*East-West J. Numer. Math.*, Vol. 0, No. 0, pp. 1–24 (2002)
 *Prepared using* jnm.sty *[Version: 06.06.2001 v1.1]*

# A Fast Solver for Fredholm Equations of the Second Kind with Weakly Singular Kernels

RAYMOND H. CHAN[*], FU-RONG LIN[†], and CHI-FAI CHAN[‡]

**Abstract** — In this paper, we consider solutions of Fredholm integral equations of the second kind where the kernel functions are asymptotically smooth or products of such functions with highly oscillatory coefficient functions. We present a scheme based on polynomial interpolation to approximate matrices $A$ from the discretization of these integral operators. Our approximation matrix $B$ is obtained by partitioning the domain on which the kernel function is defined into subdomains of different sizes and approximating the kernel function at each subdomain by interpolation polynomial at the Chebyshev points. Although $B$ is dense, it can still be constructed in $O(nk)$ operations, requires $O(nk)$ storage and the product $B\mathbf{y}$ can be obtained in $O(nk\log n)$ operations, where $n$ is the size of the matrix and $k$ is the degree of the interpolation polynomial used. We prove that the Frobenius norm $\|A-B\|_F \leqslant \epsilon$ if $k$ is of $O(\log \epsilon^{-1})$ for smooth kernels (including $\log|x-t|$) and of $O(\log\log n+\log\epsilon^{-1})$ for weakly singular kernels such as $|x-t|^{-1/2}$. Comparison with the wavelet-like method by Alpert et. al. [2] shows that our method requires less memory and is more accurate.

**Keywords:** Fredholm integral equation, polynomial interpolation, conjugate gradients

## 1. INTRODUCTION

In this paper, we consider the fast solutions of Fredholm integral equation of the second kind:

$$f(x)-d(x)\int_0^1 a(x,t)f(t)dt=g(x),\quad x\in[0,1], \tag{1.1}$$

where the kernel function $a(x,t)$ is in $L^2[0,1]^2$ and is analytic except at $x = t$, and the unknown function $f(x)$ and the right hand side function $g(x)$ are in $L^2[0,1]$. We assume that the coefficient function $d(x)$ can be oscillatory but bounded. These equations lie between the equations with smooth kernels and those with arbitrary oscillatory kernels.

[*]Department of Mathematics, Chinese University of Hong Kong, Shatin, Hong Kong. Research supported in part by HKRGC grant CUHK4212/99P and CUHK DAG grant 2060183.

[†]Department of Mathematics, Shantou University, Shantou, Guangdong, People's Republic of China. Research supported by the natural science foundation of China No. 19901017 and the natural science foundation of Guangdong province of China No. 974007.

[‡]Department of Mathematics, Chinese University of Hong Kong, Shatin, Hong Kong.

It is well-known that an integral operator with weakly singular kernel is a compact operator ([11], Theorem 2.21) and that the product of a bounded operator and a compact operator is still compact ([14], Theorem 4.18). Therefore the integral $d(x)\int_0^1 a(x,t)f(t)dt$ in (1.1) defines a compact operator. Hence for integral equations of the second kind, they are either singular or well-conditioned. Using quadrature rules such as the trapezoidal rule, (1.1) becomes a linear system

$$(I - DA)\mathbf{f} = \mathbf{g}, \tag{1.2}$$

where $I$ is the identity matrix, $D$ is a diagonal matrix and $A$ is a dense matrix corresponding to the quadrature rule and quadrature points used in discretization of the integral in (1.1).

Various direct and iterative methods have been proposed for solving (1.2), see [7] for instance. However, one overriding drawback of these methods is the high cost of working with the associated dense matrices $A$. For problems discretized with $n$ quadrature points, direct methods such as Gaussian elimination method require $O(n^3)$ operations to obtain the numerical solutions. For iterative methods such as the conjugate gradient type methods, each iteration requires matrix-vector multiplications of the form $A\mathbf{y}$, see [8]. Therefore even for well-conditioned problems, such as the second kind integral equations, the methods require $O(n^2)$ operations, which for large-scale problems is often prohibitive.

In recent years, a number of fast algorithms for (1.2) have been developed, see for instance [9, 13, 3, 2]. The fast multipole method proposed in [9] combines the use of low-order polynomial interpolation of the kernel functions with a divide-and-conquer strategy. For kernel functions that are Coulombic or gravitational in nature, it results in an order $O(n)$ algorithm for the matrix-vector multiplications. In [13], the integral equation is discretized at the Chebyshev points and the resulting matrix is approximated by a low-rank modification of the identity matrix which can be obtained in $O(n\log n)$ operations. However, the solution of the discretized system still requires $O(n^2)$ operations to obtain if the solution is not smooth. In [3], an $O(n\log n)$ algorithm is developed by exploiting the connections between the use of wavelets and their applications on Calderon-Zygmund operators. In [2], wavelet-like bases are used to transform the dense discretization matrices into sparse matrices, which are then inverted by the Schulz method. The complexity of the resulting algorithm is bounded by $O(n\log^2 n)$.

In this paper, we present a fast matrix-vector multiplication scheme based on polynomial interpolation technique. We partition the discretization matrix $A$ into sub-blocks of different sizes as in [2]. Then, a degree $k$ interpolation polynomial is used on each of the sub-blocks to obtain an approximation matrix $B$. We show that the approximation matrix $B$ can be constructed in $O(nk)$ operations and only requires $O(nk)$ storage to represent. We also show that for any vector $\mathbf{y}$, the product $B\mathbf{y}$ can be obtained in only $O(nk\log n)$ operations. Therefore, for integral equations of the second kind, the approximated systems can be solved by the conjugate gradient type methods in only $O(nk\log n)$ operations.

We then discuss the accuracy of $B$ with respect to the degree $k$ of the interpolation polynomial used in the approximation. We will concentrate on kernel functions

that are asymptotically smooth, i.e.,

$$|\mathcal{D}^m a(x,t)| \leqslant \rho m^\alpha m! |x-t|^{\delta-m}, \quad \text{for } m > 0,$$

where $\alpha > 0$, $\rho > 0$ and $\mathcal{D}^m$ is the $m$th-order partial derivative of $x$ or $t$ or both (see [16]). (We note that for problems in higher dimensional spaces, one can use a more general definition as given in [15].) For such kernel functions, we prove that $\|A - B\|_F \leqslant \epsilon$ if the degree $k$ of interpolation polynomial satisfies

$$k \geqslant \begin{cases} O(\log \epsilon^{-1}), & 0 \leqslant \delta, \\ O(\log \epsilon^{-1}) + O(\log\log n), & -1 \leqslant \delta < 0, \\ O(\log \epsilon^{-1}) + O(\log n), & \delta < -1. \end{cases} \tag{1.3}$$

In particular, for smooth kernel functions ($\delta \geqslant 0$) that include the function $\log|x-t|$, $k$ is independent of the matrix size $n$. Hence our method requires only $O(n \log \epsilon^{-1})$ storage and $O(n \log n \log \epsilon^{-1})$ operations to solve the corresponding integral equations. For weakly singular kernel functions ($-1 \leqslant \delta < 0$) such as $|x-t|^{-1/2}$, the memory requirement is $O(n(\log\log n + \log \epsilon^{-1}))$ and the computational complexity is $O(n \log n(\log\log n + \log \epsilon^{-1}))$. We note that the methods in [1, 16] both require $O(n \log n \log \epsilon^{-1})$ memory for the same accuracy for $\log|x-t|$ and other weakly singular kernel functions.

The outline of this paper is as follows. In §2, we derive our fast multiplication scheme and analyze its complexity. We will show that our approximation matrix $B$ can be obtained in $O(nk)$ operations and requires $O(nk)$ storage. Moreover, the cost for matrix-vector multiplication $B\mathbf{y}$ is just $O(nk \log n)$. In §3, we analyze the accuracy of $B$ and give a proof of (1.3). Numerical results are given in §4 to illustrate the efficiency, accuracy and stability of our approximation scheme. We will also compare the storage requirement and accuracy of our method with that in [2]. The results show that our method is more accurate and requires less storage. Finally, concluding remarks are given in §5.

## 2. THE APPROXIMATION

In this section, we present our fast matrix-vector multiplication scheme for the discretization matrix $A$ of (1.1). The idea of the scheme is to take advantage of the smoothness of the kernel function $a(x,t)$ away from the singularity where we can use low degree polynomials to approximate the function accurately. As an example mentioned in [2], for any $c > 0$, the function $\log x$ can be approximated within $4^{-9}$ accuracy on $[c, 2c]$ by using polynomials of degree at most 7. In our approximation, we partition the domain $[0,1]^2$ on which the discretization matrix $A$ is defined into subdomains of different sizes and approximate the kernel function $a(x,t)$ on each subdomain by a low degree polynomial.

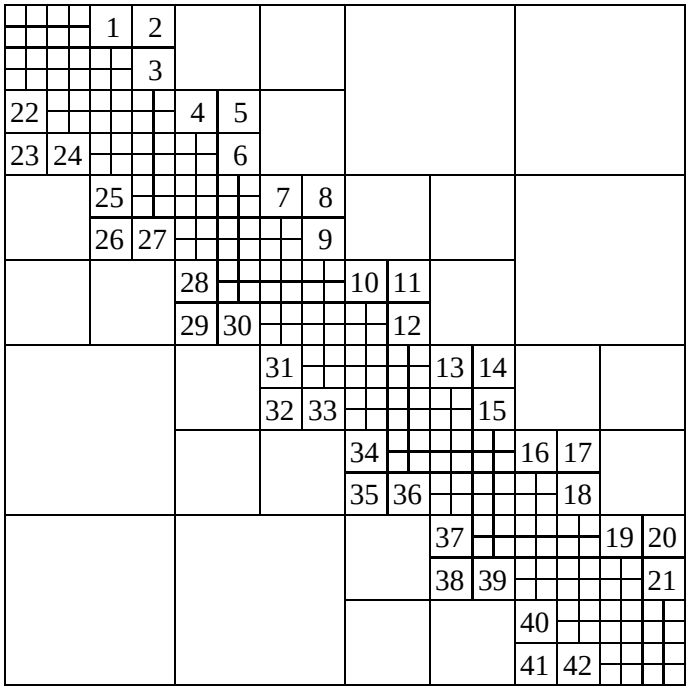


**Figure 1.** Partition of the Matrix $A$

### 2.1. The Partition

For simplicity, we let the size of $A$ be $n = 2^l k$. Here $k$ is a fixed integer that depends on the smoothness of $a(\cdot,\cdot)$ and the given accuracy $\epsilon$. Theorem 2 in §3 will give an estimate of $k$. We partition $A$ into blocks of different sizes as shown in Figure 1. Our partition is the same as that in [2]. The blocks near the diagonal are of size $k$-by-$k$, those next remote are of size $2k$-by-$2k$ and up to the largest size $2^{l-2}k$-by-$2^{l-2}k$.

By grouping the block matrices with the same size into one matrix, we see that $A$ can be written as the sum of a sequence of block matrices

$$A = A^{(0)} + A^{(1)} + \ldots + A^{(l-2)}. \tag{2.1}$$

Here $A^{(u)}$, $u = 0, \ldots, l-2$, contains only size $2^u k$-by-$2^u k$ block matrices. We can see from 1 that the number of nonzero blocks in $A^{(u)}$ is given by

$$v_u = \begin{cases} 6 \cdot 2^l - 8, & u = 0, \\ 6(2^{l-1-u} - 1), & u = 1, \ldots, l-2. \end{cases} \tag{2.2}$$

We will denote the nonzero blocks in $A^{(u)}$ by $A^{(u,v)}$ for $v = 1, \ldots, v_u$. The numbered blocks in Figure 1 give the non-zero blocks in $A^{(1)}$ and each numbered

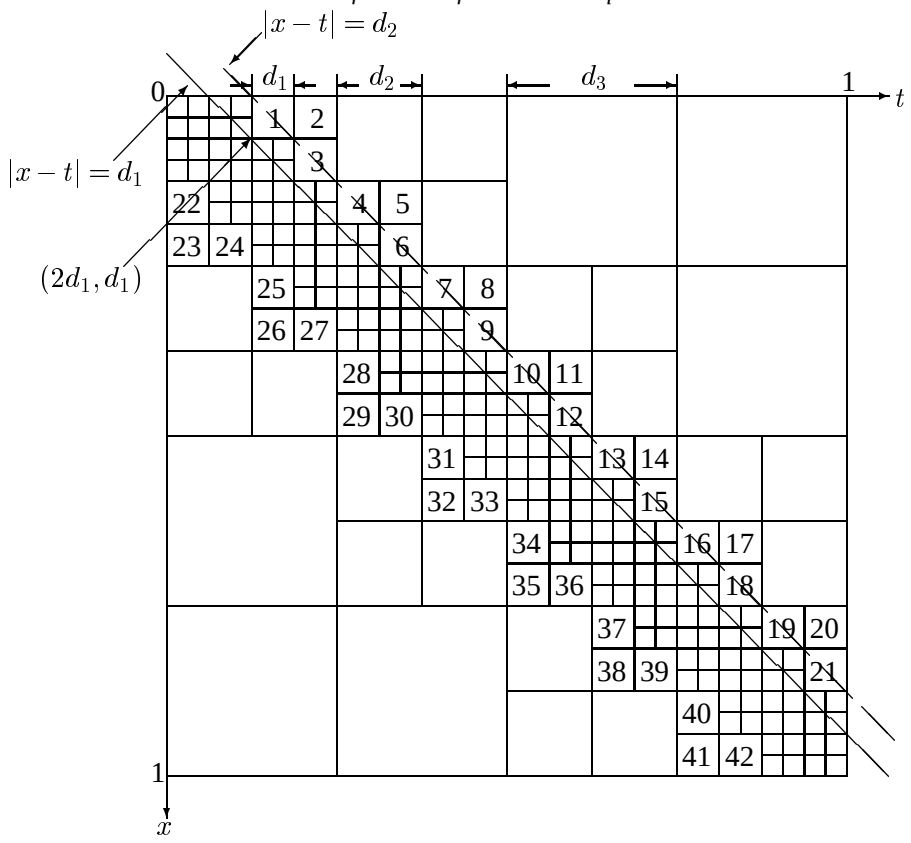


**Figure 2.** Partition of the Domain $[0,1]^2$

block $A^{(1,v)}$, $v=1,\cdots,42$, is a $2k$-by-$2k$ matrix. In a similar fashion, we partition the domain $[0,1]^2$ on which $a(\cdot,\cdot)$ (and hence $A$) is defined. More precisely, we let $\mathcal{S}^{(u,v)}$ be the sub-domain in $[0,1]^2$ on which the matrix $A^{(u,v)}$ is defined. We will call $\mathcal{S}^{(u,v)}$ the $(u,v)$-subdomain. As an illustration, the numbered subdomains in Figure 2 are $\mathcal{S}^{(1,v)}$ for $v=1,\ldots,42$.

Clearly, for a given $u$, $\mathcal{S}^{(u,v)}$ is of the same size for each $v$, and the side is of length $d_u=1/2^{l-u}$, $u=0,\cdots,l-2$. From the solid line in Figure 2, it is easy to see that all $(x,t)$ in $\mathcal{S}^{(1,1)}$ satisfy $|x-t|\geqslant|2d_1-d_1|=d_1$. In general, we have

$$|x-t|\geqslant d_u=\frac{1}{2^{l-u}},\qquad \forall(x,t)\in\mathcal{S}^{(u,v)}. \tag{2.3}$$

Equation (2.3) is required for the error analysis in §3.

### 2.2. The Sampling and the Interpolation

Our approximation matrix $B$ of $A$ is constructed by approximating each block $A^{(u,v)}$ in $A^{(u)}$ by a rank $k$ matrix $B^{(u,v)}$. The matrix $B^{(u,v)}$ is obtained by tak-

ing $k^2$ samples in $\mathcal{S}^{(u,v)}$ at the Chebyshev points and then interpolating $a(x,t)$ at these sample points. For notation simplicity, we will use the Nyström method with uniformly-spaced points to discretize (1.1) here. But we will see later that if other quadrature rules are used, our approximation scheme still works.

Thus let the entries of $A^{(u,v)}$ be defined as

$$[A^{(u,v)}]_{i,j} = \frac{1}{n-1} a(x_0^{(u,v)} + (i-1)h, t_0^{(u,v)} + (j-1)h), \quad 1 \leqslant i,j \leqslant 2^u k, \quad (2.4)$$

where $h = 1/(n-1)$. Here $A^{(u,v)}$ is defined on

$$\mathcal{S}^{(u,v)} = [x_0^{(u,v)}, x_0^{(u,v)} + (2^u k - 1)h] \times [t_0^{(u,v)}, t_0^{(u,v)} + (2^u k - 1)h].$$

The approximation matrix $B^{(u,v)}$ is obtained by interpolating the kernel function $a(\cdot,\cdot)$ on $\mathcal{S}^{(u,v)}$ at the Chebyshev points. We note that the idea to use interpolation at the Chebyshev points was also considered earlier in [12]. For a given domain $[\beta_1,\beta_2] \times [\gamma_1,\gamma_2]$, the Chebyshev points of degree $k$ are defined by

$$\begin{cases} x_r = \beta_1 + \frac{\beta_2 - \beta_1}{2}(1 + c_r), \\ t_s = \gamma_1 + \frac{\gamma_2 - \gamma_1}{2}(1 + c_s), \end{cases} \quad (2.5)$$

where

$$c_r = \cos\left(\frac{(2r-1)\pi}{2k}\right), \quad r = 1, \cdots, k$$

are the roots of the $k$th degree Chebyshev polynomial on the interval $[-1,1]$.

Let $(x_r^{(u,v)}, t_s^{(u,v)})$ $(r,s = 1,\cdots,k)$ be the Chebyshev points on $\mathcal{S}^{(u,v)}$. Using the Lagrange polynomials as basis functions, we get

$$a(x,t) \approx \sum_{r=1}^{k}\sum_{s=1}^{k} a(x_r^{(u,v)}, t_s^{(u,v)}) p_r(x, x_0^{(u,v)}) p_s(t, t_0^{(u,v)}), \qquad \forall (x,t) \in \mathcal{S}^{(u,v)}. \quad (2.6)$$

Here the Lagrange polynomial $p_r(x, x_0^{(u,v)})$ is defined as

$$p_r(x, x_0^{(u,v)}) = \prod_{\substack{s=1 \\ s \neq r}}^{k} \frac{(x - x_s^{(u,v)})}{(x_r^{(u,v)} - x_s^{(u,v)})} = \frac{\omega_r(-1 + \frac{2(x - x_0^{(u,v)})}{(2^u k - 1)h})}{\omega_r(c_r)}, \quad (2.7)$$

where

$$\omega_r(x) = \prod_{\substack{s=1 \\ s \neq r}}^{k} (x - c_s). \quad (2.8)$$

The Lagrange polynomial $p_s(t, t_0^{(u,v)})$ is defined similarly.

Combining (2.4) and (2.6), we get the approximation $B^{(u,v)}$ of $A^{(u,v)}$ as follows:

$$\begin{aligned} & [A^{(u,v)}]_{i,j} \\ = \quad & \frac{1}{n-1} a(x_0^{(u,v)} + (i-1)h, t_0^{(u,v)} + (j-1)h) \\ \approx \quad & \frac{1}{n-1} \sum_{r=1}^{k} \sum_{s=1}^{k} a(x_r^{(u,v)}, t_s^{(u,v)}) p_r(x_0^{(u,v)} + (i-1)h, x_0^{(u,v)}) \cdot \\ & \qquad \cdot p_s(t_0^{(u,v)} + (j-1)h, t_0^{(u,v)}) \\ \equiv \quad & [B^{(u,v)}]_{i,j} \end{aligned}$$

for $1 \leqslant i, j \leqslant 2^u k$. In matrix terms, $B^{(u,v)}$ is given by

$$B^{(u,v)} = L^{(u)} \Lambda^{(u,v)} (L^{(u)})^T, \tag{2.9}$$

where the $k$-by-$k$ matrix $\Lambda^{(u,v)}$ and the $2^u k$-by-$k$ matrix $L^{(u)}$ are defined respectively by

$$[\Lambda^{(u,v)}]_{r,s} = \frac{1}{n-1} a(x_r^{(u,v)}, t_s^{(u,v)}), \quad 1 \leqslant r, s \leqslant k \tag{2.10}$$

and

$$[L^{(u)}]_{i,j} = p_j(x_0^{(u,v)} + (i-1)h, x_0^{(u,v)}) = \frac{\omega_j(-1 + \frac{2(i-1)}{(2^u k - 1)h})}{\omega_j(c_j)}, \tag{2.11}$$

for $1 \leqslant i \leqslant 2^u k$ and $1 \leqslant j \leqslant k$. An important observation here is that the matrix $L^{(u)}$ does not depend on $v$.

We remark that to construct the approximation matrix $B^{(u,v)}$, we only need to evaluate the kernel function $a(\cdot, \cdot)$ at the $k^2$ sample points. There is no need to form the whole submatrix $A^{(u,v)}$. From (2.6), we see that our scheme still works if other compound $k$-point quadrature rule, such as the $k$-point Gaussian rule, is used to discretize (1.1). All we need is to change the entries of $L^{(u)}$ in (2.11) according to the quadrature points used.

### 2.3. The Approximation Matrix $B$

From (2.11), we know that the matrix $L^{(u)}$ is independent of the index $v$. By this property, we observe from Figure 1 and (2.9) that the approximation matrix $B^{(u)}$ of $A^{(u)}$ is of the form

$$B^{(u)} = [I_{2^{l-u}} \otimes L^{(u)}] \cdot \Lambda^{(u)} \cdot [I_{2^{l-u}} \otimes (L^{(u)})^T], \quad u = 1, \ldots, l-2,$$

where $I_{2^{l-u}}$ is the identity matrix of size $2^{l-u}$ and $\otimes$ is the Kronecker tensor product. Here $\Lambda^{(u)}$ has the same block structure as $A^{(u)}$ but each block $\Lambda^{(u,v)}$ in $\Lambda^{(u)}$ is of size $k$-by-$k$, see (2.10). In contrast, each block $A^{(u,v)}$ in $A^{(u)}$ is of size $2^u k$-by-$2^u k$ and that is where our saving in memory comes from. It follows from (2.1) that the approximation matrix $B$ of $A$ is

$$\begin{aligned} B &\equiv A^{(0)} + B^{(1)} + B^{(2)} + \ldots + B^{(l-2)} \\ &= A^{(0)} + \sum_{u=1}^{l-2} [I_{2^{l-u}} \otimes L^{(u)}] \cdot \Lambda^{(u)} \cdot [I_{2^{l-u}} \otimes (L^{(u)})^T]. \end{aligned} \tag{2.12}$$

In the next subsection, we study the storage requirement for $B$ and the cost of matrix-vector multiplications $B\mathbf{y}$ by using (2.12).

### 2.4. Complexity Analysis

In this subsection, we show that the approximation matrix $B$ can be constructed in $O(nk)$ operations and requires only $O(nk)$ storage. Moreover, the product $B\mathbf{y}$ can be done in $O(nk \log n)$ operations. Here $n = k2^l$ is the number of quadrature points used. In the following, we only count the number of multiplications as the number of additions is of the same order.

**Theorem 2.1.** *Let $B$ be defined as in (2.12), where $A^{(0)}$, $L^{(u)}$ and $\Lambda^{(u)}$ are defined in (2.1), (2.11) and (2.10) respectively. We have*

*(i) The storage requirement for representing $B$ is less than $9.5nk$.*

*(ii) $A^{(0)}$, $L^{(u)}$ and $\Lambda^{(u)}$ can be constructed in $2nk$ multiplications and $9nk$ function evaluations.*

*(iii) For any vector $\mathbf{y}$, the product $B\mathbf{y}$ can be obtained in $(2\log_2 n + 5)nk$ operations.*

**Proof.** *(i)* By using (2.12) to represent $B$, we only need to store $A^{(0)}$, $L^{(u)}$ and $\Lambda^{(u)}$ for $u = 1, \ldots, l-2$. Their storage requirements are summarized in Table 1.

Thus the total storage requirement is

$$(6 \cdot 2^l - 8)k^2 + \sum_{u=1}^{l-2} \left\{ 6(2^{l-1-u} - 1)k^2 + 2^u k^2 \right\} < 9.5 \cdot 2^l k^2 = 9.5nk.$$

*(ii)* From (2.2), we see that the matrix $A$ is partitioned into

$$(6 \cdot 2^l - 8) + \sum_{u=1}^{l-2} 6(2^{l-1-u} - 1) = 9 \cdot 2^l - 6l - 8$$

**Table 1.**
Storage Requirement

| Matrix | Storage | Explanation |
|---|---|---|
| $A^{(0)}$ | $(6\cdot 2^l-8)k^2$ | $A^{(0)}$ consists of $(6\cdot 2^l-8)$ blocks and each of them is of size $k$-by-$k$, see (2.2). |
| $\Lambda^{(u)}$ | $6(2^{l-1-u}-1)k^2$ | $\Lambda^{(u)}$ is a block matrix with $6(2^{l-1-u}-1)$ nonzero blocks and each nonzero block $\Lambda^{(u,v)}$ is of size $k$-by-$k$, see (2.2) and (2.10). |
| $L^{(u)}$ | $2^u k^2$ | $L^{(u)}$ is a $2^u k$-by-$k$ matrix, see (2.11). |

blocks. For each block, we require $k^2$ samples of $a(\cdot,\cdot)$, see (2.10). Hence we need

$$(9\cdot 2^l-6l-8)k^2<9nk$$

function evaluations for constructing $A^{(0)}$ and $\Lambda^{(u)}$ for $u=1,\cdots,l-2$.

Now let us discuss the cost of constructing $L^{(u)}$. It can be done by the following steps:

(a) Calculate the denominator $\{\omega_j(c_j)\}_{j=1}^k$ in (2.11). By (2.8), totally $k(k-2)$ operations are required.

(b) Construct the numerators $\omega_j(x_i)$ of $L^{(u)}$ in (2.11) column by column. Here $x_i=-1+2(i-1)/(2^uk-1)$. It is clear that $\omega_1(x_i)$ can be computed in $2^uk(k-2)$ operations. Since $\omega_{j+1}(x_i)=\omega_j(x_i)(x_i-c_j)/(x_i-c_{j+1})$, each column other than the first requires $2\cdot 2^uk$ operations. Therefore, we need $2^{u+1}k(k-1)$ operations to form the remaining $k-1$ columns.

(c) Finally, we can obtain the $(i,j)$th-entry of $L^{(u)}$ by dividing the numerators with the denominators, see (b) and (c). The cost is $2^uk^2$ operations.

Therefore, the construction of $L^{(u)}$, $u=1,\ldots,l-2$, requires

$$k(k-2)+\sum_{u=1}^{l-2}\{2^uk(k-2)+2^{u+1}k(k-1)+2^uk^2\}<2\cdot 2^lk^2=2nk$$

multiplications.
*(iii)* By (2.12), we have

$$B\mathbf{y}=A^{(0)}\mathbf{y}+\sum_{u=1}^{l-2}\left[I_{2^{l-u}}\otimes L^{(u)}\right]\cdot\Lambda^{(u)}\cdot\left[I_{2^{l-u}}\otimes (L^{(u)})^T\right]\mathbf{y}.$$

**Table 2.**
Examples of Asymptotically Smooth Functions

| $a(x,t)$ | Inequality | $\rho$ | $\alpha$ | $\delta$ |
|---|---|---|---|---|
| $\log\|x-t\|$ | $\|\mathcal{D}^m a(x,t)\| \leqslant m!\|x-t\|^{-m}$ | 1 | 0 | 0 |
| $\|x-t\|^{1/2}$ | $\|\mathcal{D}^m a(x,t)\| \leqslant 0.5m!\|x-t\|^{0.5-m}$ | 0.5 | 0 | 0.5 |
| $\|x-t\|^{-1/2}$ | $\|\mathcal{D}^m a(x,t)\| \leqslant 0.5m!\|x-t\|^{-0.5-m}$ | 0.5 | 0 | $-0.5$ |

By using the tensor structure, $\left[I_{2^{l-u}} \otimes (L^{(u)})^T\right]\mathbf{y}$ can be obtained in $2^l k^2$ operations. Since there are $6(2^{l-1-u}-1)$ sub-blocks of size $k$-by-$k$ in $\Lambda^{(u)}$ (see (2.2)), it can be easily checked that the total number of multiplications required to form $B\mathbf{y}$ is

$$(6\cdot 2^l - 8)k^2 + \sum_{u=1}^{l-2}\left\{6(2^{l-1-u}-1)k^2 + 2\cdot 2^l k^2\right\} < (2l+5)2^l k^2 = (2\log_2 n + 5)nk.$$

□

Numerical experiments in §4 show that our approximation matrices $B$ are accurate and stable and our multiplication scheme indeed attains the said complexity. Recall that Fredholm integral equations of the second kind are in general well-conditioned and hence can be solved by conjugate gradient type methods without preconditioning. The main cost in each iteration is the matrix-vector multiplication, see [8]. Thus using our scheme, the approximate equations can be solved in $O(nk\log n)$ complexity. In the next section, we will give the error analysis of our approximation scheme. In particular, we will give an estimate of $k$, the degree of the interpolation polynomial that we should use.

## 3. ERROR ANALYSIS

In this section, we present the error analysis of our approximation scheme. In the following, we use $\mathcal{D}^m$ to denote the $m$th-order partial derivative of variable $x$ or $t$ or both. We also use the symbols $\mathcal{D}_x^m$ and $\mathcal{D}_t^m$ for partial derivatives on the single variable $x$ and $t$ respectively. As in [16], we consider kernel functions $a(x,t)$ that are asymptotically smooth, i.e.,

$$|\mathcal{D}^m a(x,t)| \leqslant \rho m^\alpha m!|x-t|^{\delta-m}, \tag{3.1}$$

where $\alpha > 0,\ \rho > 0$ and $\delta$ are constants. This assumption is quite general and here are some typical examples:

Before we start, we recall that if $q(x)$ is an interpolation polynomial of a function $f(x) \in C^k[\beta_1,\beta_2]$ at the $k$ Chebyshev points of degree $k$ (see (2.5)), then

$$\sup_{x\in[\beta_1,\beta_2]} |f(x)-q(x)| \leqslant \frac{1}{2^{k-1}k!} \sup_{\beta_1\leqslant x\leqslant\beta_2} |f^{(k)}(x)| \left(\frac{\beta_2-\beta_1}{2}\right)^k, \tag{3.2}$$

see [10] (pp.284–287) for instance. For 2-dimensional interpolation polynomials at the Chebyshev points, we have the following result.

**Lemma 3.1.** *Let the function $a(x,t)$ be a function such that $\mathcal{D}_x^k\mathcal{D}_t^k a(x,t)$ is continuous in the domain $[\beta_1,\beta_2]\times[\gamma_1,\gamma_2]$. Let $q(x,t)$ be the interpolation polynomial at the $k^2$ Chebyshev points in $[\beta_1,\beta_2]\times[\gamma_1,\gamma_2]$ defined by (2.5). Then*

$$\begin{aligned}\|a-q\| \;\leqslant\; & \frac{1}{2^{k-1}k!}\left(\|\mathcal{D}_t^k a\|\left(\frac{\gamma_2-\gamma_1}{2}\right)^k+\|\mathcal{D}_x^k a\|\left(\frac{\beta_2-\beta_1}{2}\right)^k\right)\\ &+\frac{\|\mathcal{D}_t^k\mathcal{D}_x^k a\|}{(2^{k-1}k!)^2}\left(\frac{\beta_2-\beta_1}{2}\right)^k\left(\frac{\gamma_2-\gamma_1}{2}\right)^k,\end{aligned}$$

*where $\|\cdot\|$ is the supremum norm in $[\beta_1,\beta_2]\times[\gamma_1,\gamma_2]$.*

**Proof.** First of all, we define the 1-variable interpolation polynomial for a 2-dimensional function $a(x,t)$ as

$$\mathcal{I}_x a(x,t)=\sum_{j=1}^{k} a(x,t_j)p_j(t,\gamma_1) \tag{3.3}$$

and

$$\mathcal{I}_t a(x,t)=\sum_{i=1}^{k} a(x_i,t)p_i(x,\beta_1),$$

where $p_j(t,\gamma_1)$ is the Lagrange polynomial defined by the Chebyshev points $t_1,t_2,\ldots,t_k$ in $[\gamma_1,\gamma_2]$ and $p_i(x,\beta_1)$ is the Lagrange polynomial defined by the Chebyshev points $x_1,x_2,\ldots,x_k$ in $[\beta_1,\beta_2]$, see (2.7). Obviously, the interpolation polynomial $q(x,t)$ of $a(x,t)$ on the mesh $(x_i,t_j)$ $(i,j=1,\cdots,k)$ is given by

$$q(x,t)=\mathcal{I}_{xt}a(x,t)=\sum_{i=1}^{k}\sum_{j=1}^{k} a(x_i,t_j)p_i(x,\beta_1)p_j(t,\gamma_1).$$

It is not difficult to see that $\mathcal{I}_{xt}a=\mathcal{I}_t(\mathcal{I}_x a)=\mathcal{I}_x(\mathcal{I}_t a)$.

By the triangular inequality $\|a-q\|\leqslant\|a-\mathcal{I}_x a\|+\|\mathcal{I}_x a-\mathcal{I}_{xt}a\|$, we only have to estimate $\|a-\mathcal{I}_x a\|$ and $\|\mathcal{I}_x a-\mathcal{I}_{xt}a\|$. By applying (3.2) to (3.3), we get

$$\|a-\mathcal{I}_x a\|\leqslant\frac{\|\mathcal{D}_t^k a\|}{2^{k-1}k!}\left(\frac{\gamma_2-\gamma_1}{2}\right)^k. \tag{3.4}$$

Applying (3.2) to $\mathcal{I}_x a$, we also have

$$\|\mathcal{I}_x a-\mathcal{I}_{xt}a\|=\|\mathcal{I}_x a-\mathcal{I}_t(\mathcal{I}_x a)\|\leqslant\frac{\|\mathcal{D}_x^k(\mathcal{I}_x a)\|}{2^{k-1}k!}\left(\frac{\beta_2-\beta_1}{2}\right)^k. \tag{3.5}$$

From (3.3) we see that $\mathcal{D}_x^k \mathcal{I}_x a = \mathcal{I}_x \mathcal{D}_x^k a$ and hence by (3.4) we get

$$\|\mathcal{D}_x^k a - \mathcal{D}_x^k \mathcal{I}_x a\| = \|\mathcal{D}_x^k a - \mathcal{I}_x \mathcal{D}_x^k a\| \leqslant \frac{\|\mathcal{D}_t^k \mathcal{D}_x^k a\|}{2^{k-1} k!} \left(\frac{\gamma_2 - \gamma_1}{2}\right)^k.$$

Therefore

$$\|\mathcal{D}_x^k \mathcal{I}_x a\| \leqslant \|\mathcal{D}_x^k a\| + \frac{\|\mathcal{D}_t^k \mathcal{D}_x^k a\|}{2^{k-1} k!} \left(\frac{\gamma_2 - \gamma_1}{2}\right)^k. \tag{3.6}$$

Combining (3.4), (3.5) and (3.6), we finally get

$$\begin{aligned} \|a - q\| &\leqslant \|a - \mathcal{I}_x a\| + \|\mathcal{I}_x a - \mathcal{I}_{xt} a\| \\ &\leqslant \frac{\|\mathcal{D}_t^k a\|}{2^{k-1} k!} \left(\frac{\gamma_2 - \gamma_1}{2}\right)^k \\ &\quad + \frac{1}{2^{k-1} k!} \left(\frac{\beta_2 - \beta_1}{2}\right)^k \left\{ \|\mathcal{D}_x^k a\| + \frac{\|\mathcal{D}_t^k \mathcal{D}_x^k a\|}{2^{k-1} k!} \left(\frac{\gamma_2 - \gamma_1}{2}\right)^k \right\}. \end{aligned}$$

□

We now return our discussion to the accuracy of the approximation matrix $B$. Let the kernel function $a(x,t)$ satisfy assumption (3.1). We note that each subdomain $\mathcal{S}^{(u,v)}$ on which $A^{(u,v)}$ is defined is a square of length $d_u$ (see (2.3)) and all points $(x,t)$ in $\mathcal{S}^{(u,v)}$ satisfy $|x-t| \geqslant d_u$. Therefore, by (3.1) and Lemma 1, we get

$$\sup_{\mathcal{S}^{(u,v)}} |a - q^{(u,v)}| \leqslant \frac{2\rho k^\alpha k! (d_u)^{(\delta-k)}}{2^{k-1} k!} \left(\frac{d_u}{2}\right)^k + \frac{\rho (2k)^\alpha (2k)! (d_u)^{(\delta-2k)}}{(2^{k-1} k!)^2} \left(\frac{d_u}{2}\right)^{2k},$$

where $q^{(u,v)}$ is the interpolation polynomial of $a(x,t)$ at the Chebyshev points on the $\mathcal{S}^{(u,v)}$ and is given by the right hand side of (2.6). Noting that $(2k)!/(k! 2^{k-1})^2 \leqslant 2$ (which can easily be proved by mathematical induction), we have

$$\sup_{\mathcal{S}^{(u,v)}} |a - q^{(u,v)}| \leqslant (d_u)^\delta \left(\frac{2\rho k^\alpha}{2^{2k-1}} + \frac{2\rho (2k)^\alpha}{2^{2k}}\right) \leqslant (d_u)^\delta \frac{6\rho (2k)^\alpha}{2^{2k}} = (d_u)^\delta e_k, \tag{3.7}$$

where $e_k \equiv 6\rho (2k)^\alpha / 2^{2k}$. With the estimate (3.7), we have the following main result.

**Theorem 3.1.** *Let the kernel function $a(x,t)$ satisfy (3.1). Then for any $\epsilon > 0$, $\|A - B\|_F \leqslant \epsilon$ if the degree $k$ of the interpolation polynomial satisfies*

$$k \geqslant \begin{cases} c_1 + \log_2(\epsilon^{-1}), & 0 \leqslant \delta, \\ c_2 + \log_2(\epsilon^{-1}) + \frac{1}{2}\log_2 \log_2 n, & -1 \leqslant \delta < 0, \\ c_3 + \log_2(\epsilon^{-1}) + \frac{1}{2}\log_2 \log_2 n + \frac{1}{2}(-\delta - 1)\log_2 n, & \delta < -1. \end{cases} \tag{3.8}$$

*Here $c_i$ are constants depend on $\rho$ and $\alpha$ only.*

**Proof.** We have

$$\|A-B\|_F^2=\sum_{u=1}^{l-2}\|A^{(u)}-B^{(u)}\|_F^2=\sum_{u=1}^{l-2}\sum_{v=1}^{v_u}\|A^{(u,v)}-B^{(u,v)}\|_F^2.$$

Since $A^{(u,v)}$ and $B^{(u,v)}$ are $2^uk$-by-$2^uk$ matrices and $[A^{(u,v)}]_{i,j}=\frac{1}{n-1}a(\cdot,\cdot)$ (see (2.4)), by (2.2) and (3.7), we get

$$\|A-B\|_F^2\leqslant\frac{1}{(n-1)^2}\sum_{u=1}^{l-2}6(2^{l-1-u}-1)(2^uk)^2(d_u)^\delta e_k^2,$$

Recall from (2.3) that $d_u=2^{u-l}$, we have

$$\|A-B\|_F^2\leqslant\frac{6k^2e_k^2}{(n-1)^2}\sum_{u=1}^{l-2}(2^{l-1-u}-1)2^{2u}2^{\delta(u-l)}.$$

We consider the following three cases:

(i) $\delta\geqslant 0$: Note that $2^{\delta(u-l)}\leqslant 1$, we have

$$\begin{aligned}\|A-B\|_F^2 &\leqslant \frac{6k^2e_k^2}{(n-1)^2}\sum_{u=1}^{l-2}(2^{l-1-u}-1)2^{2u}\\ &= \frac{n^2-6nk+8k^2}{(n-1)^2}e_k^2\\ &= \frac{(n-3k)^2-k^2}{(n-1)^2}e_k^2<e_k^2.\end{aligned}\tag{3.9}$$

In order that $e_k\equiv 6\rho(2k)^\alpha/2^{2k}\leqslant\epsilon$, we can first choose a constant $c_0=c_0(\alpha)$ such that for all $k\geqslant c_0$, $k^\alpha/2^k\leqslant 1$. Clearly $k\geqslant\log_2(6\rho2^\alpha)-\log_2\epsilon$ is equivalent to $6\rho2^\alpha/2^k\leqslant\epsilon$. Thus (3.8) follows with

$$c_1=\max(c_0,\log_2(6\rho2^\alpha)).\tag{3.10}$$

(ii) $-1\leqslant\delta<0$: Note that $2^{\delta(u-l)}\leqslant 2^{-(u-l)}$, we have

$$\begin{aligned}\|A-B\|_F^2 &\leqslant \frac{6k^2e_k^2}{(n-1)^2}\sum_{u=1}^{l-2}(2^{l-1-u}-1)2^{2u}2^{-(u-l)}\\ &\leqslant \frac{6k^2e_k^2}{(n-1)^2}\sum_{u=1}^{l-2}(2^{2l-1})=\frac{3k^2e_k^2}{(n-1)^2}(l-2)2^{2l}\\ &= 3(l-2)(\frac{n}{n-1})^2e_k^2\leqslant 3le_k^2.\end{aligned}\tag{3.11}$$

By part (i), in order that $\sqrt{3l}e_k \leqslant \epsilon$, we need

$$k \geqslant c_1 - \log_2(\epsilon/\sqrt{3l}) = c_1 + \frac{1}{2}\log_2 3 + \frac{1}{2}\log_2\log_2 n - \log_2 \epsilon.$$

Hence (3.8) follows with $c_2 = c_1 + \frac{1}{2}\log_2 3$.

(iii) $\delta \leqslant -1$: Note that $2^{(\delta+1)u} \leqslant 1$, we have

$$\begin{aligned}
\|A-B\|_F^2 &\leqslant \frac{6k^2e_k^2}{(n-1)^2}\sum_{u=1}^{l-2}(2^{l-1-u}-1)2^{2u}2^{\delta(u-l)} \\
&\leqslant \frac{3k^2e_k^2}{(n-1)^2}\sum_{u=1}^{l-2}2^{(1-\delta)l}2^{(1+\delta)u} \\
&\leqslant 3(l-2)(\frac{n}{n-1})^2 2^{(-1-\delta)l}e_k^2 \leqslant 3l2^{(-1-\delta)l}e_k^2. \qquad (3.12)
\end{aligned}$$

Similar to part (ii), in order that $\sqrt{3l}2^{(-1-\delta)l/2}e_k \leqslant \epsilon$, we need

$$\begin{aligned}
k &\geqslant c_1 - \log_2(\epsilon/(\sqrt{3l}2^{(-1-\delta)l/2})) \\
&= c_1 + \frac{1}{2}\log_2 3 + \frac{1}{2}\log_2\log_2 n + \frac{-1-\delta}{2}\log_2 n - \log_2\epsilon.
\end{aligned}$$

Hence (3.8) follows with $c_3 = c_2 = c_1 + \frac{1}{2}\log_2 3$. □

We note that the constants $c_i$ in Theorem 2 are often small numbers. For example, for $\log|x-t|$, we have $\rho = 1$, $\alpha = 0$ and $\delta = 0$. It follows from (3.10) that $c_1 = \log_2 6 \approx 2.6$.

By Theorem 2, for a given accuracy $\epsilon$, we can choose one $k$ that satisfies (3.8) for all matrix sizes if the kernel has $\delta \geqslant 0$. For weakly singular kernels ($-1 \leqslant \delta < 0$), then $k$ will increase like $\frac{1}{2}\log_2\log_2 n$ as the matrix size $n$ increases. If the singularity is such that $\delta \leqslant -1$, then $k$ increases like $|\delta|\log_2 n$. Moreover, if $\alpha = 0$, we can see from (3.9), (3.11) and (3.12) that for a fixed $l$, we have

$$\|A-B\|_F = O(e_k) = O(2^{-2k}). \qquad (3.13)$$

These facts will be illustrated by numerical examples in §4.

Summarizing the results in Theorems 1 and 2, we have the Table 3.

These counts are better than those in [1, 16] which are of order $O(n\log(\epsilon^{-1})\log n)$ for weakly singular kernels.

## 4. NUMERICAL EXAMPLES

In this section, we consider numerical solutions of Fredholm integral equations of the second kind. Since the second kind integral equations are well-conditioned in

**Table 3.**
Complexity in Achieving $\|A-B\| \leqslant \epsilon$

| | Case 1 : $\delta \geqslant 0$ | Case 2 : $-1 \leqslant \delta < 0$ | Case 3 : $\delta < -1$ |
|---|---|---|---|
| Memory requirement | $9.5n\cdot \left\{c_1+\log_2(\epsilon^{-1})\right\}$ | $9.5n\cdot \left\{c_2+\log_2(\epsilon^{-1}) +\frac{1}{2}\log_2\log_2 n\right\}$ | $9.5n\cdot \left\{c_3+\log_2(\epsilon^{-1}) +\frac{1}{2}\log_2\log_2 n +\frac{1}{2}(-\delta-1)\log_2 n\right\}$ |
| Construction cost | $11n\cdot \left\{c_1+\log_2(\epsilon^{-1})\right\}$ | $11n\cdot \left\{c_2+\log_2(\epsilon^{-1}) +\frac{1}{2}\log_2\log_2 n\right\}$ | $11n\cdot \left\{c_3+\log_2(\epsilon^{-1}) +\frac{1}{2}\log_2\log_2 n +\frac{1}{2}(-\delta-1)\log_2 n\right\}$ |
| Cost of $B\cdot\mathbf{y}$ | $(2\log_2 n+5)n\cdot \left\{c_1+\log_2(\epsilon^{-1})\right\}$ | $(2\log_2 n+5)n\cdot \left\{c_2+\log_2(\epsilon^{-1}) +\frac{1}{2}\log_2\log_2 n\right\}$ | $(2\log_2 n+5)n\cdot \left\{c_3+\log_2(\epsilon^{-1}) +\frac{1}{2}\log_2\log_2 n +\frac{1}{2}(-\delta-1)\log_2 n\right\}$ |

general, we can solve them by using conjugate gradient type methods, see [8]. The main computational task in each iteration is the matrix-vector multiplication of the form $A\mathbf{y}$. As the discretization matrix $A$ is dense, the cost is $O(n^2)$. In the following, we overcome this difficulty by replacing $A$ with our approximation matrix $B$ given in (2.12). We will examine the accuracy of $B$ and its storage requirement. Moreover, we will compare our method with the wavelet-like method in [2].

As in [2], we discretize (1.1) by the following formulae

$$[A]_{i,j} = \begin{cases} \frac{1}{n-1}a(\frac{i-1}{n-1},\frac{j-1}{n-1}) & i \neq j \\ 0 & i = j \end{cases}$$

and

$$[D]_{i,i} = d\left(\frac{i-1}{n-1}\right), \quad i = 1,\ldots,n$$

to obtain the matrix equation (1.2). The formulae correspond to a primitive, trapezoidal-like quadrature discretization of the integral equation. We test our algorithm for the following six kernel functions:

(i) $\log|x-t|$,

(ii) $\cos(xt^2)\log|x-t|$,

(iii) $\cos(xt^2)|x-t|^{-1/2}$,

(iv) $\cos(xt^2)|x-t|^{1/2}$,

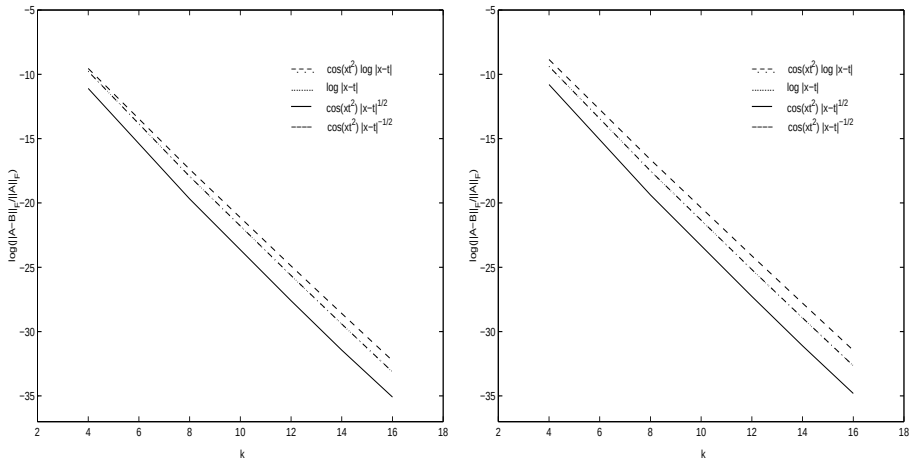


**Figure 3.** $k$ against $\log(\|A-B\|_F/\|A\|_F)$ for $l=4$ (left) and $l=10$ (right)

(v) $(1+\frac{1}{2}\sin(100x))\log|x-t|$, and

(vi) $\sin(100x)\log|x-t|$.

The kernel functions (i) to (v) are examples tested in [2]. We note that the wavelet-like method in [2] is not applicable to (vi) as $\sin(100x)$ is not a positive function. For (i) to (v), the coefficient function $d(x)\equiv 1$ and hence the matrix $D$ in (1.2) is the identity matrix. For (v) and (vi), $d(x)$ are highly oscillatory but bounded. All the numerical results are computed by MATLAB on a Sun Enterprise 4000 workstation.

### 4.1. Accuracy of the Approximation Matrix B

We measure the accuracy of our approximation matrix $B$ by computing its relative error $\|A-B\|_F/\|A\|_F$. Since the kernel functions (i), (v) and (vi) give the same approximation matrix $B$, we only give the results for kernel functions (i)–(iv) in this subsection.

(i) *The accuracy of $B$ vs $k$ when $l$ is fixed.*

Figure 3 gives the log of the relative error of $B$ against $k$ for $l=4$ and $l=10$. For other $l$, the results are similar and hence omitted. We note that the relative error of $B$ for the kernel functions $\log|x-t|$ and $\cos(xt^2)\log|x-t|$ are almost the same. Recall that the size of the matrices is $n=2^l k$. Thus the largest matrix we tested is of size 16384-by-16384. We observe from Figure 3 that for a fix $l$, $\log(\|A-B\|_F/\|A\|_F)$ is a linear function of $k$, therefore the relative error of $B$ is of the order $O(e^{-k})$. This result well matches our error estimate (3.13).

(ii) *The accuracy of $B$ vs $l$ when $k$ is fixed.*

In Figure 4, we plot the log of the relative error against $l$ for $k=4$ and $6$.

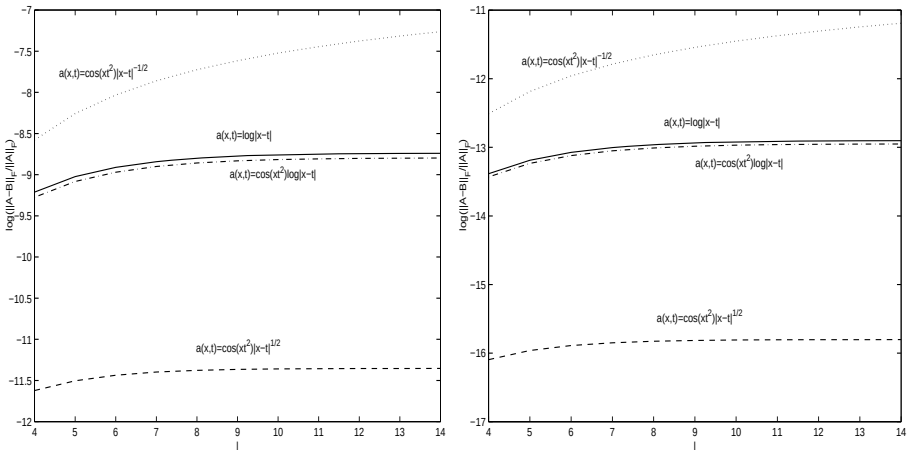


**Figure 4.** $l$ against $\log(\|A-B\|_F/\|A\|_F)$ for $k=4$ (left) and $k=6$ (right)

The results for other $k$ are similar and hence omitted. From the figure, we see that the error for smooth kernels (e.g. $\cos(xt^2)|x-t|^{1/2}$ with $\delta = 1/2$ and $\log|x-t|$ with $\delta = 0$) is almost constant independent of $l$. For kernels with a negative $\delta$ (e.g. $\cos(xt^2)|x-t|^{-1/2}$ with $\delta = -1/2$), the error increases slowly with $l$, indicating that we should use larger $k$ for larger $n$. This is in accord with Theorem 2.

### 4.2. The Conjugate Gradient Method

In the following, we test the complexity of using the conjugate gradient method to solve our approximation matrix equation $(I-DB)\mathbf{y} = \mathbf{b}$.

(i) *Cost per iteration.*

The cost per iteration of the conjugate gradient method will depend mainly on the matrix-vector multiplications. Clearly, for the original matrix $A$, $A\mathbf{y}$ will require $O(n^2)$ operations, whereas by Theorem 1, the cost for $B\mathbf{y}$ is only $O(nk\log n)$. In Figure 5, we plot the log of the costs of forming the products (in Kilo-flops) against $l$. From the slopes of the lines, we clearly see that our method attains the said complexity.

(ii) *Convergence rate.*

We solve the system $(I-DB)\mathbf{y} = \mathbf{b}$ by the CGLS method which is based on solving the normal equation of the given equation by the conjugate gradient method, see [4]. We choose a random vector as our right hand side and the zero vector as our initial guess. The stopping criterion is $\|\mathbf{r}_q\|_2/\|\mathbf{r}_0\|_2 < 10^{-10}$, where $\mathbf{r}_q$ is the residual vector at the $q$th iteration. The numbers of iterations required for convergence are given in Table 4.

Since the kernel functions we tried are at most weakly singular, we see from Table 4 that the convergence rate is linear as expected. As the product $B\mathbf{y}$ can be formed in $O(nk\log n)$ operations, the total cost of solving each of these systems

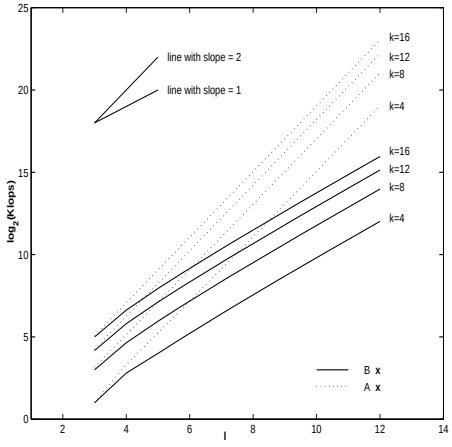


**Figure 5.** $l$ versus $\mathrm{Log}_2(\mathrm{Kflops})$

**Table 4.**
Numbers of Iterations Required for Convergence

| kernel function | $l$ | $k=4$ | $k=8$ | $k=12$ | $k=14$ | $k=16$ |
|---|---|---|---|---|---|---|
| $\cos(xt^2)\lvert x-t\rvert^{-1/2}$ | 4 | 20 | 23 | 25 | 26 | 26 |
| | 6 | 26 | 29 | 32 | 36 | 33 |
| | 8 | 33 | 32 | 32 | 32 | 32 |
| | 10 | 32 | 33 | 33 | 33 | 33 |
| $\cos(xt^2)\lvert x-t\rvert^{1/2}$ | 4 | 9 | 9 | 9 | 9 | 8 |
| | 6 | 8 | 8 | 8 | 8 | 8 |
| | 8 | 8 | 8 | 8 | 8 | 8 |
| | 10 | 8 | 8 | 8 | 8 | 8 |
| $\log\lvert x-t\rvert$<br>$(1+\frac{1}{2}\sin(100x))\log\lvert x-t\rvert$<br>$\cos(xt^2)\log\lvert x-t\rvert$<br>$\sin(100x)\log\lvert x-t\rvert$ | 4<br>6<br>8<br>10 | converges within 12 to 14 iterations for all 4 kernel functions | | | | |

is of $O(nk\log n)$ operations. We emphasize again that in order to get the solution, we only have to form the factors of $B$ which require only $O(nk)$ operations (see Theorem 1) and there is no need to form $A$.

### 4.3. Comparison of Accuracy and Storage Requirement

We now compare the accuracy and storage requirement of our method with those of the wavelet-like method [2]. We choose a fixed random vector $\mathbf{y}$ (by using `rand('seed',37)` in MATLAB) as the true solution and construct the right hand side $\mathbf{b}$ from $(I-DA)\mathbf{y}=\mathbf{b}$. We then solve

$$(I-DB)\mathbf{z}=\mathbf{b}$$

by the CGLS method, where $B$ is our approximation to $A$ as given in (2.12).

For the wavelet-like method, we obtain the sparse representation $S$ of $D^{\frac{1}{2}}AD^{\frac{1}{2}}$ by using the algorithm in [2] to get the discretization matrix in the wavelet-like coordinates and then throwing away entries smaller than the threshold $\tau$ in absolute value. (Thus we see that the wavelet-like method is not applicable to kernel functions such as (vi) where $D$ is indefinite.) Then we solve

$$D^{\frac{1}{2}}(I-S)D^{-\frac{1}{2}}\mathbf{z}=\mathbf{b}$$

by the CGLS method. We remark that the $\mathbf{z}$ so obtained is more accurate than that obtained by the Schulz method in [2] because their method will throw away small entries in the inverse of $(I-S)$.

Tables 5–6 show the storage requirement for the approximation matrices $B$ and $S$ and the accuracy of the solutions. The relative errors $\|\mathbf{y}-\mathbf{z}\|_2/\|\mathbf{y}\|_2$ of the solutions are given under the column "Error". The column "$\omega$" gives the *bandwidth* of the matrices which is defined to be the storage/$n$. We note that by Theorem 1, the bandwidth of our matrix $B$ is bounded by $9.5k$ (the number enclosed by parenthesis in Tables 5–6), and is independent of the kernel functions. In contrast, the bound for $S$ is $c\log n$, where the constant $c$ depends on the threshold $\tau$ and the kernel function considered, see [2]. We remark further that in the wavelet-like method, additional storage of $O(nk)$ is required to store the wavelet-like bases matrix $U=U_lU_{l-1}\cdots U_1$, see [2]. However, these counts are not added in Tables 5–6 for clarity.

From Tables 5–6 we observe the following:

1. For the same $k$, the solution obtained by our method is always more accurate than that of the wavelet-like method. The reasons are that the interpolation polynomial on Chebyshev points is more accurate than that on uniformly-spaced points, and the wavelet-like method discards small entries to get the sparse representation $S$. In Table 5, we tried two different thresholds $\tau$ to show that the accuracy of the wavelet-like method is not improved much by using a smaller $\tau$. We note that it will be very expensive to use Chebyshev points in the wavelet-like method.

**Table 5.**
Bandwidth $\omega$ and Relative Error for $\log|x-t|$

| | | Our Method | | Wavelet-Like Method | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $k$ | $l$ | $\omega$ | Error | $\tau$ | $\omega$ | Error | $\tau$ | $\omega$ | Error |
| 4 | 4 | 29.8 | 3.19e-5 | $10^{-5}$ | 26.2 | 6.89e-5 | $10^{-6}$ | 33.3 | 7.06e-5 |
| | 5 | 33.1 | 3.23e-5 | | 28.7 | 6.61e-5 | | 39.7 | 6.85e-5 |
| | 6 | 35.2 | 3.10e-5 | | 29.0 | 6.11e-5 | | 42.1 | 6.42e-5 |
| | 7 | 36.4 | 3.00e-5 | | 26.0 | 6.22e-5 | | 40.9 | 6.35e-5 |
| | 8 | 37.1 | 2.88e-5 | | 22.3 | 6.14e-5 | | 38.5 | 6.22e-5 |
| | | ($\leqslant$ 38.0) | | | | | | | |
| 5 | 4 | 36.9 | 2.62e-6 | $10^{-6}$ | 34.5 | 6.49e-6 | $10^{-7}$ | 40.8 | 6.51e-6 |
| | 5 | 41.3 | 2.67e-6 | | 39.1 | 5.85e-6 | | 49.4 | 5.90e-6 |
| | 6 | 43.9 | 2.76e-6 | | 40.2 | 5.99e-6 | | 53.1 | 5.88e-6 |
| | 7 | 45.5 | 2.88e-6 | | 38.8 | 6.30e-6 | | 53.4 | 6.07e-6 |
| | 8 | 46.4 | 2.91e-6 | | 36.3 | 6.08e-6 | | 51.8 | 5.92e-6 |
| | | ($\leqslant$ 47.5) | | | | | | | |
| 6 | 4 | 44.3 | 3.30e-7 | $10^{-7}$ | 43.6 | 1.55e-6 | $10^{-8}$ | 48.7 | 1.55e-6 |
| | 5 | 49.5 | 3.58e-7 | | 49.9 | 1.43e-6 | | 59.7 | 1.46e-6 |
| | 6 | 52.7 | 4.14e-7 | | 53.3 | 1.47e-6 | | 66.5 | 1.52e-6 |
| | 7 | 54.6 | 4.11e-7 | | 53.4 | 1.40e-6 | | 67.4 | 1.47e-6 |
| | 8 | 55.6 | 4.13e-7 | | 51.0 | 1.40e-6 | | 65.9 | 1.44e-6 |
| | | ($\leqslant$ 57.0) | | | | | | | |
| 7 | 4 | 51.6 | 4.03e-8 | $10^{-8}$ | 52.1 | 1.78e-7 | $10^{-9}$ | 57.6 | 1.80e-7 |
| | 5 | 57.8 | 4.06e-8 | | 60.6 | 1.66e-7 | | 70.3 | 1.66e-7 |
| | 6 | 61.5 | 4.41e-8 | | 65.1 | 1.61e-7 | | 78.2 | 1.61e-7 |
| | 7 | 63.7 | 4.36e-8 | | 66.8 | 1.61e-7 | | 80.6 | 1.61e-7 |
| | 8 | 64.9 | 4.37e-8 | | 65.7 | 1.56e-7 | | 80.0 | 1.58e-7 |
| | | ($\leqslant$ 66.5) | | | | | | | |
| 8 | 4 | 59.5 | 6.04e-9 | $10^{-9}$ | 60.1 | 4.54e-8 | $10^{-10}$ | 65.7 | 4.54e-8 |
| | 5 | 66.3 | 6.56e-9 | | 72.2 | 4.53e-8 | | 81.1 | 4.53e-8 |
| | 6 | 70.4 | 6.93e-9 | | 77.7 | 4.56e-8 | | 91.4 | 4.57e-8 |
| | 7 | 72.8 | 6.81e-9 | | 80.1 | 4.38e-8 | | 95.7 | 4.41e-8 |
| | 8 | 74.2 | 6.83e-9 | | 80.5 | 4.34e-8 | | 96.5 | 4.37e-8 |
| | | ($\leqslant$ 76.0) | | | | | | | |

**Table 6.**
Bandwidth and Relative Error for Kernel Functions (iii) $\cos(xt^2)|x-t|^{-1/2}$ and (v) $(1+\frac{1}{2}\sin(100x))\log|x-t|$

| | | Our Method | | | Wavelet-Like Method | | | | |
|---|---|---|---|---|---|---|---|---|---|
| kernel | | | (iii) | (v) | | (iii) | | (v) | |
| $k$ | $l$ | $\omega$ | Error | Error | $\tau$ | $\omega$ | Error | $\omega$ | Error |
| 4 | 4 | 29.8 | 5.67e-5 | 3.25e-5 | $10^{-5}$ | 32.3 | 1.46e-4 | 34.7 | 7.45e-5 |
| | 5 | 33.1 | 7.71e-5 | 3.15e-5 | | 37.9 | 1.68e-4 | 39.8 | 7.29e-5 |
| | 6 | 35.2 | 7.21e-5 | 3.10e-5 | | 41.1 | 1.72e-4 | 32.6 | 6.53e-5 |
| | 7 | 36.4 | 1.18e-4 | 3.06e-5 | | 41.6 | 2.14e-4 | 29.5 | 6.69e-5 |
| | 8 | 37.1 | 2.57e-4 | 2.98e-5 | | 41.7 | 8.63e-4 | 25.3 | 6.43e-5 |
| | | ($\leqslant$38.0) | | | | | | | |
| 5 | 4 | 36.9 | 4.66e-6 | 2.88e-6 | $10^{-6}$ | 41.2 | 1.30e-5 | 44.5 | 6.40e-6 |
| | 5 | 41.3 | 7.10e-6 | 2.89e-6 | | 48.7 | 1.89e-5 | 51.1 | 5.92e-6 |
| | 6 | 43.9 | 8.03e-6 | 2.97e-6 | | 53.4 | 1.63e-5 | 46.3 | 5.94e-6 |
| | 7 | 45.5 | 7.70e-6 | 3.08e-6 | | 55.4 | 7.76e-5 | 44.4 | 6.69e-6 |
| | 8 | 46.4 | 7.57e-6 | 3.12e-6 | | 55.7 | 5.19e-5 | 40.5 | 7.04e-6 |
| | | ($\leqslant$47.5) | | | | | | | |
| 6 | 4 | 44.3 | 7.46e-7 | 3.45e-7 | $10^{-7}$ | 49.9 | 4.20e-6 | 54.2 | 1.59e-6 |
| | 5 | 49.5 | 8.97e-7 | 3.53e-7 | | 60.4 | 4.88e-6 | 62.2 | 1.45e-6 |
| | 6 | 52.7 | 1.40e-6 | 4.18e-7 | | 66.6 | 6.17e-6 | 62.3 | 1.43e-6 |
| | 7 | 54.6 | 4.65e-6 | 4.19e-7 | | 69.6 | 2.03e-5 | 60.3 | 1.35e-6 |
| | 8 | 55.6 | 1.83e-6 | 4.22e-7 | | 70.8 | 9.59e-6 | 56.0 | 1.34e-6 |
| | | ($\leqslant$57.0) | | | | | | | |
| 7 | 4 | 51.6 | 9.43e-8 | 4.32e-8 | $10^{-8}$ | 59.0 | 4.43e-7 | 64.2 | 1.75e-7 |
| | 5 | 57.8 | 1.18e-7 | 4.26e-8 | | 71.2 | 4.92e-7 | 73.8 | 1.58e-7 |
| | 6 | 61.5 | 1.25e-7 | 4.67e-8 | | 79.1 | 1.38e-6 | 75.6 | 1.58e-7 |
| | 7 | 63.7 | 2.15e-6 | 4.65e-8 | | 83.7 | 2.82e-5 | 74.9 | 1.55e-7 |
| | 8 | 64.9 | 1.33e-7 | 4.64e-8 | | 85.3 | 9.55e-7 | 71.5 | 1.54e-7 |
| | | ($\leqslant$66.5) | | | | | | | |
| 8 | 4 | 59.5 | 1.71e-8 | 6.02e-9 | $10^{-9}$ | 68.0 | 1.43e-7 | 73.7 | 4.63e-8 |
| | 5 | 66.3 | 2.15e-8 | 6.73e-9 | | 83.0 | 1.73e-7 | 85.6 | 4.26e-8 |
| | 6 | 70.4 | 3.28e-8 | 7.04e-9 | | 93.2 | 1.97e-7 | 89.9 | 4.25e-8 |
| | 7 | 72.8 | 9.43e-8 | 7.20e-9 | | 99.3 | 7.33e-7 | 89.7 | 4.07e-8 |
| | 8 | 74.2 | 2.88e-8 | 7.18e-9 | | 103.0 | 2.61e-7 | 86.9 | 4.01e-8 |
| | | ($\leqslant$76.0) | | | | | | | |

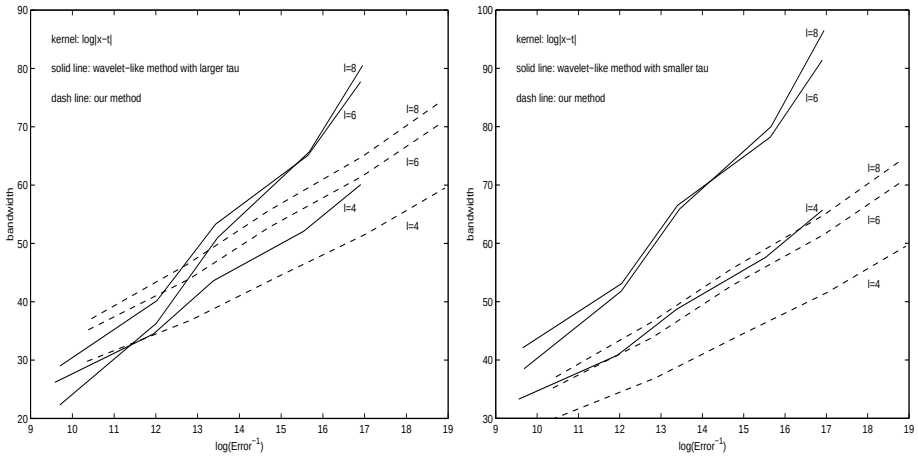


**Figure 6.** Accuracy versus Bandwidth for $\log|x-t|$

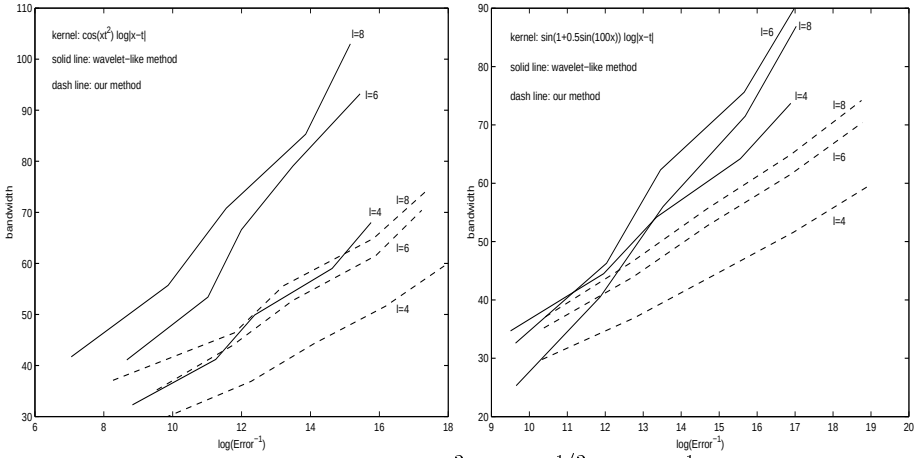


**Figure 7.** Accuracy versus Bandwidth for $\cos(xt^2)\,|x-t|^{-1/2}$ and $(1+\frac{1}{2}\sin(100x))\log|x-t|$

2. The bandwidth $\omega$ of the sparse matrix $S$ in the wavelet-like method increases more than linearly with respect to $\log(\tau^{-1})$ for fixed $l$. To illustrate this more clearly, we plot in Figures 6–7 the accuracy ($\log$(Error)) against the bandwidth $\omega$ for both methods. For clarity, we plot only the case where $l = 4, 6$ and 8. We see from the figures that the storage requirement of our method grows linearly with increasing accuracy (cf. (3.13)), whereas that of the wavelet-like method increases more than linearly. Moreover, the increment (slope) grows more rapidly with increasing $l$ for the wavelet-like method.

Finally we remark that in order to get numerical solution of high order accuracy, it is useful to apply the Kantorovitch method in conjunction with higher-order

**Table 7.**
Errors

| $l$ | 4 | 5 | 6 | 7 |
|---|---|---|---|---|
| $T$ | 0.0208 | 0.0117 | 0.0065 | 0.0036 |
| $KT$ | 1.9577e-5 | 4.5975e-6 | 1.1019e-6 | 2.6813e-7 |
| $KTA$ | 1.9569e-5 | 4.6954e-6 | 1.1013e-6 | 2.6769e-7 |

quadrature rules, say, trapezoidal rule. For instance, we can write (1.1) as

$$\left(1 - d(x)\int_0^1 a(x,t)dt\right) f(x) - d(x)\int_0^1 a(x,t)(f(t) - f(x))dt = g(x),$$

for all $x \in [0,1]$, and compute $\int_0^1 a(x,t)dt$ accurately and discretize the integral operator $\int_0^1 a(x,t)(f(t) - f(x))dt$ by trapezoidal rule. As an example, we consider the equation

$$f(x) - \int_0^1 \log|x-t| f(t)dt = g(t), \quad x \in [0,1] \tag{4.1}$$

where $g(t)$ is chosen such that the exact solution is given by $f(x) = x^2$.

Let $T$ denote that the integral operator $\int_0^1 \log|x-t| f(t)dt$ is discretized by trapezoidal rule directly. Let $KT$ denote that (4.1) is discretized by Kantorovitch method in conjunction with trapezoidal rule. Let $KTA$ denote the combination of $KT$ and our approximation scheme ($k = 8$). Define the errors in a numerical solution $f_c$ by

$$\frac{\sqrt{\sum_{i=1}^n [f_c(x_i) - f(x_i)]^2}}{\sqrt{\sum_{i=1}^n [f(x_i)]^2}}.$$

The errors of the three methods are shown in Table 7. From the results we see that the method $KTA$ is quite accurate.

## 5. CONCLUDING REMARKS

We have developed a fast multiplication scheme for integral equations of the second kind in this paper. Theorems 1 and 2 indicate that for a given accuracy, the memory requirement of our method is of order $O(nk)$, where $k$ is fixed for smooth kernels (including $\log|x-t|$) and grows like $O(\log\log n)$ for weakly singular kernel functions. Numerical results show that our scheme is more accurate and requires less storage than that of the wavelet-like method in [2].

We remark that our scheme can be extended straightforwardly to Fredholm integral equations of the first kind. For these problems, a main issue is the ill-conditioned nature of the operators. If the problem is not too ill-conditioned, such as in the cases of potential equations or Helmholtz equations in integral form, then

it can be overcome by preconditioning. Our approximation matrix $B$ given in the form of (2.12) makes it easy to construct optimal circulant preconditioners for $B$, see [5]. In fact, for potential equations in integral form, where the condition number of the problem is $O(n)$, we have proved that using the optimal circulant preconditioner for $B$, the preconditioned system is well-conditioned with condition number $O(1)$, see [6].

## REFERENCES


1. B. Alpert, A class of bases in $L^2$ for the sparse representation of integral operators. *SIAM. J. Math. Anal.*, (1993) **24**, 246–262.
2. B. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin, Wavelet-like bases for the fast solution of second-kind integral equations. *SIAM J. Sci. Comput.*, (1993) **14**, 159–184.
3. G. Beylkin, R. Coifman, and V. Rokhlin, Fast wavelet transforms and numerical algorithms I. *Comm. Pure Appl. Math.*, (1991) **46**, 141–183.
4. A. Björck, *Least Squares Methods, Handbook of Numerical Methods*, P. Ciarlet and J. Lions, ed., V 1, Elsevier, North-Holland, 1989.
5. R. Chan, W. Ng, and H. Sun, Fast construction of optimal circulant preconditioners for matrices from fast dense matrix method. BIT, (2000) **40**, 24–40.
6. R. Chan, H. Sun, and W. Ng, Circulant preconditioners for ill-conditioned boundary integral equations from potential equations. *Int. J. Numer. Meth. Engng.*, (1998) **43**, 1505–1521.
7. L. Delves and J. Mohamed, *Computational Methods for Integral Equations*. Cambridge University Press, Cambridge, 1985.
8. G. Golub and C. Van Loan, *Matrix Computations* (2nd ed.). John Hopkins University Press, Baltimore, 1989.
9. L. Greengard and V. Rokhlin, A fast algorithm for particle simulations. *J. Comput. Phys.*, (1987) **73**, 325–348.
10. D. Kincaid and W. Cheney, *Numerical Analysis*. Brooks/Cole Publishing Company, Pacific Grove, California, 1990.
11. R. Kress, *Linear Integral Equations*. Applied Mathematical Sciences, V 82, Springer-Verlag, New York, 1989.
12. S. Myagchilov and E. Tyrtyshnikov, A fast matrix-vector multiplier in discrete vortex method. *Russian J. Numer. Anal. Math. Modelling*, (1992) **7**, No.4, 325–342.
13. L. Reichel, Fast solution methods for Fredholm integral equations of the second kind. *Numer. Math.*, (1989) **57**, 719–736.
14. W. Rudin, *Functional Analysis* (2nd ed.). McGraw-Hill, New York, 1991.
15. E. Tyrtyshnikov, Mosaic-skeleton approximations. *Calcolo*, (1996) **33**, 47–57.
16. E. Tyrtyshnikov, Mosaic ranks and skeletons. *Lect. Notes Comput. Sc.*, (1997) **1196**, 505–526.