Fast Construction of Optimal Circulant Preconditioners for Matrices from Fast Dense Matrix Method

Raymond H. Chan* Wing-Fai Ng† Hai-Wei Sun‡

Abstract

In this paper, we consider solving non-convolution type integral equations by the preconditioned conjugate gradient method. The fast dense matrix method is a fast multiplication scheme that provides a dense discretization matrix A approximating a given integral equation. The dense matrix A can be constructed in $O(n)$ operations and requires only $O(n)$ storage where n is the size of the matrix. Moreover, the matrix-vector multiplication $A\mathbf{x}$ can be done in $O(n \log n)$ operations. Thus if the conjugate gradient method is used to solve the discretized system, the cost per iteration is $O(n \log n)$ operations. However, for some integral equations, such as the Fredholm integral equations of the first kind, the system will be ill-conditioned and therefore the convergence rate of the method will be slow. In these cases, preconditioning is required to speed up the convergence rate of the method. A good choice of preconditioner is the optimal circulant preconditioner which is the minimizer of $\|C - A\|_F$ in Frobenius norm over all circulant matrices C. It can be obtained by taking arithmetic averages of all the entries of A and therefore the cost of constructing the preconditioner is of $O(n^2)$ operations for general dense matrices. In this paper, we develop an $O(n \log n)$ method of constructing the preconditioner for dense matrices A obtained from the fast dense matrix method. Application of these ideas to boundary integral equations from potential theory will be given. These equations are ill-conditioned whereas their optimal circulant preconditioned equations will be well-conditioned. The accuracy of the approximation A, the fast construction of the preconditioner and the fast convergence of the preconditioned systems will be illustrated by numerical examples.

AMS(MOS) subject classifications. 45B05, 65F10, 65R20.

Key Words. Integral equations, circulant preconditioners, conjugate gradient method.

1 Introduction

Circulant matrices are matrices that have constant diagonals and that the first entry of each column is the last entry of its preceding column. More precisely, each column in the matrix is obtained by a cyclic shift of its preceding column. For an n-by-n matrix B, the optimal circulant preconditioner $c(B)$ of B is defined to be the minimizer of $\|C - B\|_F$ over all n-by-n circulant

*Department of Mathematics, Chinese University of Hong Kong, Shatin, Hong Kong. Research supported in part by HK Research Grant Council grant no. CUHK4207/97P.

†Department of Mathematics, Chinese University of Hong Kong, Shatin, Hong Kong.

‡Department of Mathematics and Physics, Guangdong University of Technology, Guangzhou, People's Republic of China. The research of this author is supported by Guangdong Provincial Natural Science Foundation of China No. 974176.

matrices C, see T. Chan [10]. Here $\|\cdot\|_F$ denotes the Frobenius norm. Since $c(B)$ is a circulant matrix, it is determined uniquely by its first column which can be obtained easily by taking the arithmetic average of the entries $b_{\alpha,\beta}$ of B. More precisely, the entries $\{c_\delta\}_{\delta=1}^n$ in the first column of $c(B)$ are given by

$$c_{\delta+1} = \frac{1}{n} \sum_{\alpha-\beta=\delta(\mathrm{mod}\ n)} b_{\alpha,\beta}, \quad \delta = 0, \ldots, n-1, \tag{1}$$

see Tyrtyshnikov [17].

Using the circulant structure of $c(B)$, the inverse $[c(B)]^{-1}$ of $c(B)$ and the matrix-vector multiplication $[c(B)]^{-1}\mathbf{x}$ for any vector $\mathbf{x}$ can be obtained in $O(n \log n)$ operations by using fast Fourier transforms, see for instance Chan and Ng [8]. Moreover, $c(B)$ is positive-definite whenever B is, see Tyrtyshnikov [17]. This makes $c(B)$ a very attractive choice of preconditioner in the preconditioned conjugate gradient method for solving the system $B\mathbf{y} = \mathbf{b}$. For then, if B is positive-definite, the preconditioned matrix is positive-definite. Moreover, the cost of multiplying $[c(B)]^{-1}$ to a vector, which is required in each iteration of the method, can be obtained in $O(n \log n)$ operations by using fast Fourier transforms.

Optimal circulant preconditioners have also been proposed and used successfully in solving convolution type integral equations, see [12, 4]. The discrete matrices from these integral equations are Toeplitz matrices if the rectangular quadrature rule is used. For non-convolution type integral equations, where the discrete matrices are no longer Toeplitz, convergence analysis of optimal circulant preconditioners has also been studied, see [6, 9]. For example, for boundary integral equations arising from potential equations, which are ill-conditioned, non-convolution type integral equations with condition number increasing like $O(n)$, the preconditioned systems have been shown to be well-conditioned, see Chan, Sun and Ng [9] and also §5.

However, there are two main difficulties in using circulant preconditioned conjugate gradient methods for non-convolution type integral equations. The first one is that the discretization matrix B corresponding to the integral equation is dense. Hence multiplying B to a vector, which is required in each iteration of the conjugate gradient method, is of $O(n^2)$ complexity. The other difficulty is that since B is dense, forming the optimal circulant preconditioner $c(B)$ using (1) will require $O(n^2)$ operations in general. In this paper, we will address the second difficulty.

To overcome the first difficulty, a number of fast multiplication schemes have been developed in recent years, see for instance [13, 16, 3, 1, 18]. These methods try to obtain an approximation A to the given integral equation such that the matrix-vector multiplication of A with any vector can be done in $O(n)$ or $O(n \log n)$ operations, depending on the smoothness of the kernel function of the integral equation. In Chan, Lin and Ng [7], we have proposed a fast dense matrix method for approximating integral equations. The method is basically the same as the skeleton method of Tyrtyshnikov [18] applied to a partition suggested in Alpert *et. al.* [1]. The approximation matrix A is a dense matrix which can be obtained in $O(n)$ operations and only $O(n)$ storage is required. Moreover, the matrix-vector multiplication $A\mathbf{x}$ can be done in $O(n \log n)$ operations.

To deal with the second difficulty mentioned above, we will develop in this paper a fast algorithm for constructing the optimal circulant preconditioner $c(A)$ for matrices A that are obtained from our fast dense matrix method. Using the special structure of our A, the circulant matrix $c(A)$ can be obtained in $O(n \log n)$ operations. Thus, the construction of the discretization matrix A and its circulant preconditioner $c(A)$, and the cost of multiplying A or $[c(A)]^{-1}$

to any vector can all be done in $O(n \log n)$ operations. Hence used in conjunction with our fast dense matrix method, the circulant preconditioned conjugate gradient method for integral equations requires only $O(n \log n)$ operations per iteration.

To illustrate the efficiency of our construction and the effect of using circulant preconditioners, we will apply these ideas in solving first kind integral equations from potential equations. These equations are known to be ill-conditioned with condition number growing like $O(n)$. Chan, Sun and Ng [9] have shown however that the problem becomes well-conditioned if it is preconditioned by optimal circulant preconditioners. In particular, the preconditioned system converges in a fixed finite number of iterations independent of the size of the discretized system. Thus, if the system is solved by using preconditioned conjugate gradient method coupled with our fast dense matrix method, the total cost of solving the system is of the order $O(n \log n)$ operations.

The outline of the paper is as follows. In §2, we briefly recall the main concept of the fast dense matrix method. Preliminary lemmas that are useful in computing the arithmetic average of diagonals of matrices are given in §3. In §4, we give an $O(n \log n)$ algorithm for constructing the optimal circulant preconditioners. Finally, in §5, we apply our ideas in solving boundary integral equations from potential equations where the use of circulant preconditioners will speed up the convergence.

2 Fast Dense Matrix Method

In this section, we give a brief introduction of the fast dense matrix method proposed in Chan, Lin and Ng [7]. In the following, n is the size of the discretization matrix under consideration, i.e. $1/n$ is proportional to the the mesh size with which we discretize the integral equation. We will set $n = k \cdot 2^l$, where k is a fixed small integer that depends on the smoothness of the kernel function of the given integral equation.

The method in [7] approximates a given integral equation or its discretization matrix by sum of low rank matrices using the partition suggested in Alpert *et. al.* [1]. It is basically the same as the skeleton method [18] applied to such a partition. With the partition, B is cut into blocks of different sizes. The blocks near the main diagonal are of size k-by-k, those next remote are of size $2k$-by-$2k$, and so forth up to the largest blocks of size $2^{l-2}k$-by-$2^{l-2}k$. By grouping blocks of the same size into one matrix, we can express the matrix B as

$$B = B^{(0)} + B^{(1)} + \cdots + B^{(l-2)}, \tag{2}$$

where $B^{(\mu)}$ consists only of blocks of size $2^\mu k$-by-$2^\mu k$. As an illustration, for $l = 5$, $B^{(2)}$ has the structure as shown in Figure 1, where each $B^{(2,\nu)}$ is a $4k$-by-$4k$ matrix. We can easily check that the number of nonzero blocks in $B^{(\mu)}$ (see Figure 1) is given by

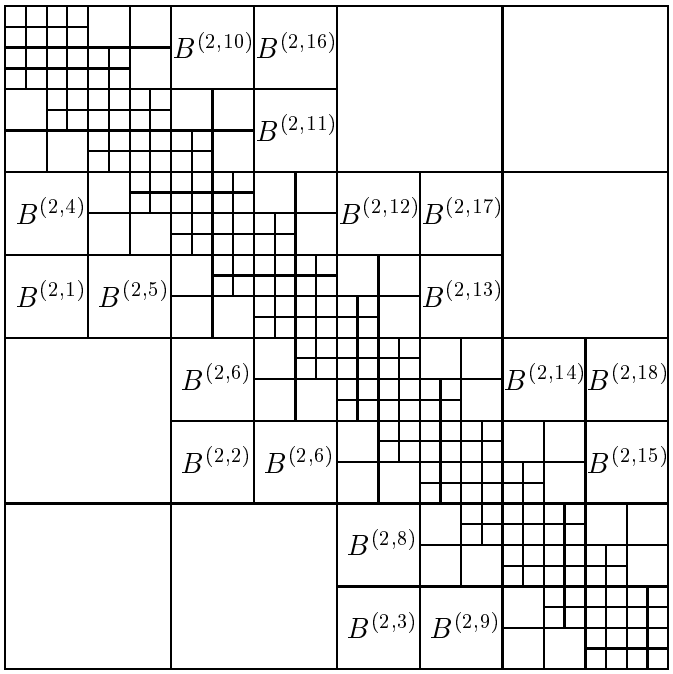
$$\nu_\mu = \begin{cases} 6 \cdot 2^l - 8 & \mu = 0, \\ 6(2^{l-1-\mu} - 1) & \mu = 1, \cdots, l-2. \end{cases} \tag{3}$$

We will denote these nonzero blocks by $B^{(\mu,\nu)}$, $\nu = 1, \cdots, \nu_\mu$.

Our approximation matrix A of B is obtained by approximating each block $B^{(\mu,\nu)}$ in $B^{(\mu)}$ by a rank k matrix $A^{(\mu,\nu)}$ by using polynomial approximation. More precisely, the approximation matrix A is also cut into blocks of sizes the same as that of B. Then A is given by

$$A = A^{(0)} + A^{(1)} + \cdots + A^{(l-2)}, \tag{4}$$

$B =$

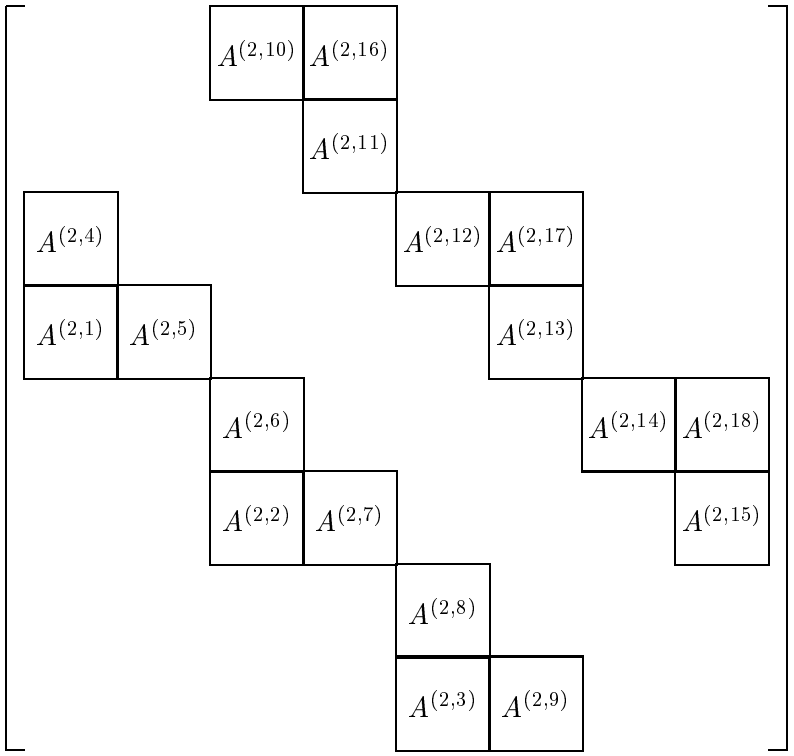


.

Figure 1. Partition of the matrix B.

with each $A^{(\mu)}$, $\mu = 0, \ldots, l-2$, consists only of blocks of size $2^\mu k$-by-$2^\mu k$, the same size as $B^{(\mu)}$. As an illustration, for $l = 5$, $A^{(2)}$ is of the form

$A^{(2)} =$



. (5)

Here each of the block $A^{(\mu,\nu)}$ is of size $2^\mu k$-by-$2^\mu k$ and is a rank k matrix of the form

$$A^{(\mu,\nu)} = (P^{(\mu)})^T \Lambda^{(\mu,\nu)} P^{(\mu)}, \tag{6}$$

where $\Lambda^{(\mu,\nu)}$ is a k-by-k sampling matrix from $B^{(\mu,\nu)}$, and $P^{(\mu)}$ is a k-by-$2^\mu k$ polynormial interpolation matrix which is the same for all $\nu = 1, \ldots, \nu_\mu$. For $\mu = 0$, $P^{(0)}$ is just the k-by-k identity matrix. Using (6) and the block structure of $A^{(\mu)}$ as depicted in (5), we see that $A^{(\mu)}$ can be written as

$$A^{(\mu)} = \left[I_{2^{l-\mu}} \otimes (P^{(u)})^T \right] \Lambda^{(\mu)} \left[I_{2^{l-\mu}} \otimes P^{(u)} \right]. \tag{7}$$

Here $I_{2^{l-\mu}}$ is the identity matrix of size $2^{l-\mu}$, $\otimes$ is the Kronecker tensor product and $\Lambda^{(\mu)}$ is a matrix having the same block structure as $A^{(\mu)}$, except that the blocks $\Lambda^{(\mu,\nu)}$ in $\Lambda^{(\mu)}$ are of size k. As an illustration, the matrix $A^{(2)}$ for $l = 5$ can be written as (cf. (5)):

$$A^{(2)} = \left[I_8 \bigotimes \left(P^{(2)} \right)^T \right] \begin{bmatrix}
 & & & & \Lambda^{(2,10)} & \Lambda^{(2,16)} & & & \\
 & & & & & \Lambda^{(2,11)} & & & \\
\Lambda^{(2,4)} & & & & & & \Lambda^{(2,12)} & \Lambda^{(2,17)} & \\
\Lambda^{(2,1)} & \Lambda^{(2,5)} & & & & & & \Lambda^{(2,13)} & \\
 & & \Lambda^{(2,6)} & & & & & & \Lambda^{(2,14)} & \Lambda^{(2,18)} \\
 & & \Lambda^{(2,2)} & \Lambda^{(2,7)} & & & & & & \Lambda^{(2,15)} \\
 & & & & \Lambda^{(2,8)} & & & & & \\
 & & & & \Lambda^{(2,3)} & \Lambda^{(2,9)} & & & &
\end{bmatrix} \left[I_8 \bigotimes P^{(2)} \right], \tag{8}$$

where each $\Lambda^{(2,v)}$ is a k-by-k matrix.

Combining (4) and (7), we then have

$$A = \sum_{\mu=0}^{l-2} A^{(\mu)} = \sum_{\mu=0}^{l-2} \left[I_{2^{l-\mu}} \otimes (P^{(\mu)})^T \right] \Lambda^{(\mu)} \left[I_{2^{l-\mu}} \otimes P^{(\mu)} \right]. \tag{9}$$

Thus for the computation of the matrix-vector product $A\mathbf{x}$ using (9), it suffices to form and store $\Lambda^{(\mu)}$ and $P^{(\mu)}$ for $\mu = 0, \ldots, l-2$ only. As shown in Chan, Lin and Ng [7], these matrices can be constructed in $O(n)$ operations and requires $O(n)$ storage, and the cost of the matrix-vector multiplication $A\mathbf{x}$ using (9) is of order $O(n \log n)$ operations.

In §4, we will discuss an $O(n \log n)$ algorithm for forming the optimal circulant preconditioner $c(A)$ of A using the decomposition in (9). In the next section, we give some lemmas that are useful in computing the arithmetic averages of diagonals of a given matrix.

3 Preliminary Lemmas

Given an n-by-n matrix B with entries $b_{\alpha,\beta}$, we define the *diagonal sums* $\{d_\delta\}_{\delta=-(n-1)}^{n-1}$ of B to be the sum along each of the diagonals of B. More precisely,

$$d_\delta \equiv \begin{cases} \sum\limits_{\substack{\alpha=\delta+1 \\ n+\delta}}^{n} b_{\alpha,\alpha-\delta}, & 0 \le \delta < n, \\ \sum\limits_{\alpha=1} b_{\alpha,\alpha-\delta}, & 0 \le -\delta < n. \end{cases} \tag{10}$$

Thus d_0 is the sum of the main diagonal entries of B, d_1 is the sum of the entries on the sub-diagonal and d_{-1} is the sum of the entries on the super-diagonal. We note that once we have the diagonal sums of B, then $c(B)$ can be obtained in $O(n)$ operations. In fact, by comparing (1) and (10), the following lemma follows.

Lemma 1 *Let $\{d_\delta\}_{\delta=-(n-1)}^{n-1}$ be the diagonal sums of B. Then the first column entries $\{c_\delta\}_{\delta=1}^{n}$ of $c(B)$ are given by $c_1 = d_0$ and*

$$c_{\delta+1} = \frac{1}{n}(d_\delta + d_{\delta-n}), \quad \delta = 1, \ldots, n-1.$$

In view of (6) and (9), to form $c(A)$ of A given in (9), we need to know how to form the diagonal sums of $A^{(\mu,\nu)}$ in (6). It is because they are the fundamental building blocks of $A^{(\mu)}$ and hence of A. In the following, we consider the cost and storage requirement for forming the diagonal sums of matrices of the form given in (6). The results are needed in §4 when we construct the optimal circulant preconditioner $c(A)$ for A. We begin with the complexity counts of forming diagonal sums for rank 1 matrices.

Lemma 2 *Let $\mathbf{p} = (p_1, \ldots, p_m)$ and $\mathbf{q} = (q_1, \ldots, q_m)$. Then the diagonal sums of the m-by-m matrix $\mathbf{p}^t\mathbf{q}$ can be obtained in $O(m \log m)$ operations and the storage required is $O(m)$.*

Proof: Recall from (10) that the diagonal sums of $\mathbf{p}^t\mathbf{q}$ are given by

$$d_\delta = \begin{cases} \sum\limits_{\substack{\alpha=\delta+1 \\ m+\delta}}^{m} p_\alpha q_{\alpha-\delta}, & 0 \le \delta < m, \\ \sum\limits_{\alpha=1} p_\alpha q_{\alpha-\delta}, & 0 \le -\delta < m. \end{cases}$$

In matrix terms, this amounts to

$$\begin{pmatrix} p_m & & & & 0 \\ p_{m-1} & p_m & & & \\ \vdots & \ddots & \ddots & & \\ p_2 & \vdots & \ddots & \ddots & \\ p_1 & p_2 & \vdots & \ddots & p_m \\ & p_1 & \ddots & \vdots & p_{m-1} \\ & & \ddots & \ddots & \vdots \\ & & & \ddots & p_2 \\ 0 & & & & p_1 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ \vdots \\ q_m \end{pmatrix} = \begin{pmatrix} d_{m-1} \\ d_{m-2} \\ \vdots \\ d_1 \\ d_0 \\ d_{-1} \\ \vdots \\ d_{-(m-2)} \\ d_{-(m-1)} \end{pmatrix}, \tag{11}$$

where the matrix is a column circulant matrix.

It is easy to augment the column circulant matrix to make it a square circulant matrix (which in fact is determined uniquely by its first column). The diagonal sums $\{d_j\}_{j=-m+1}^{m-1}$, i.e. the right hand side vector in (11), can be obtained by multiplying the augmented square circulant matrix to the augmented vector $(q_1, q_2, \ldots, q_m, 0, \ldots, 0)^t$. This matrix-vector product can be obtained efficiently by using three fast Fourier transforms of length $2m-1$, see for instance Chan and Ng [8]. Thus the cost of obtaining the diagonal sums is $O(m \log m)$ operations and the storage required is $O(m)$. □

Corollary 1 *Let P and Q be two given k-by-m matrices. Then the diagonal sums of the product P^tQ can be obtained in $O(km \log m)$ operations and the storage required is $O(m)$.*

Proof: We have

$$P^tQ = \sum_{\alpha=1}^{k} \mathbf{p}_\alpha^t \mathbf{q}_\alpha \tag{12}$$

where $\mathbf{p}_\alpha$ and $\mathbf{q}_\alpha$ are the αth row of P and Q respectively. Notice that the sum of the diagonal sums is equal to the diagonal sums of the sum. Therefore we can form the diagonal sums of each term $\mathbf{p}_\alpha^t \mathbf{q}_\alpha$ and sum them up to get the diagonal sums of P^tQ. By Lemma 2, the diagonal sums for each $\mathbf{p}_\alpha^t \mathbf{q}_\alpha$, $\alpha = 1, \ldots, k$, can be obtained in $O(m \log m)$ operations with $O(m)$ storage. Once the diagonal sums of $\mathbf{p}_\alpha^t \mathbf{q}_\alpha$ are formed, they can be accumulated to the final result and there is no need to store the intermediate diagonal sums for each $\alpha = 1, \ldots, k$. Thus the cost of obtaining the diagonal sums of P^tQ is $O(km \log m)$ operations and storage requirement is $O(m)$. □

Corollary 2 *Let P be a given k-by-m matrix and Λ be a k-by-k matrix. Then the diagonal sums of the product $P^t \Lambda P$ can be obtained in $O(km \log m) + O(k^2 m)$ operations and the storage required is $O(m)$.*

Proof: We just need to compute the product ΛP first and then apply Corollary 1 to the product $P^t(\Lambda P)$. In computing ΛP, we need one row of the product at any one time (see (12)) and therefore the total cost of forming ΛP is $k^2 m$ operations and the total storage required is $O(m)$. □

4 Construction of Optimal Circulant Preconditioner

In this section, we develop an $O(n \log n)$ method of constructing the optimal circulant preconditioner $c(A)$ for the approximation matrix A given in (9). Recall that $c(A)$ is defined as the minimizer of $\|C - A\|_F$ over all circulant matrices C. By (1), it is clear that $c(\cdot)$ is a linear operator. Therefore by (4), we have

$$c(A) = c(\sum_{\mu=0}^{l-2} A^{(\mu)}) = \sum_{\mu=0}^{l-2} c(A^{(\mu)}),$$

where we recall that $n = k \cdot 2^l$. By Lemma 1, $c(A^{(\mu)})$ can be obtained easily if we have the diagonal sums of $A^{(\mu)}$. In view of the block structure of $A^{(\mu)}$ (cf. (5)), we can have the diagonal sums of $A^{(\mu)}$ if we have the diagonal sums of its sub-blocks $A^{(\mu,\nu)}$. Thus in the following, we first consider the complexity of computing the diagonal sums of the sub-blocks $A^{(\mu,\nu)}$. Then the results will be pieced together to get the complexity counts for computing the diagonals sums of A.

We begin by noting that for $\mu = 1, \ldots, l-2$, $A^{(\mu)}$ is a block matrix made up of sub-blocks $A^{(\mu,\nu)}$ that concentrate only on four block-diagonals (cf (5)). Since the sum of diagonal sums is equal to the diagonal sums of the sum, one can obtain the diagonal sums of $A^{(\mu)}$ by summing the sub-blocks $A^{(\mu,\nu)}$ along the four block-diagonals first and computing the diagonal sums afterward. To be more specific, let us consider the example in (5) first. Here $\mu = 2$. The diagonal sums of $A^{(2)}$ can be obtained from the diagonal sums of the following four matrices:

$$\sum_{\nu=1}^{3} A^{(2,\nu)}, \quad \sum_{\nu=4}^{9} A^{(2,\nu)}, \quad \sum_{\nu=10}^{15} A^{(2,\nu)} \quad \text{and} \quad \sum_{\nu=16}^{18} A^{(2,\nu)}.$$

By (6), these four matrices can be rewritten as

$$\begin{aligned}
\sum_{\nu=1}^{3} A^{(2,\nu)} &= (P^{(2)})^t \{\sum_{\nu=1}^{3} \Lambda^{(2,\nu)}\} P^{(2)} \equiv (P^{(2)})^t \Lambda_1^{(2)} P^{(2)}, \\
\sum_{\nu=4}^{9} A^{(2,\nu)} &= (P^{(2)})^t \{\sum_{\nu=4}^{9} \Lambda^{(2,\nu)}\} P^{(2)} \equiv (P^{(2)})^t \Lambda_2^{(2)} P^{(2)}, \\
\sum_{\nu=10}^{15} A^{(2,\nu)} &= (P^{(2)})^t \{\sum_{\nu=10}^{15} \Lambda^{(2,\nu)}\} P^{(2)} \equiv (P^{(2)})^t \Lambda_3^{(2)} P^{(2)}, \\
\sum_{\nu=16}^{18} A^{(2,\nu)} &= (P^{(2)})^t \{\sum_{\nu=16}^{18} \Lambda^{(2,\nu)}\} P^{(2)} \equiv (P^{(2)})^t \Lambda_4^{(2)} P^{(2)}. \qquad (13)
\end{aligned}$$

From the diagonal sums of these four matrices, one can compute the diagonal sums of $A^{(2)}$, cf. (5). With this example in mind, it is easy to verify the following lemma.

Lemma 3 *For $\mu = 1, 2, \ldots, l-2$, the diagonal sums of $A^{(\mu)}$ can be obtained in $O(k^2 \mu 2^\mu) + O(k^2 2^{l-\mu}) + O(k^3 2^\mu)$ operations and $O(k 2^\mu)$ storage.*

Proof: As in the example above, we first have to sum the $\Lambda^{(\mu,\nu)}$ along the four block-diagonals to obtain $\Lambda_1^{(\mu)}$, $\Lambda_2^{(\mu)}$, $\Lambda_3^{(\mu)}$ and $\Lambda_4^{(\mu)}$ (cf (13) and (8)). By (6), there are $6(2^{l-1-\mu}-1)$ sub-blocks of $\Lambda^{(\mu,\nu)}$, which are all k-by-k matrices. Therefore to form $\Lambda_\alpha^{(\mu)}$, $\alpha = 1,2,3,4$, it requires at most $6(2^{l-1-\mu}-1)k^2$ operations and $4k^2$ memory.

Once $\Lambda_\alpha^{(\mu)}$ for $\alpha = 1,2,3,4$ are formed, we compute the diagonal sums of the matrices

$$(P^{(\mu)})^t \Lambda_\alpha^{(\mu)} P^{(\mu)}, \quad \alpha = 1,2,3,4,$$

(cf. (13)). We recall by (6) that $P^{(\mu)}$ are k-by-$2^\mu k$ matrices. Therefore, by Corollary 2, the diagonal sums of these four matrices can be obtained in $O(k^2 2^\mu(\mu + \log k) + k^3 2^\mu)$ operations and $O(k2^\mu)$ storage.

Once these diagonal sums are formed, we can accumulate them together to get the diagonal sums of $A^{(\mu)}$. This step requires no more than $2^{\mu+3}k$ operations since there are only four diagonal sums to accumulate and each of the diagonal sums has no more than $2^{\mu+1}k$ numbers. Combining all the complexity counts above, the lemma follows. □

Next we consider the case for $\mu = 0$.

Lemma 4 *The diagonal sums of $A^{(0)}$ can be obtained in $O(k^2 2^l)$ operations and $O(k)$ storage.*

Proof: For $\mu = 0$, the sub-blocks $A^{(0,\nu)}$ are of size k-by-k and are concentrated on 7 (instead of 4) block-diagonals next to and including the main block-diagonal, see for instance [1, Figure 4]. In other words, $A^{(0)}$ is a band matrix of band-width less than or equal to $8k$. Thus forming the diagonal sums of $A^{(0)}$ requires at most $O(kn) = O(k^2 2^l)$ operations and $8k$ memory. □

Combining Lemmas 10, 3 and 4, we have our main theorem.

Theorem 1 *For A given in (9), the cost of forming $c(A)$ for A is of $O(kn\log n) + O(k^2 n)$ operations and the storage required is $O(n)$.*

Proof: In view of Lemma 3 and 4, the cost of obtaining the diagonal sums of $\sum_{\mu=0}^{l-2} A^{(\mu)}$ is of the *order* of

$$k^2 2^l + k^2 \sum_{\mu=1}^{l-2} (\mu 2^\mu + 2^{l-\mu} + k2^\mu) \le k^2 l 2^l + k^3 2^l = kn\log n + k^2 n.$$

The memory requirement is of the *order* of

$$8k + k\sum_{\mu=1}^{l-2} 2^\mu = k2^{l-1} + 6k = \frac{n}{2} + 6k.$$

Once the diagonal sums of A are formed, by using Lemma 1, the first column of the circulant matrix $c(A)$ can be obtained in just another $O(n)$ operations. □

In the next section, we will apply our $c(A)$ to solving systems $A\mathbf{x} = \mathbf{b}$ arising from non-convolution type integral equations.

5 Boundary Integral Equations from Potential Equations

In this section, we consider solutions of potential equations

$$\begin{cases} \Delta w(x) = 0, & x \in \Omega, \\ w(x) = g(x), & x \in \partial\Omega, \end{cases}$$

where $\partial\Omega$ is a smooth close curve in $\mathbb{R}^2$ and Ω is either the bounded interior region with boundary $\partial\Omega$ or the unbounded exterior region with boundary $\partial\Omega$. In the boundary integral equation approach, the solution $w(x)$ is found by solving the density function $u(y)$ in the following Fredholm equation of the first kind:

$$-\frac{1}{2\pi}\int_{\partial\Omega} \log|x-y| u(y) dS_y = g(x), \qquad x \in \partial\Omega, \tag{14}$$

see Chen and Zhou [11, §6.12] or Chan, Sun and Ng [9].

If we define the boundary integral operator $\mathcal{B}$ as

$$(\mathcal{B}u)(x) \equiv -\frac{1}{2\pi}\int_{\partial\Omega} \log|x-y| u(y) dS_y,$$

then (14) can be written as

$$(\mathcal{B}u)(x) = g(x). \tag{15}$$

For simplicity, we parameterize the boundary $\partial\Omega$ by $(x_1(\theta), x_2(\theta))$, $0 \le \theta \le 2\pi$, and thus (15) can be expressed as

$$(\mathcal{B}u)(\theta) = \int_0^{2\pi} b(\theta,\phi) v(\phi) d\phi = g(\theta), \quad 0 \le \theta \le 2\pi, \tag{16}$$

where $v(\phi) = u(x_1(\phi), x_2(\phi))\sqrt{(x_1'(\phi))^2 + (x_2'(\phi))^2}$ and the kernel function $b(\theta,\phi)$ is given by

$$b(\theta,\phi) = -\frac{1}{4\pi}\log\left\{(x_1(\theta)-x_1(\phi))^2 + (x_2(\theta)-x_2(\phi))^2\right\}. \tag{17}$$

In order to guarantee that the operator $\mathcal{B}$ has a positive kernel, such that $\mathcal{B}$ have no eigensolutions of zero, we assume without loss of generality that

$$\mathrm{diam}(\partial\Omega) = \max_{x,y\in\partial\Omega} |x-y| < 1, \tag{18}$$

see [15, p.453] and [14, Remark2.5]. One can scale down the size of the given boundary if necessary, see Chan and Zhou [11, p.287].

The well-known advantage of the boundary integral equation approach is that the dimension of the problem is reduced by one. However, (14) is a first kind boundary integral equation having a weakly singular kernel. If the Galerkin method with piecewise polynomials basis functions is applied to discretize (16), the discrete matrix B of $\mathcal{B}$ will be ill-conditioned and has condition number increasing like $O(n)$, where n is the size of the matrix, see for instance Hsiao and Wendland [15, Remark 4]. Therefore if the system is solved by the conjugate gradient method, the number of iterations required for convergence will be increasing like $O(\sqrt{n})$.

To overcome the ill-conditioned nature of the operator $\mathcal{B}$, optimal circulant integral operators are proposed in Chan, Sun and Ng [9] to precondition (16). Circulant integral operators are

convolution operators with 2π-periodic kernels. The *optimal circulant integral operator* of a given operator $\mathcal{B}$ is defined to be the minimizer of $|||\mathcal{C}-\mathcal{B}|||$ over all circulant integral operators $\mathcal{C}$, where $|||\cdot|||$ is the Hilbert-Schmidt norm, see Gohberg, Hanke and Koltracht [12]. For $\mathcal{B}$ given in (16), the kernel function of its optimal circulant integral preconditioner $\mathcal{M}$ is given by

$$\frac{1}{2\pi}\int_0^{2\pi} b(\theta,\theta-\phi)d\theta, \qquad 0<\phi<2\pi,$$

see Chan, Sun and Ng [9]. Instead of solving (15), we solve the preconditioned equation

$$\mathcal{M}^{-1}\mathcal{B}u = \mathcal{M}^{-1}g. \tag{19}$$

It is proven in [9] that this preconditioned equation is well-conditioned.

Theorem 2 (Chan, Sun and Ng [9, Theorems 3,4]) *Let $\mathcal{B}$ be the integral operator as defined in (16) and (17) and $\mathcal{M}$ be the optimal circulant integral operator for $\mathcal{B}$. Then there exist positive constants $\gamma_2 \geq \gamma_1 > 0$ such that the spectrum of $\mathcal{M}^{-1}\mathcal{B}$ lies in $[\gamma_1,\gamma_2]$. Moreover, if the Galerkin method is used to discretize the operator $\mathcal{M}^{-1}\mathcal{B}$, then the condition number of the discretized system is of $O(1)$ independent of the size of the discretized system.*

Thus if the conjugate gradient method is used to solve the preconditioned system (19), the convergence rate of the method is expected to be linear, see Axelsson and Barker [2, p.26].

In the following, we denote by B the discretization matrix of $\mathcal{B}$ using the Galerkin method with the trapezoidal rule and A the approximation matrix to $\mathcal{B}$ using our fast dense matrix method described in §2. We note that the optimal circulant preconditioner $c(B)$ of B is equal to the discretization matrix of $\mathcal{M}$ using the rectangular quadrature rule, see Chan, Sun and Ng [9, Theorem 5]. We remark that the matrix A provides a good approximation to the discretization matrix B, see [7], [18] or Table 1 below. Since the operator norm of the operator $c(\cdot)$ in matrix 2-norm is equal to 1 (see Chan, Jin and Yeung [5, Theorem 3]), we have,

$$\|c(A)-c(B)\|_2 \leq \|A-B\|_2. \tag{20}$$

Therefore, $c(A)$ will be a good approximation to $c(B)$ and hence to $\mathcal{M}$.

We now illustrate the effectiveness of the optimal circulant preconditioners and our approximation scheme by using a problem tested in Chan, Sun and Ng [9]. We consider the solution of (16) on regions Ω with boundaries $\partial\Omega$ as depicted in Figure 2. The boundaries are defined in polar coordinates by

$$r = \cos 2\theta + f_\lambda(\theta), \qquad 0\leq\theta\leq 2\pi,$$

where $f_\lambda(\theta) = (\lambda^4 - \sin^2 2\theta)^{1/2}$ with $\lambda > 1$. Since $\delta \equiv \mathrm{diam}(\partial\Omega) > 1$, we scale the boundary so that the diameter of the new boundary satisfies $\rho = \mathrm{diam}(\partial\Omega) = 3/4 < 1$, see (18). For such scaled domains, the kernel function (17) becomes

$$\begin{aligned} b(\theta,\phi) &= -\frac{1}{2\pi}\log\frac{\rho}{\delta}|2\sin\frac{\theta-\phi}{2}| - \frac{1}{4\pi}\log\left\{4\sin^2(\theta+\phi)\cos^2(\frac{\theta-\phi}{2})\left(1+\frac{\cos 2\theta+\cos 2\phi}{f_\lambda(\theta)+f_\lambda(\phi)}\right)^2\right. \\ &\qquad \left. + (\cos 2\theta + f_\lambda(\theta))(\cos 2\phi + f_\lambda(\phi))\right\} \\ &\equiv b_1(\theta,\phi) + b_2(\theta,\phi). \end{aligned} \tag{21}$$

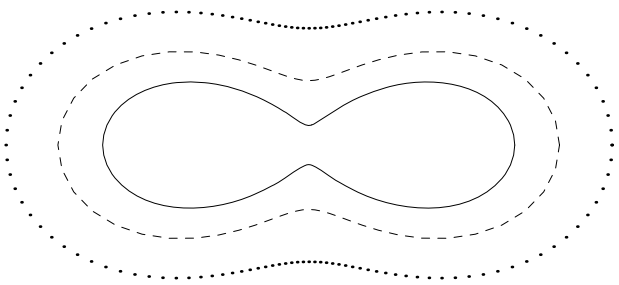


Figure 2. Solid line: $\lambda = 1.1$, dashed line: $\lambda = 1.3$, dotted line: $\lambda = 1.5$.

The right hand side $g(\theta)$ in (16) is chosen to be $g(\theta) = |\cos(\theta)|^{\frac{3}{2}}$, $0 \le \theta \le 2\pi$. All our computations were done in Matlab on an IBM 43P-133 workstation.

As in Chan, Sun and Ng [9], we discretize $[0, 2\pi]$ by uniform mesh and use the Galerkin method with piecewise constant polynomials as basis functions to discretize the equation. The integral over each element is computed by using trapezoidal rule with 3 points. Since b_1 in (21) is a 2π-periodic convolution kernel, we see that the discretization matrix B of $\mathcal{B}$ can be written as $B = C + B_2$ where C is a circulant matrix corresponding to the integration of b_1 over the elements. Thus C is determined only by its first column. From (21), we also see that

$$b_2(\theta, \phi) = b_2(\phi, \theta) = b_2(2\pi - \theta, 2\pi - \phi), \quad 0 \le \theta, \phi \le 2\pi. \tag{22}$$

Therefore, B_2 is a symmetric centro-symmetric matrix. In particular, if B_2 is an n-by-n matrix, it is determined by its upper half entries $[B_2]_{j,l}$, $1 \le j \le \lceil n/2 \rceil$, $1 \le l \le n$. The bottom half can be obtained by reflecting the upper half entries with respect to the center of the matrix.

It is clear that forming the matrix B_2 (or just its upper half) directly by integration of b_2 over the elements requires $O(n^2)$ operations. In Figure 3, we plot the log of the numbers of floating point operations in thousand (Kflops) required to form the upper half of B_2 against l for different values of k (solid lines). We recall that $n = k \cdot 2^l$. Thus the largest matrix size we tried is $n = 14 \cdot 2^8 = 3,584$. We remark that the counts do not depend on the values of λ and $\mathrm{diam}(\partial\Omega)$. We clearly see from the slope of the lines in the figure that the cost of constructing B_2 is increasing like $O(n^2)$.

Besides B_2, we also use our fast dense matrix method to approximate the integration of b_2 over the elements. This results in a matrix A_2 which can be obtained in $O(n)$ operations and requires only $O(n)$ storage, and that the matrix-vector product $A_2\mathbf{x}$ for any vector $\mathbf{x}$ can be done in $O(n \log n)$ operations, see Chan, Lin and Ng [7]. By the centro-symmetric property of b_2 (see (22)), we only need to generate the upper half of A_2. More precisely, we only need to apply our method to get the upper left and upper right $n/2$-by-$n/2$ submatrices of A_2 only. The bottom half of A_2 can be obtained by reflecting these two matrices.

In Figure 3, we plot the log of Kflops required to form the upper half of A_2 against l for different values of k (dashed lines). Since $n = k \cdot 2^l$, the largest matrix size we tried is $n = 14 \cdot 2^{12} = 57,344$. We remark that the counts do not depend on the values of λ and $\mathrm{diam}(\partial\Omega)$. We see from the slope of the lines that the cost of constructing A_2 is increasing like $O(n)$. In contrast, the cost for constructing B_2 is $O(n^2)$ (solid lines). We emphasize that there is no need to form B_2 in order to form A_2. We get A_2 by directly approximating b_2 in (21) using our fast dense matrix method.

To illustrate the accuracy of our approximation, the relative errors $\|A_2 - B_2\|_F / \|B_2\|_F$ for different λ are given in Table 1. Because generating B_2 is very expensive, we tried only matrices

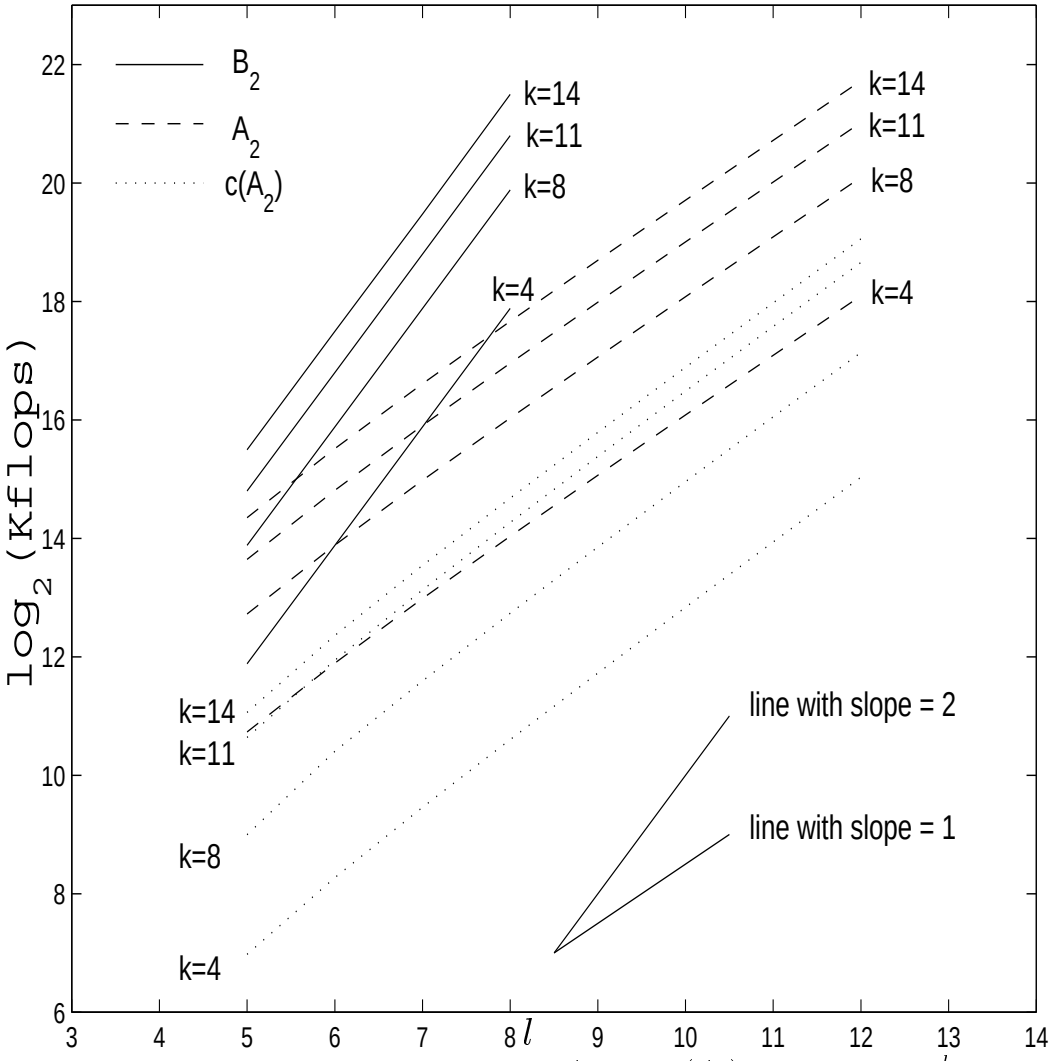


Figure 3. Cost in constructing B_2, A_2, and $c(A_2)$ where $n = k \cdot 2^l$.

k	l	$\lambda = 1.1$	$\lambda = 1.3$	$\lambda = 1.5$	k	l	$\lambda = 1.1$	$\lambda = 1.3$	$\lambda = 1.5$
4	5	3.89E-03	1.20E-03	4.37E-04	11	5	3.40E-06	9.44E-08	6.61E-09
4	6	4.58E-03	1.39E-03	5.03E-04	11	6	3.98E-06	1.09E-07	7.61E-09
4	7	4.94E-03	1.48E-03	5.38E-04	11	7	4.27E-06	1.16E-07	8.12E-09
4	8	5.13E-03	1.53E-03	5.57E-04	11	8	4.42E-06	1.19E-07	8.38E-09
8	5	5.16E-05	3.21E-06	4.39E-07	14	5	2.91E-07	3.23E-09	1.29E-10
8	6	6.00E-05	3.66E-06	5.03E-07	14	6	3.37E-07	3.70E-09	1.49E-10
8	7	6.43E-05	3.90E-06	5.36E-07	14	7	3.60E-07	3.94E-09	1.59E-10
8	8	6.65E-05	4.03E-06	5.54E-07	14	8	3.72E-07	4.06E-09	1.64E-10

Table 1: $\|B_2 - A_2\|_F / \|B_2\|_F$ for different kernels, where $n = k \cdot 2^l$.

of size up to $14 \cdot 2^8 = 3,584$. We see from the table that our approximation scheme provides a very accurate approximation A_2 to the matrix B_2 even for small k like 8.

To accelerate the convergence of the conjugate gradient method, we use the optimal circulant preconditioner $c(C + A_2)$ to precondition the system $(C + A_2)$. By the linear property of $c(\cdot)$, we see that

$$c(C + A_2) = c(C) + c(A_2) = C + c(A_2).$$

In constructing $c(A_2)$, we have also made use of the centro-symmetric property of the matrix A_2, i.e. we only need to compute the diagonal sums of the upper half of A_2. In Figure 3, we plot the log of Kflops required to form $c(A_2)$ against l for different values of k (dotted lines). Since $n = k \cdot 2^l$, the largest matrix size we tried here is also $n = 14 \cdot 2^{12} = 57,344$. We remark again that the counts do not depends on the values of λ and $\mathrm{diam}(\partial\Omega)$. We see from the figure that the cost of constructing $c(A_2)$ is increasing like $O(nkl) = O(n \log n)$. In contrast, the cost for constructing $c(B_2)$ using (1) will be of $O(n^2)$ operations.

Next we test the efficiency and accuracy of solving (19) using the approximation A_2 for B_2 and the optimal circulant preconditioner $C + c(A_2)$. Using the conjugate gradient method, we solve for the vector $\mathbf{x}$ in the non-preconditioned system (cf (16))

$$(C + B_2)\mathbf{x} = \mathbf{g}, \tag{23}$$

and for the vector $\mathbf{y}$ in the preconditioned system (cf (19))

$$c(C + A_2)^{-1}(C + A_2)\mathbf{y} = c(C + A_2)^{-1}\mathbf{g}. \tag{24}$$

We note that $c(C + A_2) = C + c(A_2)$ is a circulant matrix. Hence its inverse can be found efficiently in $O(n \log n)$ operations by using fast Fourier transforms, see Chan and Ng [8]. Thus the cost per iteration of solving (23) and (24) by conjugate gradient method is $O(n^2)$ and $O(n \log n)$ operations respectively.

For both systems (23) and (24), we choose the zero vector as the initial guess and the stopping criterion is $\|\mathbf{r}_q\|_2 / \|\mathbf{r}_0\|_2 < 10^{-10}$, where $\mathbf{r}_q$ is the residual vector at the qth iteration. The numbers of iterations required for convergence for different λ are given in Table 2, where the symbols C and I indicate if circulant preconditioning is used or not. From the table, we see that the numbers of iterations of the preconditioned systems are smaller than that of the non-preconditioned ones considerably. Notice that the iteration numbers of the preconditioned

$\rho = 3/4$		$\lambda = 1.1$			$\lambda = 1.3$			$\lambda = 1.5$		
k	l	I	C	e_n	I	C	e_n	I	C	e_n
4	5	29	9	5.90E-03	30	8	1.60E-03	31	7	1.52E-03
4	6	40	9	6.41E-03	41	7	1.75E-03	41	7	1.62E-03
4	7	54	9	6.83E-03	54	7	1.81E-03	53	7	1.65E-03
4	8	70	9	7.52E-03	71	7	1.86E-03	74	7	1.67E-03
8	5	40	9	1.91E-04	41	7	2.41E-05	41	7	4.33E-06
8	6	54	9	2.03E-04	54	7	2.59E-05	53	7	4.63E-06
8	7	70	9	2.06E-04	71	7	2.61E-05	74	7	4.70E-06
8	8	94	9	2.14E-04	94	7	2.61E-05	95	7	4.77E-06
11	5	51	9	3.76E-05	49	7	1.03E-06	49	7	9.09E-08
11	6	63	9	4.07E-05	63	7	1.14E-06	64	7	9.90E-08
11	7	86	9	4.14E-05	85	7	1.16E-06	85	7	1.01E-07
11	8	117	9	4.26E-05	118	7	1.18E-06	118	7	1.02E-07
14	5	53	9	5.83E-06	53	7	4.81E-08	54	7	5.57E-09
14	6	73	9	6.47E-06	74	7	5.36E-08	74	7	9.01E-09
14	7	93	9	6.51E-06	95	7	5.51E-08	91	7	2.19E-08
14	8	121	9	6.44E-06	123	7	6.11E-08	122	7	3.03E-08

Table 2: Numbers of Iterations and Relative Errors, where $n = k \cdot 2^l$.

systems are uniformly bounded whereas those of the original systems are increasing with n as expected.

Finally, we compare the accuracy of the solution $\mathbf{y}$ of the approximate system (24) with the solution $\mathbf{x}$ of (23). We give the relative errors $\|\mathbf{x} - \mathbf{y}\|_2 / \|\mathbf{x}\|_2$ for different λ in Table 2 under the column e_n. We see that the solution $\mathbf{y}$ provides a very accurate approximation to the solution $\mathbf{x}$ even for small k.

References

[1] B. Alpert, G. Beylkin, R. Coifman and V. Rokhlin, *Wavelets for the Fast Solution of Second-Kind Integral Equations*, SIAM J. Sci. Comput., 14 (1993), 159–184.

[2] O. Axelsson and V. Barker, *Finite Element Solution of Boundary Value Problems*, Academic Press, Orlando, 1984.

[3] G. Beylkin, R. Coifman and V. Rokhlin, *Fast Wavelet Transforms and Numerical Algorithms I*, Comm. Pure Appl. Math., 46 (1991), 141–183.

[4] R. Chan, X. Jin and M. Ng, *Circulant Integral Operators as Preconditioners for Wiener-Hopf Equations*, Integr. Equat. Oper. Theory, 21 (1995), 12–23.

[5] R. Chan, X. Jin and M. Yeung, *The Circulant Operator in the Banach Algebra of Matrices*, Lin. Algebra Appls., 149 (1991), 41–53.

[6] R. Chan and F. Lin, *Preconditioned Conjugate Gradient Methods for Integral Equations of the Second Kind Defined on the Half-Line,* J. Comput. Math., 14 (1996), 223–236.

[7] R. Chan, F. Lin and W. Ng, *Fast Dense Matrix Method for the Solution of Integral Equations of the Second Kind*, Numer. Math. J. Chinese Univ. (English Ser.), 7 (1998), 105–120.

[8] R. Chan and M. Ng, *Conjugate Gradient Methods for Toeplitz Systems*, SIAM Review, 38 (1996), 427–482.

[9] R. Chan, H. Sun and W. Ng, *Circulant Preconditioners for Ill-Conditioned Boundary Integral Equations from Potential Equations*, Int. J. Numer. Meth. Eng., 43 (1998), 1505–1521.

[10] T. Chan, *An Optimal Circulant Preconditioner for Toeplitz Systems*, SIAM J. Sci. Statist. Comput., 9 (1988), 766–771.

[11] G. Chen and J. Zhou, *Boundary Element Methods*, Academic Press, London, 1992.

[12] I. Gohberg, M. Hanke and I. Koltracht, *Fast Preconditioned Conjugate Gradient Algorithms for Wiener-Hopf Integral Equations*, SIAM J. Numer. Anal., 31 (1994), 429–443.

[13] L. Greengard and V. Rokhlin, *A Fast Algorithm for Particle Simulations*, J. Comput. Phys., 73 (1987), 325–348.

[14] G. Hsiao and R. C. MacCamy, *Solutions of Boundary Value Problems by Integral Equations of the First Kind*, SIAM Rev., 15 (1973), 687–705.

[15] G. Hsiao and W. L. Wendland, *A Finite Element Method for Some Integral Equations of the First Kind*, J. Math. Anal. Appla. 58 (1977), 447–481.

[16] L. Reichel, *Fast Solution Methods for Fredholm Integral Equations of the Second Kind*, Numer. Math., 57 (1989), 719–736.

[17] E. Tyrtyshnikov, *Optimal and Super-Optimal Circulant Preconditioners,* SIAM Matrix Anal. Appl., 13 (1992), 459–473.

[18] E. Tyrtyshnikov, *Mosaic Ranks and Skeletons*, Lect. Notes Comput. Sc., 1196 (1997), 505–526.