We describe an algorithm for the rapid direct solution of linear algebraic systems arising from the discretization of boundary integral equations of potential theory in two dimensions. The algorithm is combined with a scheme that adaptively rearranges the parameterization of the boundary in order to minimize the ranks of the off-diagonal blocks in the discretized operator, thus obviating the need for the user to supply a parameterization r of the boundary for which the distance $\|r(s)-r(t)\|$ between two points on the boundary is related to their corresponding distance $|r-s|$ in the parameter space. The algorithm has an asymptotic complexity of $O(n \log^2 n)$, where n is the number of nodes in the discretization. The performance of the algorithm is illustrated with several numerical examples.

An adaptive fast direct solver for boundary integral equations in two dimensions

W. Y. Kong, J. Bremer [†], and V. Rokhlin [*]
Technical Report YaleU/DCS/TR1418
August 21, 2009

[†] Supported in part by the Office of Naval Research under contract N00014-09-1-0318.
[*] Supported in part by the Air Force Office of Scientific Research under contract FA9550-09-1-0241, in part by the Office of Naval Research under contract N00014-07-1-0711, and in part by Schlumberger Limited under contract 1040834.1.R07554.622002.
Keywords: *Boundary value problems, Boundary integral equations, Layer potentials, Laplace's equation, Fast solver*

1 Introduction

Integral equations are one of principal tools in the analysis and solution of boundary value problems for elliptic partial differential equations. In particular, one of the standard approaches to the numerical treatment of boundary value problems for elliptic partial differential equations calls for reformulating them as boundary integral equations, discretizing the associated integral operators, and solving the resulting linear systems in order to obtain solutions in the form of layer potentials. This approach has been widely studied, especially in the context of Laplace and Helmholtz boundary value problems on smooth domains. Traditionally, an iterative solver was coupled with the appropriate fast multipole method (for example, [10]) in order to solve the discrete linear systems arising from the boundary integral equations. For problems associated with the Laplace and Helmholtz equations, the asymptotic complexity of this approach is $O(n)$ and $O(n \log n)$ respectively, with n the number of nodes in the discretization.

More recently, a number of "fast direct solvers" was developed for the solution of linear systems arising in various environments (see, for example, [4, 5, 19, 8]). Most of these schemes are based on the observation that the matrices in question have a hierarchical structure involving rank-deficient off-diagonal blocks. Matrices with such structure commonly arise from the boundary integral operators of potential theory and, in particular, the authors in [19] describe a fast direct solver for boundary integral equations in two dimensions that make use of such structure. Hierarchically rank-deficient matrices have been studied in a number of other contexts (see, for instance, [12, 13, 3]), and they are strongly related to the class of "generalized" Calderón-Zygmund operators characterized by having integral kernels $K(x, y)$ which are smooth for x well-separated from y (see [6]).

The solver in [19] operates by constructing a two-sided hierarchical factorization of the inverse of a discretized boundary integral operator. When applied to boundary integral equations of potential theory in two dimensions, the solver has an asymptotic complexity of $O(n)$, with n the number of nodes used to discretize the integral equation. The two-sided factorization, which appears to be necessary in order to obtain an algorithm that is asymptotically $O(n)$, is relatively complicated and makes the algorithm difficult to implement. Moreover, the solver in [19] relies on certain strong assumptions about the regions on which the PDEs are to be solved. In particular, it assumes that the boundary of the region is specified via a parameterization $r : [0, 1] \rightarrow \mathbb{R}^2$ such that the distance $\|r(s) - r(t)\|$ between two points on the curve is related to their corresponding distance $|r - s|$ in the parameter space. For certain complicated curves such parameterizations can be difficult to obtain, and in the absence of a parameterization with this property the resulting discretized operator can exhibit high rank off-diagonal blocks. This becomes an even more serious problem for boundary integral equations in three-dimensions since for a parameterization r of a boundary surface $\|r(s_1, t_1) - r(s_2, t_2)\|$ generally bears no relation to the Euclidean distance between (s_1, t_1) and (s_2, t_2).

In this paper, we introduce a simple direct solver that is similar to the one in [19], but operates by constructing a one-sided hierarchical factorization of the inverse of a matrix. When applied to a matrix with rank deficient off-diagonal blocks and no other structure, the solver is asymptotically $O(n^2)$ in the dimension n of the matrix. When applied to boundary integral equations in two dimensions arising from partial differential equations for which a Green's function is non-oscillatory (or, weakly oscillatory, as defined in [19]), the complexity of the solver is reduced to $O(n \log^2 n)$. Not only is this solver considerably simpler to implement than that of [19], but it also addresses an important weakness of that solver. That is, it includes an adaptive rotation scheme that rearranges the parameterization of the boundary in order to minimize the ranks of the off-diagonal blocks. This scheme obviates the need for the user to supply a parameterization r of the boundary curve for which $\|r(t) - r(s)\|$ is related to $|t - s|$ and thereby expands the class of regions to which the solver is applicable. Moreover, the scheme generalizes readily to the three dimensional setting, where it is expected to be a useful tool.

The paper is structured as follows. Section 2 provides the necessary mathematical and numerical

preliminaries. Section 3 reviews solution of boundary value problems for Laplace's equation via boundary integral equations. In Section 4, we introduce the multi-level algorithm for the construction of a compressed factorization of the inverse of a matrix; the algorithm applies generally to matrices with rank deficient off-diagonal blocks. Section 5 contains a formal description of the algorithm outlined in Section 4 and assesses its computational cost. In Section 6, we show how the algorithm presented in Section 5 can be accelerated when the matrix arises from the discretization of a boundary integral operator. Section 7 illustrates through numerical examples the performance of the algorithm when applied to boundary integral equations. Finally, in Section 8, we summarize the work and discuss possible extensions and generalizations.

2 Mathematical and numerical preliminaries

Throughout the paper we use the following notation. Given a matrix X, we let X^* denote its adjoint (the complex conjugate transpose), $\sigma_k(X)$ its kth singular value, $\|X\|_2$ its l^2-norm, and $\|X\|_F$ its Frobenius norm. Finally, given matrices A, B, C, and D, we let

$$\begin{pmatrix} A & B \end{pmatrix}, \begin{pmatrix} A \\ C \end{pmatrix}, \text{ and } \begin{pmatrix} A & B \\ C & D \end{pmatrix} \tag{2.1}$$

denote larger matrices obtained by combining the blocks A, B, C, and D.

2.1 Singular value decomposition

The singular value decomposition (SVD) is a ubiquitous tool in numerical analysis, provided in the case of real matrices by the following lemma (see, for instance, [21] for more details).

Lemma 2.1 *(SVD) For any $n \times m$ real matrix A, there exist, for some integer p, an $n \times p$ real matrix U with orthonormal columns, an $m \times p$ real matrix V with orthonormal columns, and a $p \times p$ real diagonal matrix Σ with positive diagonal entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p > 0$, such that $A = U\Sigma V^*$.*

The diagonal entries σ_i of Σ are called singular values, the columns u_i of the matrix U are called the left singular vectors, and the columns v_i of the matrix V are called the right singular vectors. The number p is called the (mathematical) rank of A. Note that the SVD of A can be written as

$$A = U\Sigma V^* = \sum_{i=1}^{p} \sigma_i u_i v_i^*, \tag{2.2}$$

where

$$U = [u_1, \ldots, u_p] \text{ and } V = [v_1, \ldots, v_p]; \tag{2.3}$$

obviously, (2.2) provides a decomposition of the matrix A into a sum of p rank one matrices.

A common application of the SVD is for the approximations of matrices, as described by the following lemma (see, for instance, [1]).

Lemma 2.2 *Suppose $A \in \mathbb{R}^{n \times m}$ has the SVD*

$$A = U\Sigma V^* = \sum_{i=1}^{p} \sigma_i u_i v_i^*, \tag{2.4}$$

and the matrix $B \in \mathbb{R}^{n \times m}$ is defined by the formula

$$B = \sum_{i=1}^{k} \sigma_i u_i v_i^*, \tag{2.5}$$

where k is an integer with $1 \leq k \leq p$. Then

$$\min ||A - A_k||_2 = ||A - B||_2 = \sigma_{k+1}, \tag{2.6}$$

where A_k ranges over the set of all $n \times m$ matrices of rank k. In other words, B is the best rank k approximation of A.

The SVD allows us to introduce the concept of the numerical rank of a matrix. For some small ε, we define the ε-rank of a matrix A via the formula

$$\text{rank}(A, \varepsilon) = \min_{||A-B||_2 \leq \varepsilon} \text{rank}(B) \tag{2.7}$$

(see [9]). In other words, $\text{rank}(A, \varepsilon)$ equals k if and only if there are exactly k singular values of A that lie above ε, i.e.,

$$\sigma_k > \varepsilon \geq \sigma_{k+1}, \tag{2.8}$$

with σ_{p+1} defined to be 0.

2.2 QR decomposition

The singular value decomposition provides the optimal rank k approximation to a given matrix; however, the SVD is relatively expensive to construct, and other, less computationally expensive, matrix factorizations are often used.

Given a real $m \times n$ matrix M, $l = \min(m, n)$, and an integer k with $0 < k \leq l$, the classical QR decomposition (see, for instance, [9]) constructs a factorization of the form

$$M\Pi = QR \equiv Q \begin{pmatrix} A_k & B_k \\ 0 & C_k \end{pmatrix}, \tag{2.9}$$

where $\Pi \in \mathbb{R}^{n \times n}$ is a permutation matrix, $Q \in \mathbb{R}^{m \times l}$ has orthonormal columns, $R \in \mathbb{R}^{l \times n}$, $A_k \in \mathbb{R}^{k \times k}$ is upper triangular with nonnegative diagonal entries, $B_k \in \mathbb{R}^{k \times (n-k)}$, and $C_k \in \mathbb{R}^{(l-k) \times (n-k)}$.

The following theorem can be found, in a slightly different form, in [11]. It asserts that, given any real $m \times n$ matrix M, there exists a factorization of the form (2.9) satisfying inequalities such that (2.9) provides a reasonable means to detect the numerical rank of M.

Theorem 2.3 *(Gu and Eisenstat) Suppose that M is a real $m \times n$ matrix, $l = \min(m, n)$, and M has p singular values. Then for any integer k with $0 < k \leq p$, there exists a factorization of the form (2.9) such that*

$$\sigma_k(A_k) \geq \frac{1}{\sqrt{1 + k(n-k)}} \sigma_k(M), \tag{2.10}$$

$$||C_k||_2 \leq \sqrt{1 + k(n-k)}\, \sigma_{k+1}(M), \tag{2.11}$$

$$||A_k^{-1} B_k||_F \leq \sqrt{k(n-k)}. \tag{2.12}$$

Theorem 2.3 implies that if $M \in \mathbb{R}^{m \times n}$ is a matrix of numerical rank k to precision ε, there exists a permutation matrix $\Pi \in \mathbb{R}^{n \times n}$ such that the first k columns of $M\Pi$ form a well-conditioned basis for the column space of M, to within ε. Let $j_1, j_2, \ldots, j_k$ be the column indices of M corresponding to the first k columns of $M\Pi$; then we will refer to the $m \times k$ matrix consisting of the columns of M numbered $j_1, j_2, \ldots, j_k$ as a *column skeleton* of M. Furthermore, in this case the inequality (2.11) implies that M can be accurately approximated by a matrix of rank k. Specifically,

$$M \approx \tilde{Q} S, \tag{2.13}$$

where $\tilde{Q}$ is the $m \times k$ matrix formed by the first k columns of Q in (2.9) and S is a $k \times n$ matrix defined by the formula

$$S = (\ A_k \quad B_k\ )\Pi^*, \tag{2.14}$$

and

$$\|M - \tilde{Q}S\|_2 \leq \sqrt{1 + k(n-k)}\varepsilon. \tag{2.15}$$

Remark 2.1 *While Theorem 2.3 asserts the existence of a QR decomposition of the form (2.9) satisfying (2.10)-(2.12), it does not address the question of how to construct it numerically. In [11], a robust, provably stable algorithm is presented that constructs a QR decomposition of the form (2.9), with (2.10)-(2.12) replaced by the weaker inequalities:*

$$\sigma_k(A_k) \geq \frac{1}{\sqrt{1 + nk(n-k)}}\sigma_k(M), \tag{2.16}$$

$$\|C_k\|_2 \leq \sqrt{1 + nk(n-k)}\sigma_{k+1}(M), \tag{2.17}$$

$$\|A_k^{-1}B_k\|_F \leq \sqrt{nk(n-k)}. \tag{2.18}$$

In this paper, we use the pivoted Gram-Schmidt algorithm (with reorthogonalization) described in [1] to construct factorizations of the form (2.13). While there are no guaranteed bounds of the form (2.15) for this algorithm, it is simple to implement and does well in practice. In particular, one applies the pivoted Gram-Schmidt algorithm to the columns of M, halting the procedure when the l^2-norm of the remaining columns falls below a preset threshold ε. This procedure computes a column skeleton for M, and if M is an $m \times n$ matrix with numerical rank k, it requires $O(mnk)$ operations. After that, a factorization of the form (2.13) is computed in $O(k^2(m+n))$ operations.

2.3 Randomized algorithms for the approximation of matrices

In [23], a randomized algorithm is presented that constructs a low-rank approximation to a matrix A in the form $A \approx \tilde{A}P$, where $\tilde{A}$ is a column skeleton of A.

Suppose A is an $m \times n$ matrix, and l and k are positive integers with $k < l < \min(m, n)$. The algorithm of [23] involves applying an $l \times n$ random matrix Φ to A, and then constructing a decomposition of the form

$$\Phi A \approx \tilde{B}P, \tag{2.19}$$

where $\tilde{B}$ is a column skeleton of ΦA, consisting of k columns of ΦA with indices $j_1, j_2, \ldots, j_k$ (for details on the exact form of Φ, see [23]). If we let $\tilde{A}$ be the $m \times k$ matrix formed by collecting the k columns of A with the same indices, then the product $\tilde{A}P$ provides an approximation to A such that

$$\|A - \tilde{A}P\|_2 \leq \sqrt{mnk}\sigma_{k+1}(A), \tag{2.20}$$

with very high probability. The probability p of (2.20) is a function of $l - k$; its actual estimates are detailed, and the reader is referred to [23] for them. Here we merely observe that (for example) $l - k = 20$ yields $p > 1 - 10^{-17}$.

The randomized approach of [23] accelerates the process of finding a column skeleton of a matrix for the purpose of constructing its QR decomposition. In order to find a column skeleton of an $m \times n$ matrix A with numerical rank k, one can first form the $l \times n$ matrix ΦA, which can be done in $O(mn \log l)$ operations (see [23]), and then apply the pivoted Gram-Schmidt algorithm to ΦA. The procedure involves a total cost of $O(mn \log l + lnk)$, and this is less expensive than applying the pivoted Gram-Schmidt algorithm directly to A, which has a cost of $O(mnk)$.

2.4 Sherman-Morrison-Woodbury formula

The Sherman-Morrison-Woodbury formula provides an expression for the inverse of a low-rank perturbation of an invertible matrix. It can be found, for example, in [9].

Lemma 2.4 *Suppose that A is an invertible $n \times n$ matrix, and that U and V are $n \times k$ matrices. Then*

$$(A + UV^*)^{-1} = A^{-1} - A^{-1}U(I + V^*A^{-1}U)^{-1}V^*A^{-1}, \tag{2.21}$$

assuming that the matrix $(I + V^*A^{-1}U)$ is invertible.

The Sherman-Morrison-Woodbury formula implies that a rank k perturbation to a matrix results in a rank k perturbation to the inverse. In the case that A is an $n \times n$ identity matrix I, (2.21) reduces to the following convenient form:

$$(I + UV^*)^{-1} = I - U(I + V^*U)^{-1}V^*. \tag{2.22}$$

Note that the second I appearing in the right-hand side of (2.22) is a $k \times k$ identity matrix.

3 Boundary integral formulations

In this section, we briefly outline the solution of certain boundary value problems for Laplace's equation via integral equation methods. Thorough treatment of the classical theory can be found in [17, 20, 7, 15]. Extension of the classical theory to the case of Lipschitz domains is discussed in [16, 22, 6].

Throughout this section, Ω will denote a bounded, smooth, simply connected domain in the plane with boundary $\partial\Omega$, Ω^c will denote the open region in the plane exterior to Ω, and dS will denote integration with respect to the arclength measure on $\partial\Omega$.

3.1 Interior Dirichlet problem

The *interior Dirichlet problem* calls for the determination of a function harmonic in Ω with prescribed values on the boundary $\partial\Omega$. That is, given a continuous $f : \partial\Omega \to \mathbb{R}$, we seek a function $u : \Omega \to \mathbb{R}$ such that

$$\begin{aligned} \Delta u(x) &= 0 \text{ for } x \in \Omega, \\ \lim_{\substack{x \to p \\ x \in \Omega}} u(x) &= f(p) \text{ for } p \in \partial\Omega. \end{aligned} \tag{3.1}$$

As is well-known, such a problem has a unique solution that can be represented as the potential of a dipole distribution σ on $\partial\Omega$:

$$u(x) = \frac{1}{2\pi} \int_{\partial\Omega} \sigma(y) \frac{\partial}{\partial \nu_y} \log|x - y| dS(y), \tag{3.2}$$

where $\frac{\partial}{\partial \nu_y}$ denotes the outward normal derivative taken at the point y. In particular, the function $u(x)$ defined by (3.2) is harmonic in Ω and the limit of $u(x)$ as x approaches the point $p \in \partial\Omega$ from the interior of Ω is given by the jump relation

$$\lim_{\substack{x \to p \\ x \in \Omega}} u(x) = \frac{1}{2}\sigma(p) + \frac{1}{2\pi} \int_{\partial\Omega} \sigma(y) \frac{\partial}{\partial \nu_y} \log|p - y| dS(y). \tag{3.3}$$

Thus, if σ satisfies the integral equation

$$\frac{1}{2}\sigma(p) + \frac{1}{2\pi} \int_{\partial\Omega} \sigma(y) \frac{\partial}{\partial \nu_y} \log|p - y| dS(y) = f(p) \tag{3.4}$$

for all $p \in \partial\Omega$, then the function $u(x)$ given by (3.2) is a solution to problem (3.1).

3.2 Exterior Dirichlet problem

The *exterior Dirichlet problem*, which consists of finding a function $u : \Omega^c \to \mathbb{R}$ such that

$$\begin{aligned} \Delta u(x) &= 0 \text{ for } x \in \Omega^c, \\ \lim_{\substack{x \to p \\ x \in \Omega^c}} u(x) &= f(p) \text{ for } p \in \partial\Omega, \end{aligned} \tag{3.5}$$

has a unique solution under the additional assumption that u behaves as $O(1)$ at infinity.

A difficulty arises, however, in applying the approach of the preceding section. Namely, the integral equation resulting from the jump relation for the exterior domain is not uniquely solvable. In particular, if we represent the solution u of (3.5) in the form (3.2):

$$u(x) = \frac{1}{2\pi} \int_{\partial\Omega} \sigma(y) \frac{\partial}{\partial \nu_y} \log|x - y| dS(y), \tag{3.6}$$

then the limit of $u(x)$ as x goes to $p \in \partial\Omega$ from Ω^c is

$$\lim_{\substack{x \to p \\ x \in \Omega^c}} u(x) = -\frac{1}{2}\sigma(p) + \frac{1}{2\pi} \int_{\partial\Omega} \sigma(y) \frac{\partial}{\partial \nu_y} \log|p - y| dS(y). \tag{3.7}$$

This leads to the integral equation

$$-\frac{1}{2}\sigma(p) + \frac{1}{2\pi} \int_{\partial\Omega} \sigma(y) \frac{\partial}{\partial \nu_y} \log|p - y| dS(y) = f(p). \tag{3.8}$$

The operator appearing on the left-hand side of equation (3.8) has a one-dimensional null space; more specifically,

$$-\frac{1}{2} + \frac{1}{2\pi} \int_{\partial\Omega} \frac{\partial}{\partial \nu_y} \log|p - y| dS(y) = 0, \text{ for all } p \in \partial\Omega. \tag{3.9}$$

Note that the exterior Dirichlet problem itself has a unique solution u, but the corresponding dipole distribution σ in representation (3.6) is only determined up to a constant because

$$\int_{\partial\Omega} \frac{\partial}{\partial \nu_y} \log|p - y| \, dS(y) = 0, \text{ for } p \in \Omega^c. \tag{3.10}$$

To overcome this difficulty, we use the modified double-layer potential to represent the solution u:

$$u(x) = \frac{1}{2\pi} \int_{\partial\Omega} \sigma(y) \left(\frac{\partial}{\partial \nu_y} \log|x - y| + 1 \right) dS(y). \tag{3.11}$$

This leads to the integral equation

$$-\frac{1}{2}\sigma(p) + \frac{1}{2\pi} \int_{\partial\Omega} \sigma(y) \left(\frac{\partial}{\partial \nu_y} \log|p - y| + 1 \right) dS(y) = f(p). \tag{3.12}$$

It is shown in [17] that (3.12) has a unique solution σ which, when inserted into (3.11), produces the (unique) solution of (3.5).

3.3 Exterior Neumann problem

The *exterior Neumann problem*

$$\begin{aligned} \Delta u(x) &= 0 \text{ for } x \in \Omega^c, \\ \lim_{\substack{x \to p \\ x \in \Omega^c}} \frac{\partial u}{\partial \nu_x}(x) &= f(p) \text{ for } p \in \partial\Omega \end{aligned} \tag{3.13}$$

is solvable, provided that

$$\int_{\partial\Omega} f(p) dS(p) = 0. \tag{3.14}$$

It admits a unique solution under the additional assumption that u goes to 0 at infinity. The solution can be represented in the form of a single layer potential arising from a charge distribution σ on $\partial\Omega$:

$$u(x) = \frac{1}{2\pi} \int_{\partial\Omega} \sigma(y) \log|x - y| dS(y). \tag{3.15}$$

The proper charge distribution σ is obtained by solving the boundary integral equation

$$\frac{1}{2}\sigma(p) + \frac{1}{2\pi} \int_{\partial\Omega} \sigma(y) \frac{\partial}{\partial \nu_p} \log|p - y| dS(y) = f(p), \tag{3.16}$$

which is derived by taking the derivative in the variable x of both sides of (3.15) with respect to the outward normal vector, taking the limit as x goes to $p \in \partial\Omega$ from the exterior of Ω, and applying the appropriate jump relation. As in the case of the interior Dirichlet problem, the integral equation (3.16) is uniquely solvable.

3.4 Interior Neumann problem

Similarly, the *interior Neumann problem*

$$\begin{aligned} \Delta u(x) &= 0 \text{ for } x \in \Omega, \\ \lim_{\substack{x \to p \\ x \in \Omega}} \frac{\partial u}{\partial \nu_x}(x) &= f(p) \text{ for } p \in \partial\Omega \end{aligned} \tag{3.17}$$

is uniquely solvable (up to a constant), provided

$$\int_{\partial\Omega} f(p) dS(p) = 0. \tag{3.18}$$

Representing the solution u as a single layer potential

$$u(x) = \frac{1}{2\pi} \int_{\partial\Omega} \sigma(y) \log|x - y| dS(y) \tag{3.19}$$

leads to the integral equation

$$-\frac{1}{2}\sigma(p) + \frac{1}{2\pi} \int_{\partial\Omega} \sigma(y) \frac{\partial}{\partial \nu_p} \log|p - y| dS(y) = f(p). \tag{3.20}$$

As in the case of the exterior Dirichlet problem, the operator appearing on the left-hand side of equation (3.20) has a one-dimensional null space. To overcome this, we consider instead the integral equation

$$-\frac{1}{2}\sigma(p) + \frac{1}{2\pi} \int_{\partial\Omega} \sigma(y) \left(\frac{\partial}{\partial \nu_p} \log|p - y| + 1 \right) dS(y) = f(p), \text{ for } p \in \partial\Omega, \tag{3.21}$$

which has a unique solution σ and, when inserted into (3.19), produces a solution of (3.17).

4 Numerical apparatus

In this section, we describe a scheme for constructing a factorization of the inverse of any matrix that possesses a hierarchical structure involving low-rank off-diagonal blocks. We first present a one-level scheme in Section 4.1, and then we discuss how it can be applied recursively to obtain a multi-level scheme in Section 4.2. In particular, the scheme we describe is applicable to matrices resulting from the discretization of boundary integral equations of potential theory.

4.1 One-level compression scheme

Consider a matrix A written in 2×2 block form:

$$A = \begin{bmatrix} D_1 & O_1 \\ O_2 & D_2 \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}, \tag{4.1}$$

where the off-diagonal blocks $O_1 \in \mathbb{R}^{n\times m}$ and $O_2 \in \mathbb{R}^{m\times n}$ are of numerical rank $k < \min(m,n)$ (to precision ε), and the diagonal blocks $D_1 \in \mathbb{R}^{n\times n}$ and $D_2 \in \mathbb{R}^{m\times m}$ are invertible. A compressed factorization of A^{-1} can be obtained by transforming A into a simpler form, and then applying the Sherman-Morrison-Woodbury formula.

First, we construct for O_1 and O_2 factorizations of the form (2.13):

$$O_1 = Q_1 S_1 + O(\varepsilon) \tag{4.2}$$

$$O_2 = Q_2 S_2 + O(\varepsilon), \tag{4.3}$$

where $Q_1 \in \mathbb{R}^{n\times k}$, $Q_2 \in \mathbb{R}^{m\times k}$, $S_1 \in \mathbb{R}^{k\times m}$, $S_2 \in \mathbb{R}^{k\times n}$, and Q_1, Q_2 have orthonormal columns. For simplicity, we will henceforth assume that the off-diagonal blocks have *exact* rank k and ignore the error terms.

Now if one applies the matrix

$$B = \begin{bmatrix} D_1^{-1} & 0 \\ 0 & D_2^{-1} \end{bmatrix} \tag{4.4}$$

to A from the left, one obtains a matrix of the form

$$\tilde{A} = \begin{bmatrix} I & U_1 S_1 \\ U_2 S_2 & I \end{bmatrix}, \tag{4.5}$$

where $U_1 = D_1^{-1} Q_1$ and $U_2 = D_2^{-1} Q_2$. Expressing $\tilde{A}$ as

$$\tilde{A} = I + UV^*, \tag{4.6}$$

where

$$U = \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix} \tag{4.7}$$

and

$$V^* = \begin{bmatrix} 0 & S_1 \\ S_2 & 0 \end{bmatrix}, \tag{4.8}$$

we now use formula (2.22) to represent the inverse of $\tilde{A}$ as:

$$\tilde{A}^{-1} = I - UCV^*, \tag{4.9}$$

where

$$C = (I + V^* U)^{-1}. \tag{4.10}$$

Thus, we have obtained a factorization of the inverse of A in the form:

$$A^{-1} = (I - UCV^*)B. \tag{4.11}$$

Note that U is $(n+m) \times 2k$, V is $2k \times (n+m)$, and C is $2k \times 2k$. As long as k is small, the factorizations (4.2), (4.3), and the C matrix (4.10) can be computed rapidly, and the inverse of A can be applied rapidly to any vector or matrix by a scheme based on (4.11).

Remark 4.1 *In the above we assume that O_1 and O_2 have the same rank. This assumption was made for notational convenience and is in no way essential to the results.*

4.2 Multi-level compression scheme

The one-level compression scheme in Section 4.1 can be applied recursively to obtain a multi-level compression scheme. In this subsection, we construct a two-level scheme; a formal description of the multi-level scheme can be found in Section 5.

Consider a matrix A with a two-level block structure:

$$A = \begin{bmatrix} \begin{bmatrix} D_{2,1} & O_{2,1} \\ O_{2,2} & D_{2,2} \end{bmatrix} & O_{1,1} \\ O_{1,2} & \begin{bmatrix} D_{2,3} & O_{2,3} \\ O_{2,4} & D_{2,4} \end{bmatrix} \end{bmatrix}, \tag{4.12}$$

with the two diagonal blocks on the first level defined by the formula

$$D_{1,1} = \begin{bmatrix} D_{2,1} & O_{2,1} \\ O_{2,2} & D_{2,2} \end{bmatrix}, \tag{4.13}$$

$$D_{1,2} = \begin{bmatrix} D_{2,3} & O_{2,3} \\ O_{2,4} & D_{2,4} \end{bmatrix}. \tag{4.14}$$

We assume that all diagonal blocks $D_{i,j}$, $i = 1, 2$; $j = 1, 2, \ldots, 2i$, are invertible square matrices, and all off-diagonal blocks $O_{i,j}$, $i = 1, 2$; $j = 1, 2, \ldots, 2i$, have numerical ranks at most k. For convenience, let us denote the matrix $(I - UCV^*)$ in (4.11) by X, so the one-level compression formula (4.11) can be rewritten as

$$A^{-1} = XB. \tag{4.15}$$

The two-level compression scheme is carried out in a bottom-up manner. First, we apply the one-level compression scheme in Section 4.1 to the diagonal blocks $D_{1,1}$ and $D_{1,2}$, obtaining factorizations of their inverses:

$$D_{1,1}^{-1} = X_{1,1}B_{1,1} \tag{4.16}$$

$$D_{1,2}^{-1} = X_{1,2}B_{1,2}. \tag{4.17}$$

That is, $X_{1,1}, B_{1,1}$ are obtained by replacing the role of A in (4.1) by that of $D_{1,1}$ in (4.13). $X_{1,2}, B_{1,2}$ are similarly obtained.

Then, we apply the one-level compression scheme to the first-level block structure of A:

$$A = \begin{bmatrix} D_{1,1} & O_{1,1} \\ O_{1,2} & D_{1,2} \end{bmatrix}, \tag{4.18}$$

making use of the fact that compressed factorizations of the inverses of $D_{1,1}$ and $D_{1,2}$ are already stored. We thus obtain a compressed factorization of A^{-1} as

$$A^{-1} = X_2 B_2, \tag{4.19}$$

where B_2 has the block form

$$B_2 = \begin{bmatrix} X_{1,1}B_{1,1} & 0 \\ 0 & X_{1,2}B_{1,2} \end{bmatrix}. \tag{4.20}$$

As long as k is small, the compressed factorizations of $D_{1,1}^{-1}$, $D_{1,2}^{-1}$, and A^{-1} can be computed rapidly. After that, we can apply A^{-1} to any vector by a recursive scheme based on equations (4.19) and (4.20). Note that the recursive scheme constructs an one-sided factorization of A^{-1} of the form

$$A^{-1} = X_2 X_1 B_1, \tag{4.21}$$

where

$$X_1 = \begin{bmatrix} X_{1,1} & 0 \\ 0 & X_{1,2} \end{bmatrix} \text{ and } B_1 = \begin{bmatrix} B_{1,1} & 0 \\ 0 & B_{1,2} \end{bmatrix}. \tag{4.22}$$

A multi-level compression scheme can be similarly obtained by applying the one-level compression scheme recursively. If A has an n-level block structure, then the scheme constructs a one-sided factorization of A^{-1} of the form

$$A^{-1} = X_n X_{n-1} \dots X_1 B_1, \tag{4.23}$$

where $X_1, \dots, X_{n-1}$ and B_1 are block-diagonal matrices. This is in contrast to the scheme in [19], which constructs instead a two-sided factorization of the inverse of a matrix. Section 5 describes the numerical algorithm in detail.

Remark 4.2 *We only need to compute and store the matrices that are needed in applying the inverse of A to another vector or matrix. For example, in the two-level scheme above, the matrices $B_{1,1}$, $B_{1,2}$, B_2, $X_{1,1}$, $X_{1,2}$, and X_2 need not be explicitly formed.*

Remark 4.3 *The multi-level compression procedure is particularly applicable to matrices that arise from the discretization of boundary integral operators of potential theory. Suppose we have the boundary integral operator*

$$\lambda\sigma(x) + \int_\Gamma K(x,y)\sigma(y)dS(y), \text{ for } x \in \Gamma, \tag{4.24}$$

where Γ denotes the boundary of a region Ω, and $K(x,y)$ is the kernel of the single or double layer potential for Laplace's equation. Let $A \in \mathbb{R}^{n\times n}$ be a matrix obtained from discretizing (4.24) using n nodes on Γ, and let Γ_0 be a segment of Γ that is discretized with n_0 nodes. Then for most regions Ω encountered in practice, the rank of interaction k between the nodes on Γ_0 and those on the rest of the boundary is bounded by the logarithm of n_0. Specifically,

$$k \le c\log(n_0), \tag{4.25}$$

where c is a constant independent of n_0 and n (see, for example, [19]). The bound (4.25) implies that in such case A admits a hierarchical block structure in which the off-diagonal blocks are of low numerical rank.

Remark 4.4 *Once a multi-level compression procedure is performed on A, one can rapidly apply A^{-1} to any vector in a recursive manner. Typically, if A is an $n \times n$ matrix arising from the discretization of a boundary integral operator, it takes $O(n \log n)$ operations to apply its inverse to a vector.*

5 Algorithm and performance

Section 4 contains an informal description of a hierarchical compression scheme for matrices with low-rank off-diagonal blocks. In this section, we give an algorithmic description of such a scheme, and estimate its efficiency.

5.1 One-level compression

In the following, we consider a matrix A in the block form (4.1) satisfying the assumptions of Section 4.1.

- Step (1) Construct for O_1 and O_2 factorizations of the form (2.13):

$$O_1 \approx Q_1 S_1 \tag{5.1}$$

$$O_2 \approx Q_2 S_2. \tag{5.2}$$

- Step (2) Compute and store D_1^{-1}, D_2^{-1}.
- Step (3) Apply D_1^{-1} and D_2^{-1} to Q_1 and Q_2 respectively, obtaining $U_1 = D_1^{-1}Q_1$ and $U_2 = D_2^{-1}Q_2$.
- Step (4) Compute $C = (I + V^*U)^{-1}$, where U is as in (4.7) and V^* is as in (4.8).
- Step (5) Store U_1, U_2, S_1, S_2, and C.

Using the randomized procedure described in Section 2.3, computing column skeletons for O_1 and O_2 requires $O(mn \log l + lnk)$ floating point operations, where l is chosen to be $k + 20$. After that, the factorizations (5.1) and (5.2) can be computed in $O(k^2(n + m))$ operations (see Remark 2.1). Thus, Step (1) requires $O(mn \log l + lnk + k^2(n + m))$ operations. Step (2) requires $O(n^3 + m^3)$ operations. Step (3) requires $O(nmk)$ operations. In Step (4), forming $(I + V^*U)$ requires $O(k^2(n+m))$ operations, while inverting it requires an additional $O(k^3)$ operations. Since $k < \min(m, n)$, the total cost is

$$T \sim mn \log l + lnk + k^2(m + n) + mnk + n^3 + m^3. \tag{5.3}$$

5.2 Multi-level compression

In this subsection we give a detailed description of the recursive, multi-level compression algorithm. We suppose that the input matrix A is represented via a multi-level block structure consisting of n levels. In particular, there are 2^r diagonal and 2^r off-diagonal blocks belonging to level r, for $r = 1, \ldots, n$. For example, if A has a two-level block structure represented by (4.12), (4.13), and (4.14), then the diagonal blocks of A belonging to level $r = 1$ are $D_{1,1}$ and $D_{1,2}$, and the off-diagonal blocks belonging to level $r = 2$ are $O_{2,1}, O_{2,2}, O_{2,3}$, and $O_{2,4}$. We assume that the diagonal blocks on all levels are invertible.

For convenience, in the algorithm below we will adopt the following notation. Let B denote the input matrix A itself or a diagonal block on level r, where $r = 1, \ldots, n - 1$. Then the hierarchical block structure of A imposes on B the following 2×2 block structure:

$$B = \begin{bmatrix} D_1 & O_1 \\ O_2 & D_2 \end{bmatrix}. \tag{5.4}$$

We define D_1 and D_2 to be the left and right children of B, B to be the parent of D_1 and D_2, and D_1 (D_2) to be the left (right) sibling of D_2 (D_1).

The following gives a description of the algorithm.

1. Computation of all column skeletons

For each level $r = 1, \ldots, n$, apply the pivoted Gram-Schmidt procedure to all off-diagonal blocks, storing a column skeleton for each off-diagonal block in terms of column indices.

2. Compression step

- Step (0) Initialize the current block B to be the input matrix A. Set level = 0. Go to Step (1).

- Step (1)

  (i) If this is the first pass of the current block B to Step (1), go to Step (2).

  (ii) If this is the second pass of B to Step (1), go to Step (3).

- Step (2)

  (i) If level $= n-1$, invert the left and right children D_1 and D_2 of B. Store D_1^{-1} and D_2^{-1}. Go back to Step (1).

  (ii) If level $< n-1$, update the current block B to be its left-child. Update level := level + 1. Go back to Step (1).

- Step (3) [Comment: See representation (5.4) of the current block B.]

  (i) Construct for O_1 and O_2 factorizations of the form (2.13):

$$O_1 \approx Q_1 S_1 \tag{5.5}$$

$$O_2 \approx Q_2 S_2. \tag{5.6}$$

  (ii) Apply the inverses of the diagonal subblocks D_1, D_2 of B to Q_1, Q_2 respectively, obtaining $U_1 = D_1^{-1} Q_1$ and $U_2 = D_2^{-1} Q_2$.

  (iii) Compute the matrix $C = (I + V^* U)^{-1}$, where U is as in (4.7), and V^* is as in (4.8).

  (iv) Store U_1, U_2, S_1, S_2, and C.

  (v) If level $= 0$, we are done. Otherwise, go to Step (4).

- Step (4)

  (i) If B is a left-child of its parent, update B to be its right-sibling. Go back to Step (1).

  (ii) If B is a right-child of its parent, update B to be its parent. Update level := level $-$ 1. Go back to Step (1).

Remark 5.1 *In effect, for each diagonal block B of A on level r, where $r = 1, \ldots, n-1$, the algorithm computes and stores the quantities pertaining to a one-level compression of B (see Step (3)). The algorithm does it in a recursive manner such that at the time the inverse of a diagonal block needs to be applied (see Step (3)(ii)), the relevant quantities needed for its rapid application have already been constructed.*

Remark 5.2 *Only the inverses of the lowest-level diagonal blocks are explicitly computed and stored by the algorithm (see Step (2)(i)). The end result is a hierarchical list of quantities that allows the rapid application of the inverse of A in a recursive manner.*

5.2.1 Computational cost

The remainder of this subsection assesses the efficiency of the multi-level compression algorithm. Let N be the size of the matrix A, and $r = 1, \ldots, R$ be the index for the levels, so that the numbers of diagonal and off-diagonal blocks that belong to level r are both equal to $p_r = 2^r$. We let n_r denote the average block size (for both diagonal and off-diagonal blocks) on level r, so that

$$n_r = \frac{n_1}{2^{r-1}}, \tag{5.7}$$

and let k_r denote the average rank of an off-diagonal block on level r.

1. Computation of all column skeletons

First, we estimate the cost of computing column skeletons for all off-diagonal blocks. For each off-diagonal block on level r, where $r = 1, \ldots, R$, we apply the randomized algorithm described in Section 2.3 to compute its column skeleton. Each block involves a cost of

$$t \sim n_r^2 \log l_r + n_r l_r k_r, \tag{5.8}$$

where we choose $l_r = k_r + 20$. For matrices with rank-deficient off-diagonal blocks, we can assume that

$$k_r \leq c\sqrt{n_r}, \tag{5.9}$$
$$\log l_r \leq c', \tag{5.10}$$

where c and c' are constants independent of n_r and N. So t in (5.8) is dominated by

$$t \sim n_r^2. \tag{5.11}$$

The cost for all off-diagonal blocks on level r is then

$$t_r \sim p_r n_r^2 \sim N n_r, \tag{5.12}$$

where we have used the fact that $p_r n_r = N$. The total cost for all R levels is thus

$$T_1 \sim \sum_{r=1}^{R} t_r \sim N \sum_{r=1}^{R} n_r \sim N \sum_{r=1}^{R} \frac{N}{2^r}. \tag{5.13}$$

So we obtain

$$T_1 \sim N^2. \tag{5.14}$$

2. Compression step

We now estimate the cost of the main part of the compression algorithm. The cost of applying a one-level compression to a diagonal block on level $R-1$ follows the analysis of Section 5.1. The only difference is that here a column skeleton is already computed, so Step (1) requires only $O(k_R^2 n_R)$ operations. The cost for compressing one diagonal block on level $R-1$ is thus

$$t \sim k_R^2 n_R + n_R^2 k_R + n_R^3 \sim n_R^3, \tag{5.15}$$

and the total cost for the level is

$$t_{R-1} \sim p_{R-1} n_R^3 \sim p_R n_R^3 \sim N n_R^2, \tag{5.16}$$

where we have used the fact that $p_R = 2p_{R-1}$ and $p_R n_R = N$.

The cost of compressing each diagonal block on level r, where $r = 1, \ldots, R-2$, comes entirely from Step (3) of the algorithm. Step 3(i) takes $O(k_{r+1}^2 n_{r+1})$ operations (see Remark 2.1), Step 3(ii) takes $O(k_{r+1} n_{r+1} \log(n_{r+1}))$ operations (see Remark 4.4), while Step 3(iii) takes $O(k_{r+1}^2 n_{r+1} + k_{r+1}^3)$ operations (see Section 5.1). Thus, the cost for each block is

$$t' \sim k_{r+1}^2 n_{r+1} + k_{r+1} n_{r+1} \log(n_{r+1}), \tag{5.17}$$

and the compression cost for level r is

$$\begin{aligned} t_r &= p_r t' \\ &\sim p_r(k_{r+1}^2 n_{r+1} + k_{r+1} n_{r+1} \log(n_{r+1})) \\ &\sim p_{r+1}(k_{r+1}^2 n_{r+1} + k_{r+1} n_{r+1} \log(n_{r+1})) \\ &\sim N(k_{r+1}^2 + k_{r+1} \log(n_{r+1})). \end{aligned} \tag{5.18}$$

Combining the estimates (5.18) for $r = 1, \ldots, R-2$ with the estimate (5.16), we obtain the total cost for the compression part of the algorithm:

$$T_2 \sim N \sum_{r=1}^{R-2} (k_{r+1}^2 + k_{r+1} \log(n_{r+1})) + N n_R^2. \tag{5.19}$$

In practice, we choose the number of levels R to be of the order $\log N$, so that n_R will be a small fixed number.

Combining (5.7) with Remark 4.3, we obtain the estimate

$$k_r \leq c'' \log(n_1) \gamma^{r-1}, \tag{5.20}$$

where c'' and γ are constants independent of r, n_1, and N. In particular, the estimate (5.20) is valid for some $\gamma < 0.95$.

Now, combining (5.20) with (5.19), we have

$$N \sum_{r=1}^{R-2} (k_{r+1}^2 + k_{r+1} \log(n_{r+1})) \sim N \sum_{r=1}^{R-2} \left(\log^2(n_1)(\gamma^2)^r + \log n_1 \gamma^r \log(n_{r+1}) \right) \tag{5.21}$$

$$\sim N \log^2 N \sum_{r=1}^{R-2} (\gamma^2)^r \tag{5.22}$$

$$\sim c''' N \log^2 N, \tag{5.23}$$

where $c''' = (1 - \gamma^2)^{-1}$. Thus T_2 has an asymptotic complexity of

$$T_2 \sim N \log^2 N. \tag{5.24}$$

Combining (5.14) and (5.24), we have

$$T \sim N^2, \tag{5.25}$$

where T denotes the cost of the entire algorithm.

6 Application of the solver to boundary integral equations

In Section 5, we presented a generic algorithm that depends only on the ranks of off-diagonal blocks of the matrix to be inverted. When applied to a dense $n \times n$ matrix A with rank-deficient off-diagonal blocks, the algorithm typically requires $O(n^2)$ operations. In the case that A is a discretization of a boundary integral operator, we can accelerate the algorithm by utilizing the geometry of the region on which the equation is to be solved, reducing the total cost to $O(n \log^2 n)$. Section 6.1 introduces a technique for computing column skeletons for off-diagonal blocks that is faster than the generic technique in Section 5, and Section 6.2 describes an adaptive rotation scheme that seeks to minimize ranks of off-diagonal blocks by rearranging the parameterization of the boundary.

6.1 An accelerated procedure for computing column skeletons

The bulk of the computational cost of the algorithm presented in Section 5.2 lies in the computation of column skeletons for the factorization of off-diagonal blocks. When the matrix under consideration is a discretization of a boundary integral operator, we can exploit the geometry of the underlying contour and compute column skeletons for the off-diagonal blocks in a hierarchical manner, reducing computational complexity of the process. In this subsection, we describe a single-block compression technique, whose details can be found in [19], and then utilize it in a hierarchical scheme for the construction of column skeletons of all off-diagonal blocks.

Consider a smooth contour $\Gamma = \Gamma_1 \cup \Gamma_2$ as in Fig. 1 (a), and let the matrix A in the block form

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \tag{6.1}$$

be the discretization of the boundary integral operator

$$\lambda\sigma(x) + \int_{\Gamma} K(x,y)\sigma(y)dS(y), \ x \in \Gamma, \tag{6.2}$$

so that, for example, $A_{1,2}$ represents the potential on Γ_1 generated by a charge distribution on Γ_2, and K is the kernel of a single or double layer potential for Laplace's equation.

Let Γ_{circ} be a circular contour surrounding Γ_2, and let Γ_{ext} be the part of Γ_1 outside of Γ_{circ} (see Fig. 1 (b)). For any $x \in \Gamma_{\text{ext}}$ and $y \in \Gamma_2$,

$$K(x,y) = \int_{\Gamma_{\text{circ}}} G(x,z)K(z,y)dS(z), \tag{6.3}$$

where G is the Green's function of Laplace's equation on the contour Γ_{circ}. By virtue of (6.3), we can interpolate the values of the potential field generated by Γ_2 on Γ_{ext} from the values of the field generated by Γ_2 on Γ_{circ}. This observation allows us to compute a column skeleton for the block $A_{1,2}$ via an entirely local operation: instead of compressing the interaction between Γ_2 and Γ_1, it suffices to compress the interaction between Γ_2 and $\hat{\Gamma}$, where $\hat{\Gamma}$ is the contour formed by the union of Γ_{circ} and the part of Γ_1 that is inside Γ_{circ}. If Γ_2 is discretized using n nodes, then typically $\hat{\Gamma}$ can be discretized using $O(n)$ nodes. A column skeleton for $A_{1,2}$ can thus be computed in $O(n^2k)$ operations, where k is the numerical rank of $A_{1,2}$. A more detailed discussion can be found in [19].

Remark 6.1 *The method described above can be applied to any partial differential equation for which Green's identities hold, and the Green's function does not have to be known explicitly. In particular, the method also works for the solution of boundary integral equations associated with the Helmholtz equation and Maxwell's equations.*

The remainder of this subsection describes, via an example, the recursive application of the single-block compression technique for the determination of the column skeletons of all off-diagonal blocks. Let A be a discretization of the integral operator (6.2) that is represented via a two-level block structure (4.12). Each off-diagonal block in (4.12) corresponds to charges on a segment of Γ, and these segments form a two-level partitioning of Γ. Suppose for example that the blocks $O_{2,3}$, $O_{2,4}$, $O_{1,1}$, and $O_{1,2}$ in (4.12) correspond to charges on the segments $\Gamma_{2,3}$, $\Gamma_{2,4}$, $\Gamma_{1,1}$, and $\Gamma_{1,2}$ respectively such that, in particular, $\Gamma_{1,1} = \Gamma_{2,3} \cup \Gamma_{2,4}$. Using the single-block technique described above, we compress the interaction of $\Gamma_{2,3}$ with the rest of the contour, obtaining a column skeleton for the block $O_{2,3}$ in terms of column indices $J_1 = [i_1, \ldots, i_k]$. Similarly, we compute a set of column indices $J_2 = [j_1, \ldots, j_l]$ for $O_{2,4}$.

Since $\Gamma_{2,3}$ and $\Gamma_{2,4}$ partition $\Gamma_{1,1}$, we can obtain a column skeleton for the block $O_{1,1}$ by downsampling from the column skeletons already computed for $O_{2,3}$ and $O_{2,4}$. In other words, J_1 and J_2 correspond

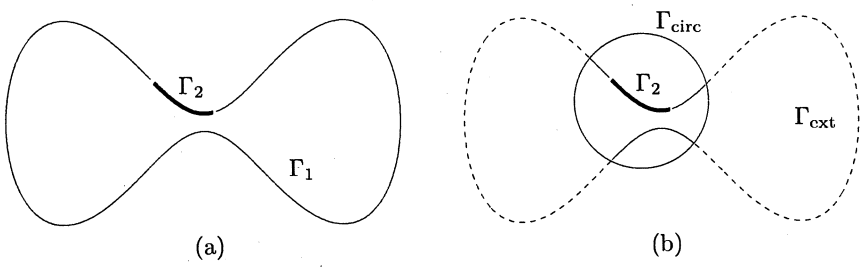

Figure 1: A contour Γ. In Fig. (a), the partitioning $\Gamma = \Gamma_1 \cup \Gamma_2$ is shown with Γ_2 drawn with a bold line. In Fig. (b), Γ_{circ} is the circular contour and Γ_{ext} is the part drawn with a dashed line.

to two clusters of charges on $\Gamma_{1,1}$, and the union of these charges belongs to a (possibly discontinuous) segment $\Gamma_{\text{pre}} \subseteq \Gamma_{1,1}$ so that, to compute a column skeleton for $O_{1,1}$, it suffices to compress the interaction between Γ_{pre} and $\Gamma_{1,2}$.

The same idea can be applied hierarchically when A has a multi-level block structure: due to the underlying geometry, the column skeletons computed for off-diagonal blocks at one level can be combined to form pre-computed column skeletons for blocks on the upper level, thus reducing computational cost to $O(N \log N)$, where N is the total number of discretization nodes (see Section 6.1.1 below).

6.1.1 Computational cost

In this subsection, we estimate the computational cost of the multi-level compression algorithm described in Section 5.2 in the context of boundary integral equations, in which the accelerated technique in Section 6.1 can be used.

We start with observing that the technique only accelerates the procedure of computing column skeletons for off-diagonal blocks, so the estimate (5.24) for the cost T_2 of the main compression part remains the same. In particular, we will show that the cost T_1 of computing all column skeletons is now $O(N \log N)$.

We will follow the notation in Section 5.2. In addition, let k'_r denote the average rank of interaction between a cluster on level r with the rest of the boundary. For all practical purposes, we have

$$k_r \leq k'_r \leq ck_r, \tag{6.4}$$

where c is a constant independent of r and N. So, for simplicity we will use k_r in the following calculation.

We first apply the pivoted Gram-Schmidt algorithm to compress the interaction of each cluster on level R with the rest of the world. By the technique of Section 6.1, we are on average compressing blocks of dimension n_R by n_R. Combining it with the randomized algorithm of Section 2.3 and following the analysis in Section 5.2.1, the total cost for level R is

$$t_R \sim p_R(n_R^2 \log l_R + n_R l_R k_R) \sim N n_R, \tag{6.5}$$

where we choose $l_R = k_R + 20$.

On levels $r = 1, \ldots, R-1$, we downsample from the column skeletons computed on the previous level, so for each cluster on level r we only need to apply the pivoted Gram-Schmidt procedure to a

block of $2k_{r+1}$ columns and n_r rows. Combining it with the randomized algorithm of Section 2.3,

$$\begin{aligned} t_r &\sim p_r((2k_{r+1})n_r \log l_r + (2k_{r+1})l_r k_r) \\ &\leq p_r(2k_r n_r \log l_r + 2k_r^2 l_r) \\ &\sim p_r k_r n_r \\ &\sim N k_r, \end{aligned} \tag{6.6}$$

where t_r is the total cost for level r, and $l_r = k_r + 20$; above we have used the assumption $k_{r+1} \leq k_r$ and the bounds (5.9) and (5.10).

Combining the estimates (6.6) for $r = 1, \ldots, R-1$ with the estimate (6.5), the same analysis as in Section 5.2.1 shows that

$$T_1 \sim N \log N, \tag{6.7}$$

where T_1 is the cost of computing column skeletons for all off-diagonal blocks. Combining (6.7) with (5.24), we have

$$T \sim N \log^2 N, \tag{6.8}$$

where T denotes the cost of the entire algorithm.

6.2 Adaptive rotation scheme

Typically, matrices that arise from the discretization of boundary integral equations of potential theory have rank-deficient off-diagonal blocks. This is the case when the corresponding segments of the boundary are "well-separated" so that interacting clusters of nodes are sufficiently far from each other. When the boundary is specified via a parameterization such that the separation between two points is not well predicted by their corresponding distance in the parameter space, there may exist interacting segments that are close to each other, leading to off-diagonal blocks that have high ranks. The dumbbell-shaped contour Γ in Fig. 2 is an example that needs to be treated with care in order to avoid such a problem. In this subsection, we introduce a scheme that adaptively arranges the partitioning of a given contour

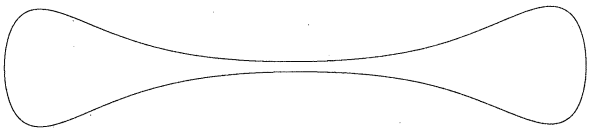

Figure 2: The dumbbell-shaped contour Γ.

so that clusters of nodes interacting with each other are well-separated in the plane.

Consider the contour Γ in Fig. 2 and the matrix A formed by discretizing the boundary integral operator (6.2) according to some given parameterization of Γ. The scheme proceeds in a hierarchical manner. On the coarsest (first) level, a "center" of Γ is chosen (the center of mass is an acceptable choice). Then, an angle θ is chosen randomly and Γ is partitioned into two segments by a line with inclination θ passing through the chosen center (see Fig. 3 (a)). This divides the discretization nodes on Γ into two clusters, and the rank of interaction k between the clusters is estimated. If the computed k is less than a pre-set threshold, the partition is accepted; otherwise, another angle θ is chosen and the above process is repeated.

Once a partition $\Gamma = \Gamma_1 \cup \Gamma_2$ is found on the coarsest level, the same procedure is applied to Γ_1 and Γ_2 respectively to obtain partitions for the second level. We then recursively apply the procedure to obtain "optimal" partitions of the contour for all levels. Finally, we rearrange the rows and columns of

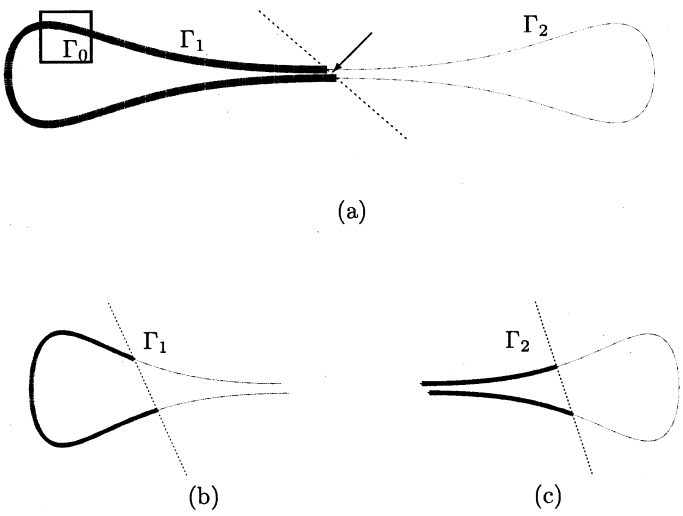

Figure 3: Fig. (a) shows, on the first level, the partition $\Gamma = \Gamma_1 \cup \Gamma_2$ obtained by the adaptive rotation scheme for the contour Γ in Fig. 2, indicated by a dashed line. The chosen center of Γ is indicated with an arrow. On the second level, partitions obtained for the segments Γ_1 and Γ_2 are shown in Fig. (b) and (c) respectively. In general, on each level a contour is split by lines into segments so that interacting clusters are well-separated from each other, resulting in rank-deficient off-diagonal blocks in the discretized operator.

the matrix A according to the partitioning. Fig. 3 shows, on the first two levels, the partitions that are typically obtained by the scheme for Γ.

The only expensive part of the scheme lies in computing the interaction rank between two clusters, and if it is done via the randomized algorithm of Section 2.3 the scheme involves a cost of $O(n^2)$, where n is the number of nodes on the contour. However, the same principle based on Green's theorem described in Section 6.1 can be used to speed up the procedure. Suppose, for example, we wish to compute the rank of an $m \times k$ matrix B corresponding to the potential generated by Γ_1 on Γ_2, as arranged in Fig. 3(a). Consider a segment Γ_0 that belongs to Γ_1 and is inside of a box, as indicated in the figure. Since Γ_0 is sufficiently separated from the dashed line, we can, by virtue of Green's theorem, replace the charges on Γ_0 by some fixed small number of artificial charges placed on the boundary of the box, and compress the interaction between the charges on the box and the nodes on Γ_2. This can be done systematically on Γ_1 by applying the method to those boxes in the structure that are well-separated from the line. As a result, we obtain an $m \times l$ matrix C whose column dimension l is typically much less than that of B, and whose rank gives a good approximation to the rank of B, provided that the artificial charges on the boxes are suitably chosen (see, for example, [19]). This approach reduces the column dimension of a block whose rank has to be computed. In the experimental results in Section 7, we will see that for typical contours the scheme involves a cost of $O(n)$, where n is the number of nodes on the contour, and the time taken by the scheme never constitutes more than 8 percent of the total solution time.

7 Numerical results

In this section, we present the results of a number of numerical experiments performed to assess the efficiency of the schemes described in Sections 5 and 6.

In each of the experiments, we apply Nyström discretization to one of the following boundary integral

equations:

$$\frac{1}{2}\sigma(p) + \frac{1}{2\pi}\int_\Gamma \sigma(y)\frac{\partial}{\partial \nu_y}\log|p-y|dS(y) = f(p) \quad (7.1)$$

$$-\frac{1}{2}\sigma(p) + \frac{1}{2\pi}\int_\Gamma \sigma(y)\left(\frac{\partial}{\partial \nu_y}\log|p-y|+1\right)dS(y) = f(p) \quad (7.2)$$

$$\frac{1}{2}\sigma(p) + \frac{1}{2\pi}\int_\Gamma \sigma(y)\frac{\partial}{\partial \nu_p}\log|p-y|dS(y) = f(p) \quad (7.3)$$

$$-\frac{1}{2}\sigma(p) + \frac{1}{2\pi}\int_\Gamma \sigma(y)\left(\frac{\partial}{\partial \nu_p}\log|p-y|+1\right)dS(y) = f(p), \quad (7.4)$$

and solve the resulting linear systems.

The kernels in equations (7.1)-(7.4) are smooth over smooth curve segments. Since the contours considered in the first three experiments are smooth, in these cases we discretize the equations using piecewise Gaussian quadrature on an equispaced mesh. On the other hand, the contour in the last experiment contains a corner point, over which the kernel is singular. In this case, special treatment is needed and we discretize the equation near the corner using piecewise Gaussian quadrature on a simply graded mesh, the details of which are described in Section 7.4 below.

We compare three methods for the solution of the resulting linear systems:

- Method 1. Using the multi-level solver combined with the adaptive rotation scheme.
- Method 2. Using the multi-level solver, without using the adaptive rotation scheme.
- Method 3. Using a "brute force" QR solver (with asymptotic complexity $O(n^3)$) to invert the linear system.

In the case of the Dirichlet problem, the right-hand side f is a potential field generated by a collection of randomly placed charges, and in the case of the Neumann problem, f is the normal derivative of the potential field generated by such a collection of charges. In each experiment the layer potential generated by the computed charge distribution σ was evaluated at a collection of 40 randomly placed points. The values obtained were then compared with the exact potential.

The solvers were implemented in Fortran 77 and compiled with the Lahey/Fujitsu Linux64 Fortran Compiler Release 8.10a. All experiments were run on a PC with an Intel Core i7 2.67 GHz processor and 12GB of memory. No attempt was made to parallelize any of the code. The following notation is used when presenting the numerical results:

R	the number of levels in the multi-level solver
N	the number of discretization nodes used
t_{rot}	the CPU time taken by the adaptive rotation scheme in Method 1
$t_{\text{solve},1}$	total CPU time taken in solving for σ by Method 1 (which includes t_{rot})
$t_{\text{solve},2}$	total CPU time taken in solving for σ by Method 2
$t_{\text{solve},3}$	total CPU time taken in solving for σ by Method 3
$E_{\text{rel},1}$	the relative l^2-norm error obtained by Method 1
$E_{\text{rel},2}$	the relative l^2-norm error obtained by Method 2
$E_{\text{rel},3}$	the relative l^2-norm error obtained by Method 3

By the relative l^2-norm error we mean the quantity $||v - v_\varepsilon||_2/||v||_2$, where $\{v^{(j)}\}_{j=1}^{40}$ denotes the exact potential field at the 40 random points and $\{v_\varepsilon^{(j)}\}_{j=1}^{40}$ denotes the potential field given by the computed σ. All timings presented are in seconds of CPU time.

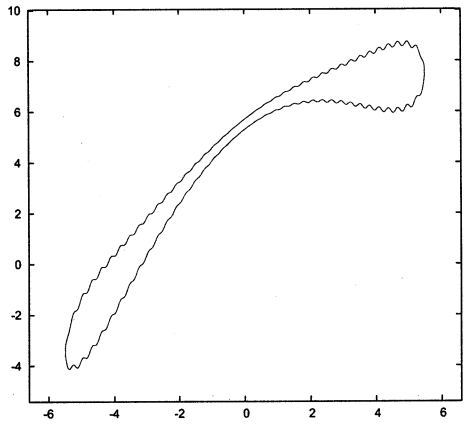

Figure 4: A rippled contour with a thin handle.

7.1 Example: a rippled contour with a thin handle

In this subsection, we present results for the rippled contour with a thin handle shown in Fig. 4. The contour was discretized using between 200 and 51200 nodes and the integral equation (7.2) associated with the exterior Dirichlet problem was solved. Table 1 presents the results.

In this example, we see that, roughly speaking, the timings $t_{\text{solve},1}$ and $t_{\text{solve},2}$ taken by the multi-level solver scale slightly more than linearly with the number of discretization nodes N. This agrees with the estimate (6.8) of its performance in Section 6.1.1. As shown in Fig. 5 (a), the contour is originally parameterized in such a way that its long, narrow "handle" part causes the pair of clusters on the coarsest (first) level to have high rank of interactions. Fig. 5 (b) shows the clusters as rearranged by the adaptive rotation scheme. We observe that the scheme reduces the CPU times roughly by the factor of 3.

Finally, we compare the multi-level solver with the "brute force" QR solver that takes $O(n^3)$ operations. We observe that for small-scale problems, the "brute force" approach is more efficient (as expected). The observed break-even point is about $n = 1100$, after which the solver of this paper is more efficient. At $n = 3200$, the solver of this paper (combined with the rotation scheme) performs about 7.5 times faster than the "brute force" scheme. Needless to say, in practice one would always use the more efficient scheme for the problem to be solved.

7.2 Example: A cross-shaped contour

In this subsection, we present results for the cross-shaped contour shown in Fig. 6. The contour was discretized using between 200 and 51200 nodes and the integral equation (7.4) associated with the interior Neumann problem was solved. Table 2 presents the results.

Similar to the contour in Section 7.1, the one here has several long and narrow parts that introduce high ranks in the off-diagonal blocks of the discretized operator. Fig. 7 shows, in particular, that in the original arrangement of the clusters interactions on the first two levels are of high rank. Fig. 8 shows the clusters as rearranged by the rotation scheme, and the reduced ranks of interactions. We observe that the scheme reduces the CPU times roughly by the factor of 4, and the time it spent constitutes less than 8 percent of the total time.

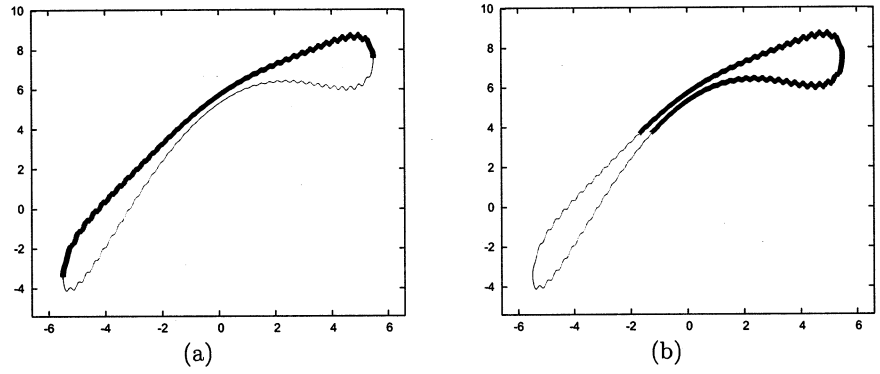

Figure 5: (a) The original arrangement of the clusters of nodes on the first level for the contour shown in Fig. 4. (b) The clusters on the first level as rearranged by the adaptive rotation scheme. The contour was discretized using 3200 nodes, and the ranks of interactions between the pairs of clusters in (a) is about 285 and in (b) is about 50.

Table 1: Computational results for the boundary integral equation (7.2) associated with the exterior Dirichlet problem on the contour shown in Fig. 4.

N	R	t_{rot}	$t_{\text{solvc},1}$	$E_{\text{rcl},1}$	$t_{\text{solvc},2}$	$E_{\text{rcl},2}$	$t_{\text{solvc},3}$	$E_{\text{rcl},3}$
200	4	$2.13e-03$	$6.73e-02$	$4.39e-01$	$1.45e-01$	$3.99e-01$	$5.04e-03$	$3.87e-01$
400	5	$6.92e-03$	$1.61e-01$	$9.03e-03$	$6.71e-01$	$9.37e-02$	$3.89e-02$	$9.94e-03$
800	6	$1.67e-02$	$4.52e-01$	$5.33e-04$	$2.78e+00$	$4.82e-04$	$2.61e-01$	$5.41e-04$
1600	7	$4.91e-02$	$1.26e+00$	$1.26e-06$	$6.90e+00$	$1.27e-06$	$2.58e+00$	$1.02e-06$
3200	8	$9.42e-02$	$2.63e+00$	$3.99e-13$	$1.04e+01$	$5.37e-13$	$1.96e+01$	$5.32e-13$
6400	9	$2.04e-01$	$5.73e+00$	$3.64e-13$	$1.69e+01$	$6.86e-13$	$1.72e+02$	$4.72e-14$
12800	10	$3.86e-01$	$1.20e+01$	$8.04e-13$	$3.42e+01$	$5.05e-13$	$1.35e+03$	$2.61e-14$
25600	11	$7.16e-01$	$2.17e+01$	$1.88e-13$	$6.16e+01$	$6.04e-13$	–	–
51200	12	$1.43e+00$	$4.88e+01$	$4.45e-13$	$1.27e+02$	$3.74e-13$	–	–

Table 2: Computational results for the boundary integral equation (7.4) associated with the interior Neumann problem on the contour shown in Fig. 6.

N	R	t_{rot}	$t_{\text{solve},1}$	$E_{\text{rel},1}$	$t_{\text{solve},2}$	$E_{\text{rel},2}$	$t_{\text{solve},3}$	$E_{\text{rel},3}$
200	4	$2.22e-03$	$7.00e-02$	$1.80e-02$	$1.19e-01$	$1.62e-02$	$4.65e-03$	$1.72e-02$
400	5	$7.30e-03$	$1.92e-01$	$2.69e-04$	$4.75e-01$	$2.61e-04$	$3.35e-02$	$2.66e-04$
800	6	$1.86e-02$	$5.54e-01$	$1.59e-07$	$1.74e+00$	$1.62e-07$	$2.56e-01$	$1.62e-07$
1600	7	$4.99e-02$	$1.34e+00$	$6.32e-12$	$6.03e+00$	$6.40e-12$	$2.31e+00$	$5.87e-12$
3200	8	$1.11e-01$	$3.00e+00$	$1.21e-12$	$1.51e+01$	$8.73e-13$	$1.96e+01$	$1.16e-12$
6400	9	$2.73e-01$	$6.36e+00$	$9.81e-13$	$2.44e+01$	$2.63e-13$	$1.61e+02$	$7.46e-13$
12800	10	$5.63e-01$	$1.25e+01$	$3.25e-13$	$4.61e+01$	$2.16e-13$	$1.38e+03$	$1.63e-13$
25600	11	$2.10e+00$	$2.67e+01$	$5.69e-13$	$9.44e+01$	$9.60e-13$	–	–
51200	12	$3.37e+00$	$6.13e+01$	$2.83e-13$	$1.97e+02$	$3.44e-13$	–	–

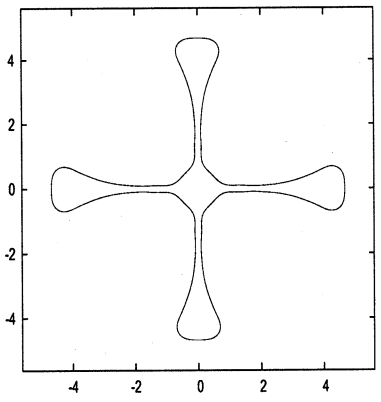

Figure 6: A cross-shaped contour. Its arclength is about 40.

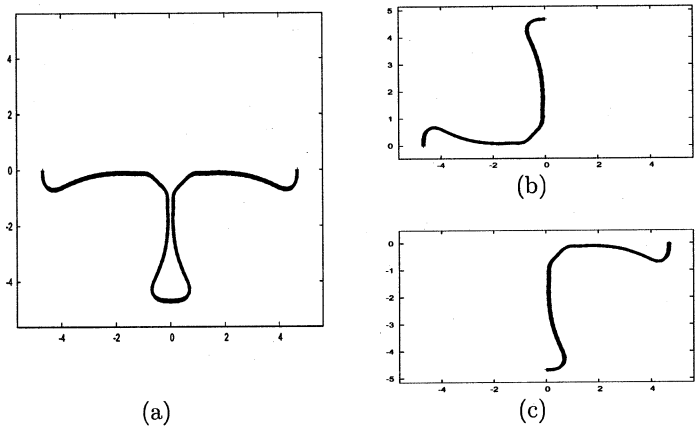

Figure 7: The original arrangement of the clusters of nodes on the contour in Fig. 6 on (a): the first level and (b),(c): the second level. The contour was discretized using 3200 nodes. The rank of interactions in (a) is about 270 and the ranks of interactions in (b) and (c) are both about 150.

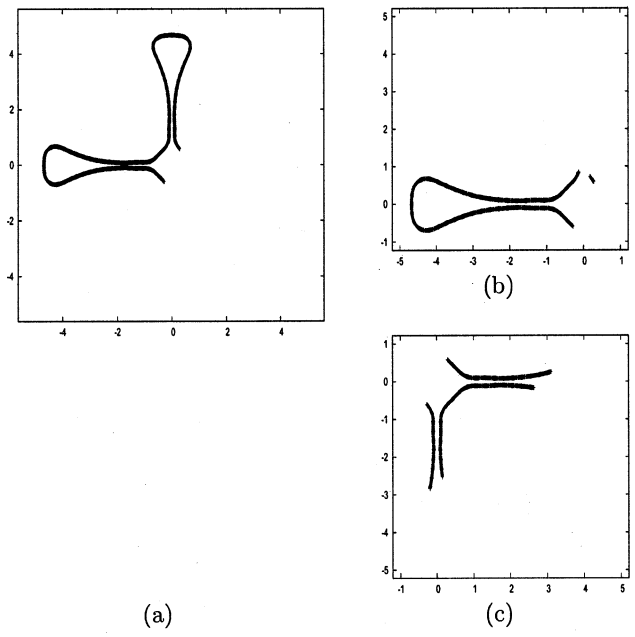

Figure 8: The clusters on the contour in Fig. 6 on (a): the first level and (b),(c): the second level, as rearranged by the adaptive rotation scheme. The rank of interactions in (a) is about 64 and the ranks of interactions in (b) and (c) are both about 45.

7.3 Example: A tank-shaped contour

In this subsection, we present results for the tank-shaped contour in Fig. 9. The contour was discretized using between 200 and 51200 nodes and the integral equation (7.3) associated with the exterior Neumann problem was solved. Table 3 presents the results.

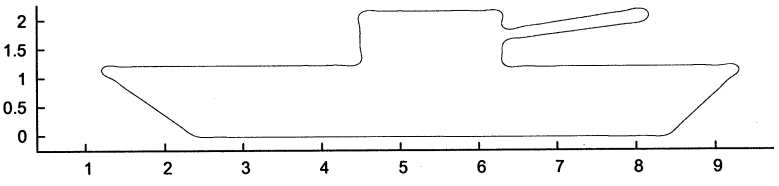

Figure 9: A tank-shaped contour. Its arclength is about 23.

We observe that the asymptotic complexity of the multi-lever solver remains the same as for the contours in Sections 7.1 and 7.2, but the times $t_{\text{solve},2}$ involve smaller constants. Here, Fig. 10 (a) shows the original arrangement of the clusters on the first level and the associated rank of interactions, and Fig. 10 (b) shows the clusters as rearranged by the rotation scheme. In particular, the reduction in rank is not as substantial as those in the previous examples, and the reduction in CPU times is relatively insignificant.

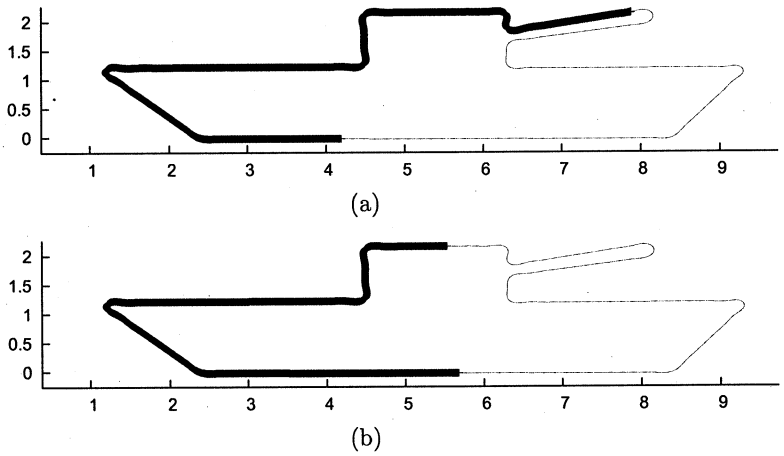

Figure 10: (a) The original arrangement of the clusters of nodes on the first level for the contour in Fig. 9. (b) The clusters as rearranged by the adaptive rotation scheme. Each cluster contains about 1600 nodes, and the ranks of interactions between the pairs of clusters in (a) is about 120 and in (b) is about 40.

7.4 Example: A PacMan-shaped contour

In this subsection, we present results on the PacMan-shaped contour Γ shown in Fig. 11, which contains a single corner γ_0 of exterior angle 0.6 radian. The boundary integral equation (7.1) associated with the interior Dirichlet problem is solved on Γ.

Table 3: Computational results for the boundary integral equation (7.3) associated with the exterior Neumann problem on the contour shown in Fig. 9.

N	R	t_{rot}	$t_{\text{solve},1}$	$E_{\text{rel},1}$	$t_{\text{solve},2}$	$E_{\text{rel},2}$	$t_{\text{solve},3}$	$E_{\text{rel},3}$
200	4	$2.23e-03$	$6.15e-02$	$1.19e-03$	$6.99e-02$	$1.29e-03$	$4.67e-03$	$1.14e-03$
400	5	$7.48e-03$	$1.51e-01$	$1.60e-05$	$2.10e-01$	$1.17e-05$	$3.35e-02$	$1.15e-05$
800	6	$1.92e-02$	$4.26e-01$	$2.28e-11$	$5.70e-01$	$2.61e-11$	$2.56e-01$	$2.40e-11$
1600	7	$5.37e-02$	$9.10e-01$	$1.03e-12$	$1.38e+00$	$1.00e-12$	$2.31e+00$	$1.11e-12$
3200	8	$1.07e-01$	$1.91e+00$	$1.37e-13$	$2.81e+00$	$2.53e-13$	$1.92e+01$	$1.20e-13$
6400	9	$2.63e-01$	$4.07e+00$	$4.23e-13$	$5.67e+00$	$4.09e-13$	$1.51e+02$	$2.98e-13$
12800	10	$6.38e-01$	$8.92e+00$	$1.48e-13$	$1.22e+01$	$2.63e-13$	$1.38e+03$	$1.70e-13$
25600	11	$1.14e+00$	$1.94e+01$	$2.35e-14$	$2.79e+01$	$1.10e-13$	–	–
51200	12	$2.42e+00$	$4.92e+01$	$7.70e-14$	$6.44e+01$	$8.29e-14$	–	–

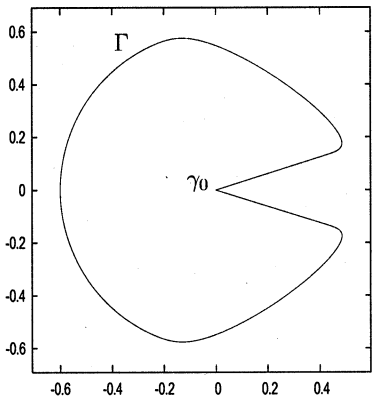

Figure 11: A PacMan-shaped contour Γ with a single corner γ_0 of exterior angle 0.6 radian.

Below, we describe a simple expedient we used to obtain a high-accuracy discretization of a boundary with a corner. The discretization of the boundary integral equation

$$\frac{1}{2}\sigma(p) + \frac{1}{2\pi}\int_{\partial\Omega} \sigma(y)\frac{\partial}{\partial\nu_y}\log|p-y|dS(y) = f(p) \tag{7.5}$$

via the Nyström method with piecewise Gaussian quadrature displays high rates of convergence, so long as the kernel $K(x,y)$ and the layer density σ are smooth. However, if the contour Γ contains corner points, then the kernel $K(x,y)$ is singular, and so is the layer density σ. As a result, piecewise Gaussian quadrature over an equispaced mesh will not be accurate. Standard approaches for discretizing equations of the form (7.5) near a corner point γ_0 amount to using a dense mesh of points near γ_0 (see, for instance, [14, 18, 2]). Following the terminology of [14], we call a subdivision of the interval $[0,1]$ into subintervals with the endpoints

$$\frac{1}{2^j},\ j = 0, 1, 2, \ldots, s, \tag{7.6}$$

a *simply graded mesh.* In this experiment, the integral equation (7.5) is discretized over a small segment containing γ_0 by first mapping $[0,1]$ onto a small interval on each side of γ_0. We then use Gaussian quadrature to discretize the image of each of the subintervals comprising the simply graded mesh on $[0,1]$. Note that the resulting discretization omits a small region around γ_0. The refinement of the simply graded mesh around γ_0 is controlled by adjusting a cut-off ε_{cut} on the minimum length of the subintervals comprising the mesh; in other words, given ε_{cut}, we choose s in (7.6) to be the smallest integer such that

$$\frac{1}{2^s} < \varepsilon_{\text{cut}}. \tag{7.7}$$

The part of Γ away from γ_0 is discretized via an equispaced mesh. The same number of Gaussian nodes is used for each of the subintervals in the mesh discretizing Γ. Fig. 12 depicts the subintervals that comprise the mesh discretizing Γ when ε_{cut} is set at 10^{-8}. In this setting, the simply graded mesh around the corner γ_0 contains about 1150 nodes. The proof of convergence of such a discretization to the solution of the underlying integral equation is somewhat involved, and can be found, for example, in [14].

Table 4 presents the result of the experiment. We observe that Γ is originally parameterized in such a way that discretization nodes to each side of γ_0 belong to separate clusters (see Fig. 13). As we decrease ε_{cut}, the distribution of nodes around γ_0 becomes denser and the rank of interactions r_{top} of the clusters increases. In particular, the $O(N\log^2 N)$ estimate (6.8) on the cost of the multi-level solver does not apply anymore; indeed, it relies on the assumption (4.25) that the ranks of the off-diagonal blocks depend logarithmically on the block sizes, and this assumption is not valid in this case (see Table 4).

For this problem, it takes $\varepsilon_{\text{cut}} = 10^{-15}$ to attain an error on the order of 10^{-10}. This corresponds to using about 4370 nodes in total to discretize Γ, about 2150 of which used for the simply graded mesh around the corner. At this setting, the multi-level solver, combined with the rotation scheme, performs about 13.5 times faster than the "brute force" scheme. By extrapolation, we expect the multi-level solver to gain more considerable advantage in the case of solving (7.5) on domains with multiple corners.

Remark 7.1 *The error in Table 4 is the best precision achieved using a simply graded mesh around the corner in double precision arithmetic. To get a higher precision, one can either run the experiment in extended precision arithmetic, or adopt discretization methods other than the simply graded mesh.*

8 Generalizations and conclusions

We have presented a numerical algorithm for the construction of compressed factorizations of inverses of matrices possessing rank-deficient off-diagonal blocks. When applied to matrices arising from the

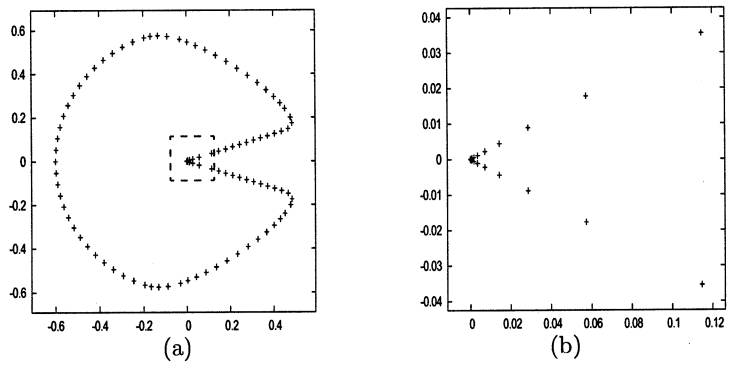

Figure 12: (a) The mesh used to discretize Γ when ε_{cut} is set at 10^{-8}. There are in total 136 subintervals and about 3260 nodes. (b) A close-up of the simply graded mesh around the corner, contained by the dashed box in (a). The mesh consists of 48 subintervals and about 1150 nodes.

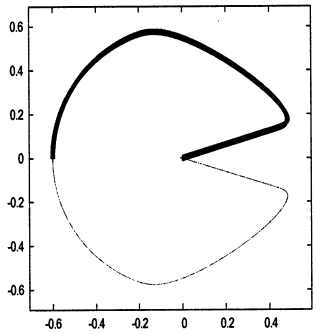

Figure 13: The arrangement of the clusters on the first level under the original parameterization of Γ.

Table 4: Computational results for the boundary integral equation (7.1) associated with the interior Dirichlet problem on the contour Γ shown in Fig. 11. Here, r_{top} denotes the approximate rank of interactions of the clusters on the first level under the original parameterization of Γ, as arranged in Fig. 13.

ε_{cut}	N	t_{rot}	$t_{\text{solve},1}$	$E_{\text{rel},1}$	$t_{\text{solve},2}$	$E_{\text{rel},2}$	r_{top}	$t_{\text{solve},3}$	$E_{\text{rel},3}$
$1.0e-05$	2784	$9.07e-02$	$1.32e+00$	$9.52e-05$	$4.86e+00$	$7.66e-07$	192	$1.26e+01$	$7.22e-05$
$1.0e-07$	3072	$8.85e-02$	$1.57e+00$	$8.48e-06$	$9.05e+00$	$7.20e-06$	252	$1.72e+01$	$7.24e-06$
$1.0e-09$	3408	$9.58e-02$	$2.02e+00$	$5.92e-07$	$2.10e+01$	$5.54e-07$	330	$2.27e+01$	$4.96e-07$
$1.0e-11$	3744	$9.74e-02$	$2.32e+00$	$3.60e-08$	$3.96e+01$	$3.11e-08$	401	$3.15e+01$	$3.39e-08$
$1.0e-13$	4032	$1.04e-01$	$2.85e+00$	$4.04e-09$	$6.14e+01$	$3.78e-09$	464	$3.74e+01$	$3.40e-09$
$1.0e-15$	4368	$1.07e-01$	$3.51e+00$	$3.84e-10$	$9.90e+01$	$3.59e-10$	536	$4.76e+01$	$3.30e-10$

discretization of boundary integral equations associated with the solution of Laplace's equation in two dimensions, the algorithm typically has a cost proportional to $N \log^2 N$, where N is the number of nodes in the discretization of the boundary.

Several straightforward generalizations of the scheme of this paper suggest themselves:

1. The adaptive rotation scheme of this paper generalizes to three dimensions, where it addresses an outstanding problem of considerable interest. In particular, the scheme extends the applicability of fast direct solvers to integral operators defined on boundary surfaces, which are usually specified via parameterizations $r : [0,1] \times [0,1] \rightarrow \mathbb{R}^3$ such that the distance between two points (s_1, t_1) and (s_2, t_2) in the parameter space bears no relation to the distance $\|r(s_1, t_1) - r(s_2, t_2)\|$ between their corresponding points on the surface. A generalization of the scheme to this setting will allow fast direct solvers to utilize the geometry of the boundary that is usually not easily extractable from a given parameterization.

2. The scheme of this paper can be readily extended to the case of boundary integral equations associated with the Helmholtz equation:

$$\Delta u + \omega^2 u = 0, \tag{8.1}$$

assuming ω is not too large. This work is in progress and will be reported at a later date.

3. The direct solver presented in this paper can be modified to compute a compressed complete orthogonal decomposition of the input matrix, rather than a compressed factorization of the inverse. Such an algorithm will have applications to least-squares solutions of rank-deficient systems of equations:

$$Ax = b, \tag{8.2}$$

where A is an n by n matrix of rank $k < n$.

References

[1] A. BJÖRCK, *Numerical methods for least squares problems*, SIAM, Philadelphia, 1996.

[2] G. CHANDLER, *Galkerin's method for boundary integral equations on polygonal domains*, J. Austral. Math. Soc., Series B, 26 (1984), pp. 1-13.

[3] S. CHANDRASEKARAN, M. GU, T. PALS, *A fast ULV decomposition solver for hierarchically semi-separable representations*, SIAM Journal on Matrix Analysis and Applications, Volume 28, Issue 3 (August 2006), pp. 603-622.

[4] Y. CHEN, *A fast direct algorithm for the Lippmann-Schwinger integral equation in two dimensions*, Adv. Comput. Math. 16(2-3) (2002) pp. 175-190.

[5] W.C. CHEW, *An n^2 algorithm for the multiple scattering solution of n scatterers*, Micro. Optical Tech. Lett. 2 (1989) pp. 380-383.

[6] R. COIFMAN, Y. MEYER, *Wavelets: Calderón-Zygmund and multilinear Operators*, Cambridge University Press, 1997.

[7] G. FOLLAND, *Introduction to Partial Differential Equations*, Princeton University Press, Princeton, N.J., 1976.

[8] D. GINES, G. BEYLKIN, J. DUNN, *LU factorization of non-standard forms and direct multiresolution solvers*, Appl. Comput. Harmon. Anal. 5 (2) (1998) 156-201, MR99c:65087.

[9] G. GOLUB, C. V. LOAN, *Matrix Computations, Second Edition*, Johns Hopkins University Press, Baltimore, 1989.

[10] L. GREENGARD, V. ROKHLIN, *A fast algorithm for particle simulations*, J. Computational Physics 73 (1987), pp. 325-348.

[11] MING GU, STANLEY C. EISENSTAT, *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM J. Sci. Comput. 17(4) (1996) pp. 848-869, MR 97h:65053

[12] W. HACKBUSCH, *A sparse matrix arithmetic based on H-matrices. I. Introduction to H-matrices*, Computing 62 (2) (1999) pp. 89-108, MR 2000c:65039.

[13] W. HACKBUSCH, S. BÖRM, *Data-sparse approximation by H^2-matrices*, Computing 69 (1) (2002) pp. 1-35, MR 1 954 142.

[14] J. HELSING, R. OJALA, *Corner singularities for elliptic problems: Integral equations, graded meshes, quadrature, and compressed inverse preconditioning*, Journal of Computational Physics 227, (2008).

[15] O. KELLOG, *Foundations of Potential Theory*, Dover, New York, 1953.

[16] C. KENIG, *Elliptic boundary value problems on Lipschitz domains*, Beijing Lectures in Harmonic Analysis, Ann. of Math. Stud., 112 (1986), pp. 131-183.

[17] R. KRESS, *Integral Equations*, Springer-Verlag, New York, 1999.

[18] R. KRESS, *A Nyström method for boundary integral equations in domains with corners*, Numerische Mathematik, 58 (1991).

[19] P. G. MARTINSSON, V. ROKHLIN, *A fast direct solver for boundary integral equations in two dimensions*, Journal of Computational Physics 205 (2005) pp. 1-23

[20] S. MIKHLIN, *Integral Equations*, Pergamon Press, New York, 1957.

[21] J. STOER, R. BULIRSCH, *Introduction to Numerical Analysis, Second Edition*, Springer-Verlag, 1993.

[22] G. VERCHOTA, *Layer potentials and boundary value problems for Laplace's equation in Lipschitz domains*, Journal of Functional Analysis, 59 (1984), pp. 572-611.

[23] F. WOOLFE, E. LIBERTY, V. ROKHLIN, M. TYGERT, *A fast randomized algorithm for the approximation of matrices*, Applied and Computational Harmonic Analysis 25 (2008) pp. 335-36.