of rank $2n$. We constructed the Level-1 circulant preconditioner, the Level-2 circulant preconditioner, and the BTTB preconditioner for $(\mu^*L + T^*T)$ instead of $(\mu^*L^*L + T^*T)$ (cf. (13)).

The number of iterations and the CPU timings of the four methods are listed in Table 4. From Table 4, we see again that the method $PCG_{BTTB}$ is much faster than other three methods and also converges the fastest in the sense of the number of iterations. The restored images are shown in Fig. 3(a)–(c), respectively.

5. Concluding remarks

We proposed a BTTB matrix as a preconditioner for BTTB least squares problems and proved that the BTTB preconditioner is efficient one. We also discussed how to construct an approximation of the BTTB preconditioner efficiently by using trigonometric polynomial approximation technique. Numerical results show that our BTTB preconditioner is more efficient than Level-1 and Level-2 circulant preconditioners. The BTTB preconditioner proposed in this paper is well-defined only for generating functions without zeros. We will consider generating functions with zeros in our future research.

Acknowledgements

We thank the anonymous referees for their detailed and valuable comments and suggestions.

References