Fast direct solvers for elliptic partial differential equations

by

A. Gillman

B.S., California State University, Northridge, 2003

M.S., California State University, Northridge, 2006

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Applied Mathematics

2011

This thesis entitled:
Fast direct solvers for elliptic partial differential equations
written by A. Gillman
has been approved for the Department of Applied Mathematics

Per-Gunnar Martinsson

Prof. Gregory Beylkin

Dr. Brad Alpert

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Gillman, A. (Ph.D., Applied Mathematics)

Fast direct solvers for elliptic partial differential equations

Thesis directed by Prof. Per-Gunnar Martinsson

The dissertation describes fast, robust, and highly accurate numerical methods for solving boundary value problems associated with elliptic PDEs such as Laplace's and Helmholtz' equations, the equations of elasticity, and time-harmonic Maxwell's equation. In many areas of science and engineering, the cost of solving such problems determines what can and cannot be modeled computationally.

Elliptic boundary value problems may be solved either via discretization of the PDE (e.g., finite element methods) or by first reformulating the equation as an integral equation, and then discretizing the integral equation. In either case, one is left with the task of solving a system of linear algebraic equations that could be very large. There exist a broad range of schemes with linear complexity for solving these equations (multigrid, preconditioned Krylov methods, etc). Most of these schemes are based on "iterative" techniques that build a sequence of approximate solutions that converges to the exact solution. In contrast, the methods described here are "direct" in the sense that they construct an approximation to the inverse (or LU/Cholesky factorization) of the coefficient matrix. Such direct solvers tend to be more robust, versatile, and stable than iterative methods, but have until recently been considered prohibitively expensive for large scale problems. The objective of the dissertation is to demonstrate that in important environments it is possible to construct an approximate inverse with linear computational cost. The methods are for a single solve competitive with the best iterative methods, and can be far faster than any previously available methods in situations where the same coefficient matrix is used in a sequence of problems.

In addition, a new discretization technique for elliptic boundary value problems is proposed. The idea is to first compute the solution operator of a large collection of small domains. The small domains are chosen such that the operator is easily computed to high accuracy. A global

equilibrium equation is then built by equating the fluxes through all internal domain boundaries. The resulting linear system is well-suited to the newly developed fast direct solvers.

Dedication

I dedicate this dissertation to my family, especially my parents, Maudi and Larry, who have patiently supported me through many years of schooling.

Acknowledgements

Foremost, I would like thank my advisor, Gunnar Martinsson. He has gone above and beyond as an advisor, allowing me to work independently, travel to *many* conferences and introducing me to the fast algorithm community. This dissertation would not be possible without him.

Thank you to my committee members for their time and interest in this work. A special thank you to Bradley Alpert for his comments on the manuscript. In addition to serving on my committee, Gregory Beylkin has provided guidance throughout my time at Boulder for this I am grateful.

The faculty in the Department of Applied Mathematics have been fabulous. Jim Curry and Keith Julien were extremely helpful in dealing with the traditional struggles of graduate student life. The Grandview Gang, especially Tom Manteuffel and Steve McCormick, have been supportive and interested in the various projects I have worked on.

Rabia Djellouli's passion and excitement for mathematics inspired me to pursue graduate school. His continued encouragement, support, wisdom and friendship are priceless to me.

During my studies, I have been fortunate to make some great friends that helped ease the stress. Without them, I might not of gotten enjoy the beauty that is Colorado.

Last but not least, thank you to my sisters, Christina and Jennifer who were always there to help me keep things in perspective.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

The dissertation describes fast solution techniques for elliptic boundary value problems, of for example linear elasticity, Stokes, Helmholtz, and time-harmonic Maxwell equations, that are commonly used in the modeling of physical phenomena and hence appear repeatedly in many areas of science and engineering. By developing efficient techniques for solving these problems, this thesis is a contribution in the effort to expand the range of problems that may be modeled computationally.

Broadly speaking, existing numerical methods fall into two categories:

Direct discretization of the partial differential equation (PDE): This is probably the most used solution approach. Common discretization methods include finite element, finite difference and spectral element methods.

Recast the PDE as an integral equation: The integral equation is then discretized with for example a Nyström method or boundary element method. This solution technique is possible when the fundamental solution is known.

For any technique, after discretization one must solve a system of linear algebraic equations that often involves a very large number of degrees of freedom.

Consider the linear system that arises from a discretization of an elliptic boundary value problem

$$\mathsf{A}\mathbf{u} = \mathbf{f}, \tag{1.1}$$

where A is an $N \times N$ matrix, $\mathbf{u}$ and $\mathbf{f}$ are vectors. To solve for $\mathbf{u}$, the classic Gaussian elimination

has computational cost $O(N^3)$. Over the last several decades, a number of fast methods (e.g. multigrid, FFT, FMM) have been developed to solve these linear systems. By "fast," we mean that the computational cost of solving the problem grows as $O(N \log^p N)$ where N is the size of the problem and p is a small integer, normally $p = 0, 1$, or 2. Most fast schemes are based on iterative techniques which build a sequence of approximate solutions and often need a problem specific preconditioner in order to accelerate convergence. As an alternative, we propose the use of fast direct solvers. A **direct solver** constructs an operator T such that

$$\|\mathsf{A}^{-1} - \mathsf{T}\| < \epsilon,$$

where ϵ is a given computational tolerance. (While the $N \times N$ matrix T is dense, usually it is stored in some kind of data-sparse format requiring $O(N)$ memory.)

Remark 1. *Sometimes it is more practical to form an approximate factorization (e.g. LU, Cholesky) of* A*, where linear solves involving the factors are fast. The construction of such factorizations is technically very similar to the problem of constructing an approximate inverse. For the simplicity of presentation, we limit our discussion to approximating the inverse.*

1.1 Advantages of direct solvers

Direct solvers offer several advantages over iterative ones:

Speed-up by large factors for problems involving multiple right hand sides: In many situations, an equation such as (A.1) needs to be solved for several different right-hand sides $\mathbf{f}$. Iterative techniques have a limited ability to take advantage of the fact that the operator is the same in each solve. On the other hand, for a direct method each solve beyond the first simply involves a matrix-vector multiply with the pre-computed inverse. The time required for applying the inverse to a vector is typically much smaller than even the time required for a single application of the original operator using standard techniques.

The ability to solve relatively ill-conditioned problems: Direct solvers allow for the rapid and ac-

curate solution of linear systems involving relatively ill-conditioned matrices. In the context of boundary value problems, such ill-conditioning can be caused by physical ill-conditioning (as observed, e.g., when solving the equations of elasticity on domains with high aspect ratios, or when solving scattering problems near a resonant frequency), but may also be introduced as a side-effect of the mathematical formulation (e.g. when a PDE is discretized with high-order finite elements or when an integral equation formulation based on first kind Fredholm equations is used.)

Increased reliability: Existing iterative methods can be extremely efficient, but their performance relies in subtle ways on spectral properties of the coefficient matrix. In situations where good preconditioners are not available, convergence can be slow, or even non-existent. Direct solvers are inherently more robust, and the prospect of obtaining versatile solvers that work reliably for a broad range of linear systems is one of the key motivations of the work presented.

1.2 Overview of direct solution techniques

The direct solvers described in this dissertation are applicable to several different computational environments. This section describes four such environments and how direct solvers apply to each one.

1.2.1 Boundary integral equations

For some boundary value problems, it is possible to use potential theory to reformulate the problem into a boundary integral equation (BIE) that is well-conditioned. Upon discretization (via e.g. a Nyström or Boundary element method), the resulting linear system is data-sparse in the sense that all off-diagonal blocks admit a low-rank factorization. In particular, for many one-dimensional boundary integral equations (corresponding to PDEs defined in the plane), the matrix that needs be inverted is *Hierarchically Semi-separable* (HSS) [71, 16]. Chapter 2 details a fast inversion technique for HSS matrices that scales linearly with the number of discretization points. Numerical results illustrate the robustness, accuracy and the scaling of the method. In particular,

our implementation demonstrates that a linear system corresponding to an elongated domain with corners involving approximately 10^5 discretization points can be solved to six digits of accuracy in about 50 seconds on a standard desktop computer. This work is also presented in a manuscript that is currently in review [39].

The fast inversion technique has a wide range of applications. For instance, the linear system corresponding the discretization of a one-dimensional boundary integral equation where the boundary is space filling, while not HSS, can be handled by this method. The method builds an approximate inverse with computational cost $O(N^{1.5})$, where N is the number of discretization points. After this matrix is constructed, computing each solution requires linear computational cost. In [38], we present a fast method for constructing the solution operator for boundary value problems in non-homogeneous media by combining the HSS inversion scheme and homogenization techniques. Additionally, the HSS inversion technique serves as a basis for many of the fast solution techniques presented in this thesis.

1.2.2 Finite element or finite difference matrices

One of the first "fast" direct solution techniques for the large sparse system that arises from the finite element or finite difference discretization of a PDE is the nested dissection method [35]. This divide and conquer technique is based on the advantageous reordering of the discretization points minimizes fill-in and results in a method that has $O(N^{1.5})$ computational cost, where N is the number of discretization points. In Chapter 3, we describe a variation of the nested dissection method. It turns out the intermediate worker arrays are HSS matrices. Thus using fast dense matrix algebra for HSS matrices allows the solution technique to be accelerated to linear complexity. Numerical results show that the first solve for a problem involving 16 million unknowns takes about 7 minutes on a standard desktop computer. Each additional solve takes about 0.04 seconds. We are currently working on a manuscript presenting this work.

1.2.3 Poisson problems on infinite regular lattices

The solution of the free-space Poisson problem

$$-\Delta u(\mathbf{x}) = f(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^2, \tag{1.2}$$

takes the form of a convolution

$$u(\mathbf{x}) = \int_{\mathbb{R}^2} \phi_{\text{cont}}(\mathbf{x} - \mathbf{y})\, f(\mathbf{y})\, d\mathbf{y}, \tag{1.3}$$

where ϕ_{cont} is the fundamental solution of the Laplace operator,

$$\phi_{\text{cont}}(\mathbf{x}) = -\frac{1}{2\pi} \log |\mathbf{x}|. \tag{1.4}$$

Approximating the source function f by a sum of point charges, the integral (1.3) is converted to a sum

$$u_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} \phi_{\text{cont}}(\mathbf{x}_i - \mathbf{x}_j)\, f(\mathbf{x}_j), \qquad i = 1,\, 2,\, \ldots,\, N \tag{1.5}$$

which can be computed rapidly via the Fast Multiple method (FMM) [44, 42, 45]. (An analogous conversion appears if the integral (1.3) is approximated via a quadrature rule.)

We mirror this solution technique for the Poisson problem on an infinite regular lattice

$$[\mathsf{A}u](\mathbf{m}) = f(\mathbf{m}), \qquad \mathbf{m} \in \mathbb{Z}^2, \tag{1.6}$$

where $f = f(\mathbf{m})$ and $u = u(\mathbf{m})$ are scalar valued functions on $\mathbb{Z}^2$ and f has finite support. The techniques we will describe apply to many constant coefficient elliptic difference operators, but for simplicity, suppose that A is the so-called discrete Laplace operator

$$[\mathsf{A}\, u](\mathbf{m}) = 4u(\mathbf{m}) - u(\mathbf{m} + \mathbf{e}_1) - u(\mathbf{m} - \mathbf{e}_1) - u(\mathbf{m} + \mathbf{e}_2) - u(\mathbf{m} - \mathbf{e}_2), \qquad \mathbf{m} \in \mathbb{Z}^2. \tag{1.7}$$

In (1.7), $\mathbf{e}_1 = [1, 0]$ and $\mathbf{e}_2 = [0, 1]$ are the canonical basis vectors in $\mathbb{Z}^2$. With the discrete fundamental solution [27, 56, 60, 36] defined via the normalized Fourier integral

$$\phi(\mathbf{m}) = \frac{1}{(2\,\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \frac{\cos(t_1 m_1 + t_2 m_2) - 1}{4\, \sin^2(t_1/2) + 4\, \sin^2(t_2/2)}\, dt_1\, dt_2, \qquad \mathbf{m} = [m_1,\, m_2] \in \mathbb{Z}^2, \tag{1.8}$$

the explicit analytic solution to (1.6) is

$$u(\mathbf{m}) = [\phi * f](\mathbf{m}) = \sum_{\mathbf{n}\in\mathbb{Z}^2} \phi(\mathbf{m}-\mathbf{n})\, f(\mathbf{n}). \tag{1.9}$$

In Chapter 4, we describe a lattice FMM for computing (1.9) rapidly. This method is similar in concept to the so-called "kernel-independent" FMMs but achieves additional speed up by exploiting the discrete geometry. A manuscript presenting this work will be submitted for publication shortly [37].

1.2.4 Elliptic difference equations defined on lattices

In Section 1.2.3, we described a lattice analog of the exact solution to the free space continuum Poisson equation. In this section, we describe a lattice analog of a continuum Laplace boundary value problem. To illustrate, consider the discrete boundary value problem

$$\begin{cases} [\mathsf{A}u](\mathbf{m}) &= 0, & \mathbf{m}\in\Omega, \\ u(\mathbf{m}) &= g(\mathbf{m}), & \mathbf{m}\in\Gamma, \end{cases} \tag{1.10}$$

where Ω is a subset of $\mathbb{Z}^2$ with boundary Γ, parallels the established solution technique for the continuum problem

$$\begin{cases} [-\Delta\, u](\mathbf{x}) = 0, & \mathbf{x}\in\Omega\subset\mathbb{R}^2, \\ u(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x}\in\Gamma. \end{cases} \tag{1.11}$$

Recall that the solution to (1.11) can be expressed as

$$u(\mathbf{x}) = \int_\Gamma D(\mathbf{x},\mathbf{y})\,\sigma(\mathbf{y})\,d\ell(\mathbf{y}), \tag{1.12}$$

where D is the so called *double layer kernel*,

$$D(\mathbf{x},\mathbf{y}) = \frac{\partial}{\partial\mathbf{n}(\mathbf{y})}\Phi(\mathbf{x}-\mathbf{y}) = \mathbf{n}(\mathbf{y})\cdot\nabla_{\mathbf{y}}\Phi(\mathbf{x}-\mathbf{y}) = \frac{\mathbf{n}(\mathbf{y})\cdot(\mathbf{x}-\mathbf{y})}{2\,\pi\,|\mathbf{x}-\mathbf{y}|^2},$$

where $\mathbf{n}(\mathbf{y})$ is the unit normal vector of Γ at $\mathbf{y}$. The function $\sigma(\mathbf{x})$ is the solution to the following second kind Fredholm equation

$$\frac{1}{2}\sigma(\mathbf{x}) + \int_\Gamma D(\mathbf{x},\mathbf{y})\,\sigma(\mathbf{y})\,d\ell(\mathbf{y}) = g(\mathbf{x}), \qquad \mathbf{x}\in\Gamma. \tag{1.13}$$

Hence, in Chapter 5, we propose the solution u of (1.10) may be written as

$$u(\mathbf{m}) = \sum_{n \in \Gamma} d(\mathbf{m}, \mathbf{n})\, \sigma(\mathbf{n}), \tag{1.14}$$

where d is a discrete analog of the continuum double layer potential (1.13). The boundary charge distribution $\sigma(\mathbf{n})$ is chosen such that

$$\sum_{n \in \Gamma} d(\mathbf{m}, \mathbf{n})\, \sigma(\mathbf{n}) = g(\mathbf{m}), \qquad \mathbf{m} \in \Gamma. \tag{1.15}$$

As with the linear system resulting from the Nyström discretization of (1.13), equation (1.15) is well-conditioned and HSS. Thus σ can be found rapidly using the fast inversion technique presented in Chapter 2.

Chapter 5 also presents techniques for handling lattices with imperfections such as inclusions. The technique introduces an additional unknown for every imperfection in the lattice. By doing this, we are able to utilize both the lattice FMM and the HSS solver, resulting in a solution technique that scales as $O(N_{\text{inc}} + N_{\text{boundary}})$, where N_{inc} denotes the number of inclusion points, and N_{boundary} denotes the number of points on the boundary of Ω.

The paper [36] provides an additional presentation of this work.

1.3 Alternative discretization technique

In addition to the collection of direct solvers, this dissertation also presents a new discretization technique for elliptic boundary value problems in Chapter 6.

The idea basic idea of the method is to first compute the solution operator also known as the Neumann-to-Dirichlet operator for a large collection of small domains. These small domains are chosen such that:

- The union of the small domains is the entire domain.
- Each small domain shares at most a portion of its boundary with any other domain.
- The solution operator may be computed easily via least squares technique.

Using the fluxes through the boundary of each of the small domains as unknowns, a global equilibrium equation is formed. The linear system is ideally suited for fast direct solvers. Preliminary results are presented.

Chapter 2

A linear complexity direct solver for integral equations on one-dimensional domains

This chapter describes techniques for numerically solving equations of the type

$$a(t)\,q(t) + \int_0^T b(t,t')\,q(t')\,dt' = f(t), \qquad t \in I, \tag{2.1}$$

where $I = [0, T]$ is an interval on the line, and where $a : I \to \mathbb{R}$ and $b : I \times I \to \mathbb{R}$ are given functions. We observe that a boundary integral equation (BIE) such as

$$a(\mathbf{x})\,q(\mathbf{x}) + \int_\Gamma b(\mathbf{x},\mathbf{x}')\,q(\mathbf{x}')\,dl(\mathbf{x}') = f(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{2.2}$$

where Γ is a simple curve in the plane takes the form (2.1) upon parameterization of the curve. The case of a domain Γ that consists of several non-connected simple curves can also be handled.

Upon discretization, equation (2.1) takes the form

$$\mathsf{A}\,\mathbf{q} = \mathbf{f} \tag{2.3}$$

where A is a dense matrix of size, say, $N \times N$. When N is large, standard practice for rapidly solving a system such as (2.3) is to use an iterative solver (such as GMRES, conjugate gradients, etc.) in which the matrix-vector multiplications are accelerated via a "fast" method such as the Fast Multipole Method (FMM) [44], panel clustering [48], Barnes-Hut [5], etc. When the integral equation (2.1) is a Fredholm equation of the second kind, the iteration typically converges rapidly, and a linear solver of effectively $O(N)$ complexity results. In contrast, this chapter reviews and

extends a number of recently developed *direct* solvers that in a single pass compute a data-sparse representation of a matrix S (a "solution operator") that satisfies

$$\mathsf{S} \approx \mathsf{A}^{-1}.$$

Once a representation of S is available, the solution of (2.3) is of course easily constructed:

$$\mathbf{q} \approx \mathsf{S}\,\mathbf{f}. \tag{2.4}$$

We will demonstrate that in many important environments (such as, e.g., the BIEs associated with Laplace's equation in the plane), the matrix S can be constructed in $O(N)$ operations.

The direct solver presented scales linearly for most boundary integral equations associated with the classical boundary value problems of mathematical physics (Laplace, elasticity, Helmholtz, Yukawa, Stokes, etc.) There are two important exceptions for which it has $O(N^{1.5})$ computational cost (1) problems involving highly oscillatory kernels such as Helmholtz equation at short wavelengths, and (2) domain boundaries that tend to "fill space" in the sense illustrated in Figure 2.1. We will demonstrate that both high accuracy and speed can be maintained even for non-smooth boundaries.

The direct solver is also applicable to many integral equations of the form (2.1) that arise in the analysis of special functions [76], in evaluating conformal maps [68], and in the analysis of two-point boundary value problems [72].

Using the direct solver for (2.1) can be viewed as a process consisting of four steps. Letting ε denote a user specified computational tolerance, the four steps are:

(i) Quadrature nodes and quadrature weights for a Nyström discretization are created: The interval

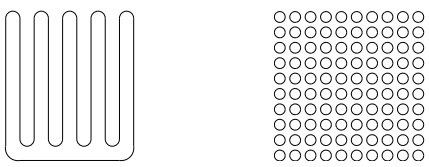


Figure 2.1: Contours for which the direct solver will *not* achieve $O(N)$ complexity.

$[0, T]$ is split into panels, and Gaussian nodes are placed on each panel. Customized quadrature weights are constructed using the method of [79] which ensures high accuracy even in the presence of weakly singular kernels (and for BIEs on domains with corners).

(ii) Construction of the coefficient matrix: The matrix A in (2.3) is an $N \times N$ matrix that is dense, but whose off-diagonal blocks are to high accuracy rank-deficient. We exploit this fact, and compute an approximant $\mathsf{A}_{\text{approx}}$ which is stored in the data-sparse format very similar to the *Hierarchically Semi-Separable (HSS)* format of [71, 16].

(iii) Inversion of the coefficient matrix: The approximant $\mathsf{A}_{\text{approx}}$ of the coefficient matrix is inverted using a variation of the technique of [72, 61] to produce the solution operator $\mathsf{S} = \mathsf{A}_{\text{approx}}^{-1}$. The inversion is exact up to round-off errors.

(iv) Application of the approximate inverse: The solution operator S is applied to the given data vector $\mathbf{f}$ to produce the solution $\mathbf{q}$, *cf.* (2.4).

Each of the four steps typically requires $O(N)$ work when applied to the standard equations of mathematical physics (with the two exceptions mentioned previously). The constants of proportionality depend on the specific environment, but in general, Step *(ii)* is the most expensive. The cost of Step *(iv)* is tiny, meaning that the proposed procedure is particularly effective in environments where a sequence of equations with the same coefficient matrix need to be solved.

Remark 2. *The computations in Steps (iii) and (iv) are independent of the specific problem being solved, and can be implemented as "black-box" codes.*

The general idea of exploiting rank-deficiencies in the off-diagonal blocks of the matrix A in (2.3) is the foundation for many "fast" matrix-vector multiplication algorithms (e.g. the Fast Multipole Method [44], panel clustering [48], Barnes-Hut [5]) which improve the efficiency of iterative solvers. The observation that such rank-deficiencies can also be to accelerate dense matrix algebra, such as matrix inversion, matrix-matrix multiplication, etc., was made in earlier work on $\mathcal{H}$-matrices [49]. The early versions of these methods have $O(N(\log N)^p)$ complexity for some small

integer p. Some of these operations were later accelerated to $O(N)$ complexity in the context of $\mathcal{H}^2$-matrices [11].

The direct solver described in this chapter is an evolution of the scheme presented in [61], which in turn draws on the earlier work [7, 66, 72]. The major difference between the method in [61] is the separation of the compression and inversion steps. Besides making the presentation of the algorithm much clearer, this separation allows for other improvements, including:

Improved versatility: Separating the task of compression from the task of inversion makes it much easier to apply the direct solver to new applications. If a BIE with a different kernel is to be solved, a slight modification of the compression step (Step (ii)) is sufficient. It also opens up the possibility of combining the direct solver with generic compression techniques based on randomized sampling, e.g., those described in [57].

Improved quadratures: The version of the algorithm described in this chapter is compatible with the quadratures of [51, 12] which enable the handling of BIEs defined on domains with corners, and the quadratures of [79] which simplify the handling of singular kernels.

Improved theoretical analysis: The presented direct solver is expressed transparently as a telescoping matrix factorization. This allows for a simplified error and stability analysis, as illustrated by, e.g., Lemma 1 and Corollary 1.

Improved interoperability with other data-sparse matrix formats: The new version of the algorithm makes it clear that the data-sparse format used to represent both the coefficient matrix and its inverse are essentially identical to the *Hierarchically Semi-Separable* (HSS) format of [71, 16]. This opens up the possibility of combining the compression techniques described in this chapter with recently developed inversion and factorization algorithms for HSS matrices [19].

Remark 3. *This chapter uses the terms "block separable" (BS) and "hierarchically block separable" (HBS). The HBS format is essentially identical to the HSS format. The terms BS and HBS were introduced for local purposes only since they clarify the description of the algorithm. There is no*

intention to replace the well-established term "HSS."

The chapter proceeds by explaining in detail each of the four steps that comprise the direct solver. Section 2.1 introduces notation and reviews the Nyström discretization method for integral equations. Section 2.2 describes an accelerated direct solver based on a simplistic tessellation of an $N \times N$ matrix A into $p \times p$ blocks in such a way that all off-diagonal blocks are rank deficient. This method has complexity $O(p^{-2}\, N^3 + p^3\, k^3)$ where k is the rank of the off-diagonal blocks. To attain better asymptotic complexity, a more complicated hierarchical tessellation of the matrix must be implemented. This data structure is described in Section 2.3, and an $O(N\, k^2)$ inversion technique is then described in Section 2.4. Section 2.5 describes efficient techniques for computing the data-sparse representation in the first place. Then some numerical experiments are presented illustrating the performance of the proposed method. Section 2.7 describes possible extensions of the work.

2.1 Preliminaries

This section introduces notation, and briefly reviews some known techniques.

2.1.1 Notation

We say that a matrix U is *orthogonal* if its columns form an orthonormal set. An orthonormal matrix U preserves geometry in the sense that $|\mathsf{U}\,\mathbf{x}| = |\mathbf{x}|$ for every vector $\mathbf{x}$. We use the notation of [41] to denote submatrices: If A is an $m \times n$ matrix with entries $\mathsf{A}(i, j)$, and if $I = [i_1,\, i_2,\, \ldots,\, i_p]$ and $J = [j_1,\, j_2,\, \ldots,\, j_q]$ are two index vectors, then the associated $p \times q$ submatrix is expressed as

$$\mathsf{A}(I, J) = \begin{bmatrix} a_{i_1,j_1} & \cdots & a_{i_1,j_q} \\ \vdots & & \vdots \\ a_{i_p,j_1} & \cdots & a_{i_p,j_q} \end{bmatrix}.$$

For column- and row-submatrices, we use the standard abbreviations

$$\mathsf{A}(:\,, J) = \mathsf{A}([1,\, 2,\, \ldots,\, m], J), \qquad \text{and} \qquad \mathsf{A}(I, :\,) = \mathsf{A}(I, [1,\, 2,\, \ldots,\, n]).$$

2.1.2 The Interpolatory Decomposition (ID)

An $m \times n$ matrix B of rank k admits the factorization

$$\mathsf{B} = \mathsf{U}\,\mathsf{B}(J, :\,),$$

where $J = [j_1, \ldots, j_k]$ is a vector of integers such that $1 \leq j_i \leq m$, and U is a $m \times k$ matrix that contains the $k \times k$ identity matrix I_k (specifically, $\mathsf{U}(J, :\,) = \mathsf{I}_k$). Moreover, no entry of U is larger than one. Computing the ID of a matrix is in general combinatorially expensive, but if the restriction on element size of U is relaxed slightly to say that, for instance, each entry of U is bounded by 2, then very efficient schemes are available. See [46, 21] for details.

2.1.3 Nyström discretization of integral equations in one dimension

In this section, we very briefly describe some variations of the classical Nyström method for discretizing an integral equation such as (2.1). The material is well-known and we refer to [3] for details.

For an integral equation with a smooth kernel $k(t, t')$, the Nyström method is particularly simple. The starting point is a quadrature rule for the interval $[0, T]$ with nodes $\{t_i\}_{i=1}^N \subset [0,\, T]$ and weights $\{\omega_i\}_{i=1}^N$ such that

$$\int_0^T b(t_i, t')\, q(t')\, dt' \approx \sum_{j=1}^{n} b(t_i, t_j)\, q(t_j)\, \omega_j, \qquad i = 1,\, 2,\, \ldots, N.$$

Then the discretized version of (2.1) is obtained by enforcing that

$$a(t_i)\, q(t_i) + \sum_{j=1}^{n} b(t_i, t_j)\, q(t_j)\, \omega_j = f(t_i), \qquad i = 1,\, 2,\, \ldots, N. \tag{2.5}$$

We write (2.5) compactly as

$$\mathsf{A}\,\mathbf{q} = \mathbf{f},$$

where A is the $N \times N$ matrix with entries

$$\mathsf{A}(i, j) = \delta_{i,j}\, a(t_i) + b(t_i,\, t_j)\, \omega_j, \qquad i, j = 1,\, 2,\, 3,\, \ldots,\, N.$$

where $\mathbf{f}$ is the vector with entries

$$\mathbf{f}(i) = f(t_i), \qquad i = 1,\, 2,\, 3,\, \ldots,\, N,$$

and where $\mathbf{q}$ is the approximate solution which satisfies

$$\mathbf{q}(i) \approx q(t_i), \qquad i = 1,\, 2,\, 3,\, \ldots,\, N.$$

We have found that using a composite quadrature rule with a 10-point standard Gaussian quadrature on each panel is a versatile and highly accurate choice.

Remark 4 (Singular kernels)**.** *Some of the numerical examples described in Section 5.5 involve kernels with logarithmically singular kernels,*

$$k(t, t') \sim \log|t - t'|, \qquad \text{as } t' \to t.$$

A standard quadrature rule designed for smooth functions would lose almost all accuracy on the panels where t and t' are close, but this can be remedied by modifying the matrix entries near the diagonal. For instance, when Gaussian quadrature nodes are used, the procedure described in [79] gives very accurate results. Alternatively, the Rokhlin-Kapur [52] procedure starts with a standard trapezoidal rule and modifies the weights near the end points to achieve high order convergence. This is a simpler method than the modified Gaussian rule of [79] but typically also produces lower accuracy.

Remark 5 (Contours with corners)**.** *Discretizing an integral equation such as (2.2) may be challenging if the contour Γ is not smooth. When $\mathbf{x} \in \Gamma$ is a corner point, the function $\mathbf{x}' \mapsto b(\mathbf{x}, \mathbf{x}')$ typically has a singularity at $\mathbf{x}$. It has been demonstrated [51, 12] that in many cases of practical interest, it is nevertheless possible to use standard quadrature weights designed for smooth functions, as long as the discretization is locally refined near the corner. The drawback is that such refinement may increase the system size in an undesirable way but as [51] demonstrates, the system size can be reduced via a local pre-computation. In this text, we demonstrate that it is alternatively possible to use general purpose direct solvers to achieve the same effect.*

2.2 Inversion of block separable matrices

In this section, we define what it means for a matrix to be "block separable" and describe a simple technique for inverting such a matrix.

Let A be an $np \times np$ matrix that is blocked into $p \times p$ blocks, each of size $n \times n$:

$$\mathsf{A} = \begin{bmatrix} \mathsf{D}_1 & \mathsf{A}_{1,2} & \mathsf{A}_{1,3} & \cdots & \mathsf{A}_{1,p} \\ \mathsf{A}_{2,1} & \mathsf{D}_2 & \mathsf{A}_{2,3} & \cdots & \mathsf{A}_{2,p} \\ \vdots & \vdots & \vdots & & \vdots \\ \mathsf{A}_{p,1} & \mathsf{A}_{p,2} & \mathsf{A}_{p,3} & \cdots & \mathsf{D}_p \end{bmatrix}. \tag{2.6}$$

We say that A is "block separable" with "block-rank" k if for $\tau = 1, 2, \ldots, p$, there exist $n \times k$ matrices U_τ and V_τ such that each off-diagonal block $\mathsf{A}_{\sigma,\tau}$ of A admits the factorization

$$\begin{array}{ccccccc} \mathsf{A}_{\sigma,\tau} & = & \mathsf{U}_\sigma & \tilde{\mathsf{A}}_{\sigma,\tau} & \mathsf{V}_\tau^*, & \sigma, \tau \in \{1, 2, \ldots, p\}, & \sigma \neq \tau. \\ n \times n & & n \times k & k \times k & k \times n & & \end{array} \tag{2.7}$$

Observe that the columns of U_σ must form a basis for the columns of all off-diagonal blocks in row σ, and analogously, the columns of V_τ must form a basis for the rows in all the off-diagonal blocks in column τ. When (2.7) holds, the matrix A admits a block factorization

$$\begin{array}{ccccccc} \mathsf{A} & = & \underline{\mathsf{U}} & \tilde{\mathsf{A}} & \underline{\mathsf{V}}^* & + & \underline{\mathsf{D}}, \\ np \times np & & np \times kp & kp \times kp & kp \times np & & np \times np \end{array} \tag{2.8}$$

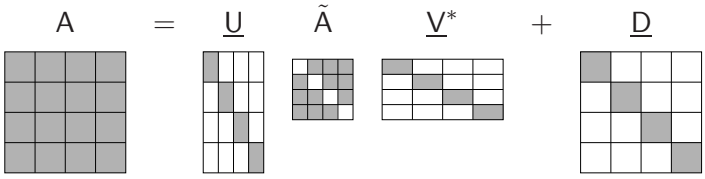
where

$$\mathsf{U} = \mathrm{diag}(\mathsf{U}_1, \mathsf{U}_2, \ldots, \mathsf{U}_p), \quad \mathsf{V} = \mathrm{diag}(\mathsf{V}_1, \mathsf{V}_2, \ldots, \mathsf{V}_p), \quad \mathsf{D} = \mathrm{diag}(\mathsf{D}_1, \mathsf{D}_2, \ldots, \mathsf{D}_p),$$

and

$$\tilde{\mathsf{A}} = \begin{bmatrix} 0 & \tilde{\mathsf{A}}_{12} & \tilde{\mathsf{A}}_{13} & \cdots \\ \tilde{\mathsf{A}}_{21} & 0 & \tilde{\mathsf{A}}_{23} & \cdots \\ \tilde{\mathsf{A}}_{31} & \tilde{\mathsf{A}}_{32} & 0 & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}.$$

The block structure of formula (2.8) for $p = 4$ is illustrated below:



(2.9)

The idea is that by excising the diagonal blocks from A, we obtain a rank-deficient matrix $\mathsf{A} - \underline{\mathsf{D}}$ that can be factored with block diagonal flanking matrices: $\mathsf{A} - \underline{\mathsf{D}} = \underline{\mathsf{U}}\,\tilde{\mathsf{A}}\,\underline{\mathsf{V}}^*$.

The inverse of a block-separable matrix can rapidly be constructed using the following simple variation of the classical Sherman-Morrison-Woodbury formula:

Lemma 1. *Suppose that* A *is an* $N \times N$ *invertible matrix. Suppose further that* K *is a positive integer such that* $K < N$*, that* A *admits the decomposition*

$$\begin{array}{ccccccc} \mathsf{A} & = & \mathsf{U} & \tilde{\mathsf{A}} & \mathsf{V}^* & + & \mathsf{D}, \\ N \times N & & N \times K & K \times K & K \times N & & N \times N \end{array} \tag{2.10}$$

and that the matrices D*,* $(\mathsf{V}^*\,\mathsf{D}^{-1}\,\mathsf{U})$*, and* $\left(\tilde{\mathsf{A}} + (\mathsf{V}^*\,\mathsf{D}^{-1}\,\mathsf{U})^{-1}\right)$ *are invertible. Then*

$$\mathsf{A}^{-1} = \mathsf{E}\,(\tilde{\mathsf{A}} + \hat{\mathsf{D}})^{-1}\,\mathsf{F}^* + \mathsf{G}, \tag{2.11}$$

where

$$\hat{\mathsf{D}} = \left(\mathsf{V}^*\,\mathsf{D}^{-1}\,\mathsf{U}\right)^{-1}, \tag{2.12}$$

$$\mathsf{E} = \mathsf{D}^{-1}\,\mathsf{U}\,\hat{\mathsf{D}}, \tag{2.13}$$

$$\mathsf{F} = (\hat{\mathsf{D}}\,\mathsf{V}^*\,\mathsf{D}^{-1})^*, \tag{2.14}$$

$$\mathsf{G} = \mathsf{D}^{-1} - \mathsf{D}^{-1}\,\mathsf{U}\,\hat{\mathsf{D}}\,\mathsf{V}^*\,\mathsf{D}^{-1}. \tag{2.15}$$

When A is block-separable, (2.10) holds with block diagonal matrices U, V, and D. The matrices $\hat{\mathsf{D}}$, E, F, and G can then be evaluated rapidly, and Lemma 1 can be said to reduce the task of inverting the $np \times np$ matrix A, to the task of inverting the $kp \times kp$ matrix $\tilde{\mathsf{A}} + \hat{\mathsf{D}}$.

Proof of Lemma 1: Consider the equation

$$\left(\mathsf{U}\,\tilde{\mathsf{A}}\,\mathsf{V}^* + \mathsf{D}\right)\mathbf{q} = \mathbf{u}. \tag{2.16}$$

We will prove that (2.11) holds by proving that the solution $\mathbf{q}$ of (2.16) is the right hand side of (2.11) applied to $\mathbf{u}$. First we set

$$\hat{\mathbf{q}} = \mathsf{V}^* \, \mathbf{q}. \tag{2.17}$$

Then (2.16) can be written

$$\mathsf{U} \, \tilde{\mathsf{A}} \, \hat{\mathbf{q}} + \mathsf{D} \, \mathbf{q} = \mathbf{u}. \tag{2.18}$$

Solving (2.18) for $\mathbf{q}$ and inserting the result in (2.17), we obtain

$$(I + \underbrace{\mathsf{V}^* \, \mathsf{D}^{-1} \, \mathsf{U}}_{=\hat{\mathsf{D}}^{-1}} \, \tilde{\mathsf{A}}) \, \hat{\mathbf{q}} = \mathsf{V}^* \, \mathsf{D}^{-1} \, \mathbf{u}. \tag{2.19}$$

Multiplying (2.19) by $\hat{\mathsf{D}}$ we find that

$$(\hat{\mathsf{D}} + \tilde{\mathsf{A}}) \, \hat{\mathbf{q}} = \underbrace{\hat{\mathsf{D}} \, \mathsf{V}^* \, \mathsf{D}^{-1}}_{=\mathsf{F}^*} \, \mathbf{u}. \tag{2.20}$$

Now note that from (2.18) it also follows that

$$\mathbf{q} = -\mathsf{D}^{-1} \, \mathsf{U} \, \tilde{\mathsf{A}} \, \hat{\mathbf{q}} \; + \mathsf{D}^{-1} \, \mathbf{u}. \tag{2.21}$$

From (2.20) we know that

$$\tilde{\mathsf{A}} \, \hat{\mathbf{q}} = -\hat{\mathsf{D}} \, \hat{\mathbf{q}} + \mathsf{F}^* \, \mathbf{u}. \tag{2.22}$$

Inserting (2.22) into (2.21), we obtain

$$\mathbf{q} = -\mathsf{D}^{-1} \, \mathsf{U} \, \big(-\hat{\mathsf{D}} \, \hat{\mathbf{q}} + \mathsf{F}^* \, \mathbf{u}\big) + \mathsf{D}^{-1} \, \mathbf{u} = \underbrace{\mathsf{D}^{-1} \, \mathsf{U} \, \hat{\mathsf{D}}}_{=\mathsf{E}} \, \hat{\mathbf{q}} + \underbrace{\big(\mathsf{D}^{-1} - \mathsf{D}^{-1} \, \mathsf{U} \, \mathsf{F}^*\big)}_{=\mathsf{G}} \, \mathbf{u}. \tag{2.23}$$

Solving (2.20) for $\hat{\mathbf{q}}$ and inserting the result in (2.23), we obtain the expression (2.11). □

The technique provided by the lemma is a useful tool in its own right. It often reduces the cost of inverting an $N \times N$ matrix from the $O(N^3)$ cost of Gaussian elimination, to $O(N^{1.8})$, see Remark 6. Even more significant is that for many matrices, including the ones under consideration in this chapter, the process described can be continued recursively which leads to an $O(N)$ inversion scheme. The required hierarchical representations of matrices are described in Section 2.3, and the $O(N)$ inversion scheme is given in Section 2.4.

Remark 6. *To assess the computational cost of applying Lemma 1, suppose that* A *is a BS matrix whose* $p \times p$ *blocks of size* $n \times n$ *satisfy (2.7). Then evaluating the factors* $\hat{\mathsf{D}}$, E, F, *and* G *requires* $O(p\,n^3)$ *operations. Evaluating* $(\tilde{\mathsf{A}} + \hat{\mathsf{D}})^{-1}$ *requires* $O(p^3\,k^3)$ *operations. The total cost* T_{BS} *of evaluating the factors in formula (2.11) therefore satisfies:*

$$T_{\mathrm{BS}} \sim p\,n^3 + p^3\,k^3. \tag{2.24}$$

The cost (2.24) should be compared to the $O(p^3\,n^3)$ *cost of evaluating* A^{-1} *directly. To elucidate the comparison, suppose temporarily that we are given a matrix* A *of fixed size* $N \times N$, *and can choose how many blocks* p *we wish to partition it into. Then* $n \approx N/p$, *and the total cost satisfies*

$$T_{\mathrm{BS}} \sim p^{-2}\,N^3 + p^3\,k^3. \tag{2.25}$$

If the rank of interaction k *is independent of the block size, we can set* $p \sim N^{3/5}$, *whence*

$$T_{\mathrm{BS}} \sim k^3\,N^{9/5}. \tag{2.26}$$

In practice, the numerical rank of the off-diagonal blocks typically increases slightly as the block size n *is increased, but the increase tends to be moderate. For instance, for the matrices under consideration in this chapter, one typically sees* $k \sim \log(n) \sim \log(N/p)$. *In such an environment, setting* $p \sim N^{3/5}\,(\log N)^{-3/5}$ *transforms (2.25) to, at worst,*

$$T_{\mathrm{BS}} \sim (\log N)^{6/5}\,N^{9/5}.$$

The calculations in this remark do not include the cost of actually constructing the factors U, V, D. *For a general matrix, this cost is* $O(k\,N^2)$, *but for the matrices under consideration in this chapter, techniques with better asymptotic complexity are described in Section 2.5.*

We close by showing that when the matrix A is symmetric positive definite, the assumption in Lemma 1 that certain intermediate matrices are invertible can be omitted:

Corollary 1. *Let* A *be a symmetric positive definite (spd) matrix that admits a factorization*

$$\begin{array}{ccccccccc} \mathsf{A} & = & \mathsf{U} & & \tilde{\mathsf{A}} & & \mathsf{U}^* & + & \mathsf{D}, \\ N \times N & & N \times K & & K \times K & & K \times N & & N \times N \end{array} \tag{2.27}$$

where $\ker(\mathsf{U}) = \{\mathbf{0}\}$ *and* D *is a block diagonal submatrix of* A*. Then the matrices* D, $\left(\mathsf{U}^*\,\mathsf{D}^{-1}\,\mathsf{U}\right)$, *and* $\tilde{\mathsf{A}} + (\mathsf{U}^*\,\mathsf{D}^{-1}\mathsf{U})^{-1}$ *are spd (and hence invertible).*

Proof of Corollary 1: That D is spd follows immediately from the fact that it is a block diagonal submatrix of a spd matrix.

To show that $\mathsf{U}^*\,\mathsf{D}^{-1}\,\mathsf{U}$ is spd, we pick any $\mathbf{x} \neq \mathbf{0}$, set $\mathbf{y} = \mathsf{D}^{-1}\,\mathsf{U}\,\mathbf{x}$, observe that $\mathbf{y} \neq 0$ since $\ker(\mathsf{U}) = \{\mathbf{0}\}$, and then we find that $\langle \mathsf{U}^*\,\mathsf{D}^{-1}\mathsf{U}\mathbf{x}, \mathbf{x}\rangle = \langle \mathsf{D}^{-1}\,\mathsf{U}, \mathsf{U}\mathbf{x}\rangle = \langle \mathbf{y}, \mathsf{D}\mathbf{y}\rangle > 0$ since D is spd.

It remains only to prove that $\tilde{\mathsf{A}} + (\mathsf{U}^*\mathsf{D}^{-1}\mathsf{U})^{-1}$ is spd. To this end, define $\hat{\mathsf{D}}$ and E via

$$\hat{\mathsf{D}} = (\mathsf{U}^*\mathsf{D}^{-1}\mathsf{U})^{-1}$$

$$\mathsf{E} = \mathsf{D}^{-1}\mathsf{U}\hat{\mathsf{D}}.$$

Then

$$\begin{aligned}\tilde{\mathsf{A}} + \hat{\mathsf{D}} = \hat{\mathsf{D}}\,\left(\hat{\mathsf{D}}^{-1}\,\tilde{\mathsf{A}}\,\hat{\mathsf{D}}^{-1} + \hat{\mathsf{D}}^{-1}\right)\,\hat{\mathsf{D}} &= \hat{\mathsf{D}}\,\left(\mathsf{U}^*\,\mathsf{D}^{-1}\,\mathsf{U}\,\tilde{\mathsf{A}}\,\mathsf{U}^*\mathsf{D}^{-1}\,\mathsf{U} + \hat{\mathsf{D}}^{-1}\right)\,\hat{\mathsf{D}} \\ &= \hat{\mathsf{D}}\,\left(\mathsf{U}^*\,\mathsf{D}^{-1}\,(\mathsf{A} - \mathsf{D})\mathsf{D}^{-1}\,\mathsf{U} + \mathsf{U}^*\mathsf{D}^{-1}\mathsf{U}\right)\,\hat{\mathsf{D}} = \hat{\mathsf{D}}\,\mathsf{U}^*\,\mathsf{D}^{-1}\,\mathsf{A}\mathsf{D}^{-1}\,\mathsf{U}\hat{\mathsf{D}} = \mathsf{E}^*\mathsf{A}\mathsf{E}.\end{aligned}$$

That $\tilde{\mathsf{A}} + (\mathsf{U}^*\mathsf{D}^{-1}\mathsf{U})^{-1}$ is spd now follows since $\ker(\mathsf{E}) = \{\mathbf{0}\}$ and A is spd. □

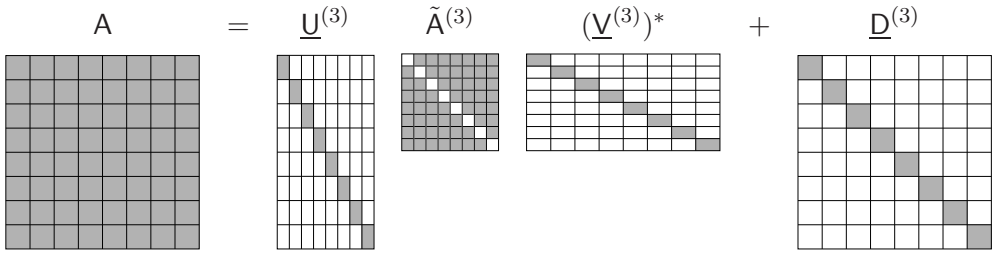
Remark 7. *The proof of Corollary 1 demonstrates that the stability of the method can readily be assessed by tracking the conditioning of the matrices* E*. If these matrices are close to orthogonal (i.e. all their singular values are similar in magnitude) then the compressed matrix* $\tilde{\mathsf{A}} + \hat{\mathsf{D}}$ *has about the same distribution of singular values as* A *since* $\tilde{\mathsf{A}} + \hat{\mathsf{D}} = \mathsf{E}^*\,\mathsf{A}\,\mathsf{E}$*.*

2.3 Hierarchically block separable matrices

In this section, we define what it means for an $N \times N$ matrix A to be HBS. Section 2.3.1 informally describes the basic ideas. Section 4.5 describes a simple binary tree of subsets of the index vector $[1, 2, \ldots, N]$. Section 2.3.3 provides a formal definition of the HBS property. Section 2.3.4 describes how an HBS matrix may be expressed a telescoping factorization. Section 2.3.5 describes an $O(N)$ procedure for applying an HBS matrix to a vector.

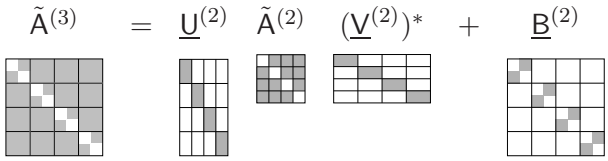
2.3.1 Heuristic description

The HBS property first of all requires A to be BS. Supposing that A consists of 8×8 blocks, this means that A admits a factorization, *cf.* (2.9):



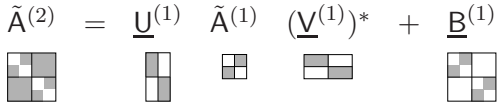
(2.28)

The superscripts on the right-hand side of (2.28) indicate that the factorization is at "level 3." We next require the smaller matrix $\tilde{\mathsf{A}}^{(3)}$ to be BS, and to admit the analogous factorization:



(2.29)

In forming (2.29), we reblocked the matrix $\tilde{\mathsf{A}}^{(3)}$ by merging blocks in groups of four. The purpose is to "reintroduce" rank deficiencies in the off-diagonal blocks. In the final step in the hierarchy, we require that upon reblocking, $\tilde{\mathsf{A}}^{(2)}$ is BS and admits a factorization:

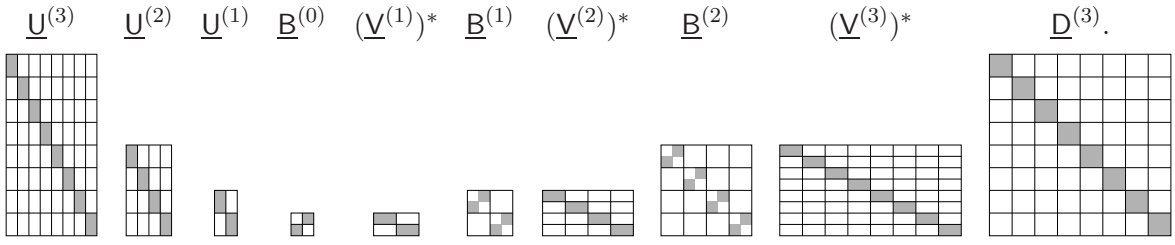


(2.30)

Combining (2.28), (2.29), and (2.30), we find that A can be expressed as

$$\mathsf{A} = \underline{\mathsf{U}}^{(3)}\big(\underline{\mathsf{U}}^{(2)}\big(\underline{\mathsf{U}}^{(1)}\,\underline{\mathsf{B}}^{(0)}\,(\underline{\mathsf{V}}^{(1)})^* + \underline{\mathsf{B}}^{(1)}\big)(\underline{\mathsf{V}}^{(2)})^* + \underline{\mathsf{B}}^{(2)}\big)(\underline{\mathsf{V}}^{(3)})^* + \underline{\mathsf{D}}^{(3)}. \tag{2.31}$$

The block structure of the right hand side of (2.31) is:



In other words, the HBS property lets us completely represent a matrix in terms of certain block diagonal factors. The total cost of storing these factors is $O(N\,k)$, and the format is an example of a so-called "data-sparse" representation of the matrix.

Remark 8. *In describing the HBS property, we assumed that all off-diagonal blocks at all levels have the same rank k. We do this for notational clarity only. In practice, the minimal rank tends to vary slightly from block to block, and to moderately increase on the coarser levels. In the numerical examples in Section 5.5, all codes determine the rank adaptively for each block.*

2.3.2 Tree structure

The HBS representation of an $N \times N$ matrix A is based on a partition of the index vector $I = [1, 2, \ldots, N]$ into a binary tree structure. For simplicity, we limit attention to binary tree structures in which every level is fully populated. We let I form the root of the tree, and give it the index 1, $I_1 = I$. We next split the root into two roughly equi-sized vectors I_2 and I_3 so that $I_1 = I_2 \cup I_3$. The full tree is then formed by continuing to subdivide any interval that holds more than some preset fixed number n of indices. We use the integers $\ell = 0, 1, \ldots, L$ to label the different levels, with 0 denoting the coarsest level. A *leaf* is a node corresponding to a vector that never got split. For a non-leaf node τ, its *children* are the two boxes σ_1 and σ_2 such that $I_\tau = I_{\sigma_1} \cup I_{\sigma_2}$, and τ is then the *parent* of σ_1 and σ_2. Two boxes with the same parent are called *siblings*. These definitions are illustrated in Figure 6.6

Remark 9. *The HBS format works with a broad range of different tree structures. It is permissible to split a node into more than two children if desirable, to distribute the points in an index set unevenly among its children, to split only some nodes on a given level, etc. This flexibility is essential when A approximates a non-uniformly discretized integral operator; in this case, the partition tree it constructed based on a geometric subdivision of the domain of integration. The spatial geometry of a box then dictates whether and how it is to be split. The algorithms presented in this chapter can easily be modified to accommodate general trees.*

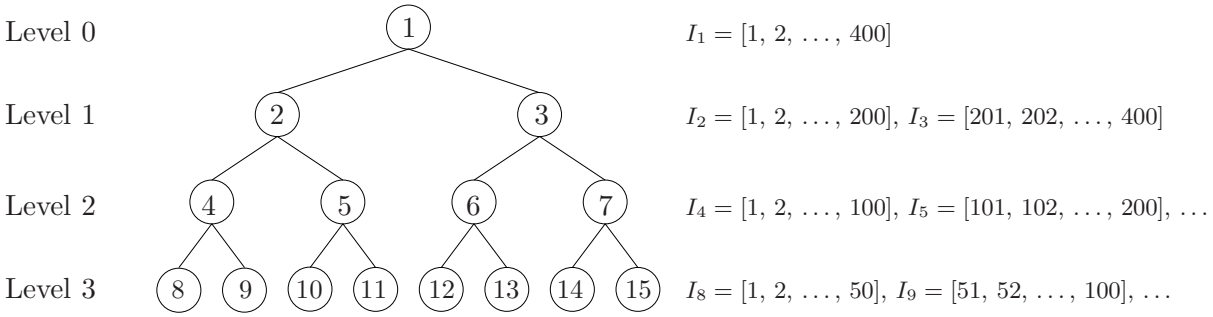


Figure 2.2: Numbering of nodes in a fully populated binary tree with $L = 3$ levels. The root is the original index vector $I = I_1 = [1, 2, \ldots, 400]$.

2.3.3 Definition of the HBS property

We are now prepared to rigorously define what it means for an $N \times N$ matrix A to be *hierarchically block separable* with respect to a given binary tree $\mathcal{T}$ that partitions the index vector $J = [1, 2, \ldots, N]$. For simplicity, we suppose that the tree has L fully populated levels, and that for every leaf node τ, the index vector I_τ holds precisely n points, so that $N = n\,2^L$. Then A is HBS with block rank k if the following two conditions hold:

(1) Assumption on ranks of off-diagonal blocks at the finest level: For any two distinct leaf nodes τ and τ', define the $n \times n$ matrix

$$\mathsf{A}_{\tau,\tau'} = \mathsf{A}(I_\tau, I_{\tau'}). \tag{2.32}$$

Then there must exist matrices U_τ, $\mathsf{V}_{\tau'}$, and $\tilde{\mathsf{A}}_{\tau,\tau'}$ such that

$$\begin{array}{ccccccc} \mathsf{A}_{\tau,\tau'} & = & \mathsf{U}_\tau & & \tilde{\mathsf{A}}_{\tau,\tau'} & & \mathsf{V}^*_{\tau'}. \\ n \times n & & n \times k & & k \times k & & k \times n \end{array} \tag{2.33}$$

(2) Assumption on ranks of off-diagonal blocks on level $\ell = L-1, L-2, \ldots, 1$: The rank assumption at level ℓ is defined in terms of the blocks constructed on the next finer level $\ell+1$: For any distinct nodes τ and τ' on level ℓ with children σ_1, σ_2 and σ'_1, σ'_2, respectively, define

$$\mathsf{A}_{\tau,\tau'} = \begin{bmatrix} \tilde{\mathsf{A}}_{\sigma_1,\sigma'_1} & \tilde{\mathsf{A}}_{\sigma_1,\sigma'_2} \\ \tilde{\mathsf{A}}_{\sigma_2,\sigma'_1} & \tilde{\mathsf{A}}_{\sigma_2,\sigma'_2} \end{bmatrix}. \tag{2.34}$$

	Name:	Size:	Function:
For each leaf node τ:	D_τ	$n \times n$	The diagonal block $\mathsf{A}(I_\tau, I_\tau)$.
	U_τ	$n \times k$	Basis for the columns in the blocks in row τ.
	V_τ	$n \times k$	Basis for the rows in the blocks in column τ.
For each parent node τ:	B_τ	$2k \times 2k$	Interactions between the children of τ.
	U_τ	$2k \times k$	Basis for the columns in the (reduced) blocks in row τ.
	V_τ	$2k \times k$	Basis for the rows in the (reduced) blocks in column τ.

Figure 2.3: An HBS matrix A associated with a tree $\mathcal{T}$ is fully specified if the factors listed above are provided.

Then there must exist matrices U_τ, $\mathsf{V}_{\tau'}$, and $\tilde{\mathsf{A}}_{\tau,\tau'}$ such that

$$\begin{array}{ccccccc} \mathsf{A}_{\tau,\tau'} & = & \mathsf{U}_\tau & \tilde{\mathsf{A}}_{\tau,\tau'} & \mathsf{V}_{\tau'}^*. \\ 2k \times 2k & & 2k \times k & k \times k & k \times 2k \end{array} \tag{2.35}$$

The two points above complete the definition. An HBS matrix is now fully described if the basis matrices U_τ and V_τ are provided for each node τ, and in addition, we are for each leaf τ given the $n \times n$ matrix

$$\mathsf{D}_\tau = \mathsf{A}(I_\tau, I_\tau), \tag{2.36}$$

and for each parent node τ with children σ_1 and σ_2 we are given the $2k \times 2k$ matrix

$$\mathsf{B}_\tau = \begin{bmatrix} 0 & \tilde{\mathsf{A}}_{\sigma_1,\sigma_2} \\ \tilde{\mathsf{A}}_{\sigma_2,\sigma_1} & 0 \end{bmatrix}. \tag{2.37}$$

Observe in particular that the matrices $\tilde{\mathsf{A}}_{\sigma_1,\sigma_2}$ are only required when $\{\sigma_1, \sigma_2\}$ forms a sibling pair. Figure 2.3 summarizes the required matrices.

Remark 10. *The definition of the HBS property given in this section is flexible in the sense that we do not enforce any conditions on the factors U_τ, V_τ, and $\tilde{\mathsf{A}}_{\tau,\tau'}$ other than that (2.33) and (2.35) must hold. For purposes of numerical stability, further conditions are sometimes imposed. The perhaps strongest such condition is to require the matrices U_τ and $\mathsf{V}_{\tau'}$ in (2.33) and (2.35) be orthonormal, see e.g. [71, 16] (one may in this case require that the matrices $\tilde{\mathsf{A}}_{\tau,\tau'}$ be diagonal, so that (2.33) and (2.35) become singular value decompositions.) If a "general" HBS matrix is given, it can easily be converted to this more restrictive format via, e.g., Algorithm 1. A choice that we*

have found highly convenient is to require (2.33) and (2.35) to be interpolatory decompositions (see Section 2.1.2). Then every U_τ *and* $\mathsf{V}_{\tau'}$ *contains a* $k \times k$ *identity matrix (which greatly accelerates computations), and each* $\tilde{\mathsf{A}}_{\tau,\tau'}$ *is a submatrix of the original matrix* A.

Remark 11. *The definition (2.33) transparently describes the functions of the basis matrices* U_τ *and* V_τ *whenever* τ *is a leaf node. The definition (2.35) of basis matrices for non-leaf nodes is perhaps less intuitive. Their meaning can be made clearer by defining what we call "extended" basis matrices* $\mathsf{U}_\tau^{\text{extend}}$ *and* $\mathsf{V}_\tau^{\text{extend}}$*. For a leaf node* τ *we simply set*

$$\mathsf{U}_\tau^{\text{extend}} = \mathsf{U}_\tau, \qquad \textit{and} \qquad \mathsf{V}_\tau^{\text{extend}} = \mathsf{V}_\tau.$$

For a parent node τ *with children* σ_1 *and* σ_2*, we set*

$$\mathsf{U}_\tau^{\text{extend}} = \begin{bmatrix} \mathsf{U}_{\sigma_1}^{\text{extend}} & 0 \\ 0 & \mathsf{U}_{\sigma_2}^{\text{extend}} \end{bmatrix} \mathsf{U}_\tau, \qquad \textit{and} \qquad \mathsf{V}_\tau^{\text{extend}} = \begin{bmatrix} \mathsf{V}_{\sigma_1}^{\text{extend}} & 0 \\ 0 & \mathsf{V}_{\sigma_2}^{\text{extend}} \end{bmatrix} \mathsf{V}_\tau.$$

Then for any distinct nodes τ *and* τ' *on level* ℓ*, we have*

$$\begin{array}{ccccccc} \mathsf{A}(I_\tau, I_{\tau'}) & = & \mathsf{U}_\tau^{\text{extend}} & \tilde{\mathsf{A}}_{\tau,\tau'} & (\mathsf{V}_\tau^{\text{extend}})^*. \\ n\,2^{L-\ell} \times n\,2^{L-\ell} & & n\,2^{L-\ell} \times k & k \times k & k \times n\,2^{L-\ell} \end{array}$$

2.3.4 Telescoping factorizations

In the heuristic description of the HBS property in Section 2.3.1, we claimed that any HBS matrix can be expressed as a telescoping factorization with block diagonal factors. We will now formalize this claim. Given the matrices defined in Section 2.3.3 (and summarized in Figure 2.3), we define the following block diagonal factors:

$$\underline{\mathsf{D}}^{(\ell)} = \text{diag}(\mathsf{D}_\tau : \tau \text{ is a box on level } \ell), \qquad \ell = 0,\, 1,\, \ldots,\, L, \tag{2.38}$$

$$\underline{\mathsf{U}}^{(\ell)} = \text{diag}(\mathsf{U}_\tau : \tau \text{ is a box on level } \ell), \qquad \ell = 1,\, 2,\, \ldots,\, L, \tag{2.39}$$

$$\underline{\mathsf{V}}^{(\ell)} = \text{diag}(\mathsf{V}_\tau : \tau \text{ is a box on level } \ell), \qquad \ell = 1,\, 2,\, \ldots,\, L, \tag{2.40}$$

$$\underline{\mathsf{B}}^{(\ell)} = \text{diag}(\mathsf{B}_\tau : \tau \text{ is a box on level } \ell), \qquad \ell = 0,\, 1,\, \ldots,\, L-1,. \tag{2.41}$$

ALGORITHM 1 (reformatting an HBS matrix)

Given the factors U_τ, V_τ, $\tilde{\mathsf{A}}_{\sigma_1,\sigma_2}$, D_τ *of an HBS matrix in general format, this algorithm computes new factors (that overwrite the old) such that all* U_τ *and* V_τ *are orthonormal, and all* $\tilde{\mathsf{A}}_{\sigma_1,\sigma_2}$ *are diagonal.*

loop over levels, finer to coarser, $\ell = L-1, L-2, \ldots, 0$
  loop over all parent boxes τ on level ℓ,
    Let σ_1 and σ_2 denote the children of τ.

$[\mathsf{U}_1, \mathsf{R}_1] = \mathtt{qr}(\mathsf{U}_{\sigma_1})$.	$[\mathsf{U}_2, \mathsf{R}_2] = \mathtt{qr}(\mathsf{U}_{\sigma_2})$.
$[\mathsf{V}_1, \mathsf{S}_1] = \mathtt{qr}(\mathsf{V}_{\sigma_1})$.	$[\mathsf{V}_2, \mathsf{S}_2] = \mathtt{qr}(\mathsf{V}_{\sigma_2})$.
$[\mathsf{X}_1, \Sigma_{12}, \mathsf{Y}_2] = \mathtt{svd}(\mathsf{R}_1\, \tilde{\mathsf{A}}_{\sigma_1,\sigma_2}\, \mathsf{S}_2^*)$.	$[\mathsf{X}_2, \Sigma_{21}, \mathsf{Y}_1] = \mathtt{svd}(\mathsf{R}_2\, \tilde{\mathsf{A}}_{\sigma_2,\sigma_1}\, \mathsf{S}_1^*)$.
$\mathsf{U}_{\sigma_1} \leftarrow \mathsf{U}_1\, \mathsf{X}_1$.	$\mathsf{U}_{\sigma_2} \leftarrow \mathsf{U}_2\, \mathsf{X}_2$.
$\mathsf{V}_{\sigma_1} \leftarrow \mathsf{V}_1\, \mathsf{Y}_1$.	$\mathsf{V}_{\sigma_2} \leftarrow \mathsf{V}_2\, \mathsf{Y}_2$.
$\mathsf{B}_{\sigma_1\sigma_2} \leftarrow \Sigma_{12}$.	$\mathsf{B}_{\sigma_2\sigma_1} \leftarrow \Sigma_{21}$.

    if $l > 0$

$$\mathsf{U}_\tau \leftarrow \begin{bmatrix} \mathsf{X}_1^*\, \mathsf{R}_1 & 0 \\ 0 & \mathsf{X}_2^*\, \mathsf{R}_2 \end{bmatrix} \mathsf{U}_\tau. \qquad \mathsf{V}_\tau \leftarrow \begin{bmatrix} \mathsf{Y}_1^*\, \mathsf{S}_1 & 0 \\ 0 & \mathsf{Y}_2^*\, \mathsf{S}_2 \end{bmatrix} \mathsf{V}_\tau.$$

    end if
  end loop
end loop

Furthermore, we let $\tilde{\mathsf{A}}^{(\ell)}$ denote the block matrix whose diagonal blocks are zero, and whose off-diagonal blocks are the blocks $\tilde{\mathsf{A}}_{\tau,\tau'}$ for all distinct τ, τ' on level ℓ. With these definitions,

$$\begin{array}{ccccccc} \mathsf{A} & = & \underline{\mathsf{U}}^{(L)} & \tilde{\mathsf{A}}^{(L)} & (\underline{\mathsf{V}}^{(L)})^* & + & \underline{\mathsf{D}}^{(L)}; \\ n\,2^L \times n\,2^L & & n\,2^L \times k\,2^L & k\,2^L \times k\,2^L & k\,2^L \times n\,2^L & & n\,2^L \times n\,2^L \end{array} \tag{2.42}$$

for $\ell = L-1,\, L-2,\, \ldots,\, 1$ we have

$$\begin{array}{ccccccc} \tilde{\mathsf{A}}^{(\ell+1)} & = & \underline{\mathsf{U}}^{(\ell)} & \tilde{\mathsf{A}}^{(\ell)} & (\underline{\mathsf{V}}^{(\ell)})^* & + & \underline{\mathsf{B}}^{(\ell)}; \\ k\,2^{\ell+1} \times k\,2^{\ell+1} & & k\,2^{\ell+1} \times k\,2^{\ell} & k\,2^{\ell} \times k\,2^{\ell} & k\,2^{\ell} \times k\,2^{\ell+1} & & k\,2^{\ell+1} \times k\,2^{\ell+1} \end{array} \tag{2.43}$$

and finally

$$\tilde{\mathsf{A}}^{(1)} = \underline{\mathsf{B}}^{(0)}. \tag{2.44}$$

2.3.5 Matrix-vector multiplication

The telescoping factorizations in Section 2.3.4 easily translate into a formula for evaluating the matrix-vector product $\mathbf{u} = \mathsf{A}\,\mathbf{q}$ once all factors in an HBS representation have been provided. The resulting algorithm has computational complexity $O(N\,k)$ (assuming that $n = O(k)$), and is given as Algorithm 2.

2.4 Inversion of hierarchically block separable matrices

In this section, we describe an algorithm for inverting an HBS matrix A. The algorithm is exact (in the absence of round-off errors) and has asymptotic complexity $O(N\,k^2)$. It is summarized as Algorithm 3, and as the description indicates, it is very simple to implement. The output of Algorithm 3 is a set of factors in a data-sparse representation of A^{-1} which can be applied to a given vector via Algorithm 4. Technically, the scheme consists of recursive application of Lemma 1, and its derivation is given in the proof of the following theorem:

Theorem 1. *Let A be an invertible $N \times N$ HBS matrix with block-rank k. Suppose that at the finest level of the tree structure, there are 2^L leaves that each holds n points (so that $N = n\,2^L$), and that $n \leq 2\,k$. Then a data-sparse representation of A^{-1} that is exact up to rounding errors can*

ALGORITHM 2 (HBS matrix-vector multiply)

Given a vector $\mathbf{q}$ *and a matrix* A *in HBS format, compute* $\mathbf{u} = \mathsf{A}\,\mathbf{q}$.

loop over all leaf boxes τ
  $\hat{\mathbf{q}}_\tau = \mathsf{V}_\tau^*\,\mathbf{q}(I_\tau)$.
end loop

loop over levels, finer to coarser, $\ell = L-1,\ L-2,\ \ldots, 1$
  loop over all parent boxes τ on level ℓ,
    Let σ_1 and σ_2 denote the children of τ.
    $\hat{\mathbf{q}}_\tau = \mathsf{V}_\tau^* \begin{bmatrix} \hat{\mathbf{q}}_{\sigma_1} \\ \hat{\mathbf{q}}_{\sigma_2} \end{bmatrix}$.
  end loop
end loop

$\hat{\mathbf{u}}_1 = 0$
loop over all levels, coarser to finer, $\ell = 1,\ 2,\ \ldots,\ L-1$
  loop over all parent boxes τ on level ℓ
    Let σ_1 and σ_2 denote the children of τ.
    $\begin{bmatrix} \hat{\mathbf{u}}_{\sigma_1} \\ \hat{\mathbf{u}}_{\sigma_2} \end{bmatrix} = \mathsf{U}_\tau\,\hat{\mathbf{u}}_\tau + \begin{bmatrix} 0 & \mathsf{B}_{\sigma_1,\sigma_2} \\ \mathsf{B}_{\sigma_2,\sigma_1} & 0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{q}}_{\sigma_1} \\ \hat{\mathbf{q}}_{\sigma_2} \end{bmatrix}$.
  end loop
end loop

loop over all leaf boxes τ
  $\mathbf{u}(I_\tau) = \mathsf{U}_\tau\,\hat{\mathbf{u}}_\tau + \mathsf{D}_\tau\,\mathbf{q}(I_\tau)$.
end loop

be computed in $O(N\,k^2)$ operations via a process given as Algorithm 3, provided that none of the matrices that need to be inverted is singular. The computed inverse can be applied to a vector in $O(N\,k)$ operations via Algorithm 4.

Proof: We can according to equation (2.42) express an HBS matrix as

$$\mathsf{A} = \underline{\mathsf{U}}^{(L)}\,\tilde{\mathsf{A}}^{(L)}\,(\underline{\mathsf{V}}^{(L)})^* + \underline{\mathsf{D}}^{(L)}.$$

Lemma 1 immediately applies, and we find that

$$\mathsf{A}^{-1} = \underline{\mathsf{E}}^{(L)}\left(\tilde{\mathsf{A}}^{(L)} + \hat{\underline{\mathsf{D}}}^{(L)}\right)^{-1}(\underline{\mathsf{F}}^{(L)})^* + \underline{\mathsf{G}}^{(L)}, \tag{2.45}$$

where $\underline{\mathsf{E}}^{(L)}$, $\underline{\mathsf{F}}^{(L)}$, $\hat{\underline{\mathsf{D}}}^{(L)}$ and $\underline{\mathsf{G}}^{(L)}$ are defined via (2.12), (2.13), (2.14), and (2.15).

To move to the next coarser level, we set $\ell = L-1$ in formula (2.43) whence

$$\tilde{\mathsf{A}}^{(L)} + \hat{\underline{\mathsf{D}}}^{(L)} = \underline{\mathsf{U}}^{(L-1)}\,\tilde{\mathsf{A}}^{(L-1)}\,(\underline{\mathsf{V}}^{(L-1)})^* + \underline{\mathsf{B}}^{(L-1)} + \hat{\underline{\mathsf{D}}}^{(L)}. \tag{2.46}$$

We define

$$\tilde{\underline{\mathsf{D}}}^{(L-1)} = \underline{\mathsf{B}}^{(L-1)} + \hat{\underline{\mathsf{D}}}^{(L)} = \left[\begin{array}{cc|cc|cc|c} \hat{\mathsf{D}}_{\tau_1} & \mathsf{B}_{\tau_1\tau_2} & 0 & 0 & 0 & 0 & \cdots \\ \mathsf{B}_{\tau_1\tau_2} & \hat{\mathsf{D}}_{\tau_2} & 0 & 0 & 0 & 0 & \cdots \\ \hline 0 & 0 & \hat{\mathsf{D}}_{\tau_3} & \mathsf{B}_{\tau_3\tau_4} & 0 & 0 & \cdots \\ 0 & 0 & \mathsf{B}_{\tau_3\tau_4} & \hat{\mathsf{D}}_{\tau_4} & 0 & 0 & \cdots \\ \hline 0 & 0 & 0 & 0 & \hat{\mathsf{D}}_{\tau_5} & \mathsf{B}_{\tau_5\tau_6} & \cdots \\ 0 & 0 & 0 & 0 & \mathsf{B}_{\tau_5\tau_6} & \hat{\mathsf{D}}_{\tau_6} & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{array}\right],$$

where $\{\tau_1,\,\tau_2,\,\ldots,\,\tau_{2^L}\}$ is a list of the boxes on level L. Equation (2.46) then takes the form

$$\tilde{\mathsf{A}}^{(L)} + \hat{\underline{\mathsf{D}}}^{(L)} = \underline{\mathsf{U}}^{(L-1)}\,\tilde{\mathsf{A}}^{(L-1)}\,(\underline{\mathsf{V}}^{(L-1)})^* + \tilde{\underline{\mathsf{D}}}^{(L-1)}. \tag{2.47}$$

(We note that in (2.47), the terms on the left hand side are block matrices with $2^L \times 2^L$ blocks, each of size $k \times k$, whereas the terms on the right hand side are block matrices with $2^{L-1} \times 2^{L-1}$ blocks, each of size $2\,k \times 2\,k$.) Now Lemma 1 applies to (2.47) and we find that

$$\left(\tilde{\mathsf{A}}^{(L)} + \hat{\underline{\mathsf{D}}}^{(L)}\right)^{-1} = \underline{\mathsf{E}}^{(L-1)}\left(\tilde{\mathsf{A}}^{(L-1)} + \hat{\underline{\mathsf{D}}}^{(L-1)}\right)^{-1}(\underline{\mathsf{F}}^{(L-1)})^* + \underline{\mathsf{G}}^{(L-1)},$$

where $\underline{\mathsf{E}}^{(L-1)}$, $\underline{\mathsf{F}}^{(L-1)}$, $\underline{\hat{\mathsf{D}}}^{(L-1)}$ and $\underline{\mathsf{G}}^{(L-1)}$ are defined via (2.12), (2.13), (2.14), and (2.15).

The process by which we went from step L to step $L-1$ is then repeated to move up to coarser and coarser levels. With each step, the size of the matrix to be inverted is cut in half. Once we get to the top level, we are left with the task of inverting the matrix

$$\tilde{\mathsf{A}}^{(1)} + \underline{\hat{\mathsf{D}}}^{(1)} = \begin{bmatrix} \hat{\mathsf{D}}_2 & \mathsf{B}_{2,3} \\ \mathsf{B}_{3,2} & \hat{\mathsf{D}}_3 \end{bmatrix} \tag{2.48}$$

The matrix in (2.48) is of size $2\,k \times 2\,k$, and we use brute force to evaluate

$$\mathsf{G}^{(0)} = \mathsf{G}_1 = \begin{bmatrix} \hat{\mathsf{D}}_2 & \mathsf{B}_{2,3} \\ \mathsf{B}_{3,2} & \hat{\mathsf{D}}_3 \end{bmatrix}^{-1}.$$

To calculate the cost of the inversion scheme described, we note that in order to compute the matrices $\underline{\mathsf{E}}^{(\ell)}$, $\underline{\mathsf{F}}^{(\ell)}$, $\underline{\mathsf{G}}^{(\ell)}$, and $\underline{\hat{\mathsf{D}}}^{(\ell)}$ on level ℓ, we need to perform dense matrix operations on 2^ℓ blocks, each of size at most $2\,k \times 2\,k$. Since the leaf boxes each hold at most $2\,k$ points, so that $2^{L+1}\,k \leq N$, the total cost is

$$\text{COST} \sim \sum_{\ell=1}^{L} 2^\ell\, 8\, k^3 \sim 2^{L+4}\, k^3 \sim N\, k^2.$$

This completes the proof. □

Remark 12. *Algorithm 3 produces a representation of* A^{-1} *that is not exactly in HBS form since the matrices* G_τ *do not have zero diagonal blocks like the matrices* B_τ*, cf. (2.37). However, a simple technique given as Algorithm 5 converts the factorization provided by Algorithm 3 into a standard HBS factorization. If a factorization in which the expansion matrices are all orthonormal is sought, then further post-processing via Algorithm 1 will do the job.*

Remark 13. *Algorithm 3 provides four formulas for the matrices* $\{\mathsf{E}_\tau, \mathsf{F}_\tau, \mathsf{G}_\tau, \hat{\mathsf{D}}_\tau\}_\tau$*. The task of actually computing the matrices can be accelerated in two ways: (1) Several of the matrix-matrix products in the formulas are recurring, and the computation should be organized so that each matrix-matrix product is evaluated only once. (2) When interpolatory decompositions are used, multiplications involving the matrices* U_τ *and* V_τ *can be accelerated by exploiting that they each contain a* $k \times k$ *identity matrix.*

ALGORITHM 3 (inversion of an HBS matrix)

loop over all levels, finer to coarser, $\ell = L,\, L-1,\, \ldots, 1$
  loop over all boxes τ on level ℓ,
    if τ is a leaf node
      $\tilde{\mathsf{D}}_\tau = \mathsf{D}_\tau$
    else
      Let σ_1 and σ_2 denote the children of τ.
      $\tilde{\mathsf{D}}_\tau = \begin{bmatrix} \hat{\mathsf{D}}_{\sigma_1} & \mathsf{B}_{\sigma_1,\sigma_2} \\ \mathsf{B}_{\sigma_2,\sigma_1} & \hat{\mathsf{D}}_{\sigma_2} \end{bmatrix}$
    end if
    $\hat{\mathsf{D}}_\tau = \left(\mathsf{V}_\tau^* \, \tilde{\mathsf{D}}_\tau^{-1} \, \mathsf{U}_\tau\right)^{-1}$.
    $\mathsf{E}_\tau = \tilde{\mathsf{D}}_\tau^{-1} \, \mathsf{U}_\tau \, \hat{\mathsf{D}}_\tau$.
    $\mathsf{F}_\tau^* = \hat{\mathsf{D}}_\tau \, \mathsf{V}_\tau^* \, \tilde{\mathsf{D}}_\tau^{-1}$.
    $\mathsf{G}_\tau = \hat{\mathsf{D}}_\tau - \tilde{\mathsf{D}}_\tau^{-1} \, \mathsf{U}_\tau \, \hat{\mathsf{D}}_\tau \, \mathsf{V}_\tau^* \, \tilde{\mathsf{D}}_\tau^{-1}$.
  end loop
end loop
$\mathsf{G}_1 = \begin{bmatrix} \hat{\mathsf{D}}_2 & \mathsf{B}_{2,3} \\ \mathsf{B}_{3,2} & \hat{\mathsf{D}}_3 \end{bmatrix}^{-1}$.

ALGORITHM 4 (application of inverse)

Given a vector $\mathbf{u}$*, compute* $\mathbf{q} = \mathsf{A}^{-1}\,\mathbf{u}$ *using the compressed representation of* A^{-1} *resulting from Algorithm 3.*

loop over all leaf boxes τ
  $\hat{\mathbf{u}}_\tau = \mathsf{F}_\tau^*\,\mathbf{u}(I_\tau)$.
end loop

loop over all levels, finer to coarser, $\ell = L,\ L-1,\ \ldots, 1$
  loop over all parent boxes τ on level ℓ,
    Let σ_1 and σ_2 denote the children of τ.
    $\hat{\mathbf{u}}_\tau = \mathsf{F}_\tau^* \begin{bmatrix} \hat{\mathbf{u}}_{\sigma_1} \\ \hat{\mathbf{u}}_{\sigma_2} \end{bmatrix}$.
  end loop
end loop

$$\begin{bmatrix} \hat{\mathbf{q}}_2 \\ \hat{\mathbf{q}}_3 \end{bmatrix} = \hat{\mathsf{G}}_1 \begin{bmatrix} \hat{\mathbf{u}}_2 \\ \hat{\mathbf{u}}_3 \end{bmatrix}.$$

loop over all levels, coarser to finer, $\ell = 1,\ 2,\ \ldots,\ L-1$
  loop over all parent boxes τ on level ℓ
    Let σ_1 and σ_2 denote the children of τ.
    $\begin{bmatrix} \hat{\mathbf{q}}_{\sigma_1} \\ \hat{\mathbf{q}}_{\sigma_2} \end{bmatrix} = \mathsf{E}_\tau\,\hat{\mathbf{u}}_\tau + \mathsf{G}_\tau \begin{bmatrix} \hat{\mathbf{u}}_{\sigma_1} \\ \hat{\mathbf{u}}_{\sigma_2} \end{bmatrix}$.
  end loop
end loop

loop over all leaf boxes τ
  $\mathbf{q}(I_\tau) = \mathsf{E}_\tau\,\hat{\mathbf{q}}_\tau + \mathsf{G}_\tau\,\mathbf{u}(I_\tau)$.
end loop

Remark 14. *The assumption in Theorem 1 that none of the matrices to be inverted is singular is undesirable. When* A *is spd, this assumption can be done away with (cf. Corollary 1), and it can further be proved that the inversion process is numerically stable. When* A *is non-symmetric, the intermediate matrices often become ill-conditioned. We have empirically observed that if we enforce that* $\mathsf{U}_\tau = \mathsf{V}_\tau$ *for every node (the procedure for doing this for non-symmetric matrices is described in Remark 16), then the method is remarkably stable, but we do not have any supporting theory.*

Remark 15. *The assumption in Theorem 1 that the block-rank* k *remains constant across all levels is slightly unnatural. In applications to integral equations on 1D domain, one often finds that the rank of interaction depends logarithmically on the number of points in a block. It is shown in [61] that inclusion of such logarithmic growth of the interaction ranks does not change the* $O(N)$ *total complexity.*

2.5 Computing the HBS representation of a boundary integral operator

Section 2.3 describes a particular way of representing a class of "compressible" matrices in a hierarchical structure of block-diagonal matrices. For any matrix whose HBS rank is k, these factors can via straight-forward means be computed in $O(N^2\,k)$ operations. In this section, we describe an $O(N\,k^2)$ technique for computing an HBS representation of the matrix resulting upon Nyström discretization of a BIE.

2.5.1 A basic compression scheme for leaf nodes

In this section, we describe how to construct for every leaf node τ, interpolatory matrices U_τ and V_τ of rank k, and index vectors $\tilde{I}_\tau^{(\mathrm{row})}$ and $\tilde{I}_\tau^{(\mathrm{col})}$ such that, cf. (2.33),

$$\mathsf{A}(I_\tau, I_{\tau'}) = \mathsf{U}_\tau\,\mathsf{A}(\tilde{I}_\tau^{(\mathrm{row})},\,\tilde{I}_{\tau'}^{(\mathrm{col})})\,\mathsf{V}_{\tau'}^*, \qquad \tau \neq \tau'. \tag{2.49}$$

The first step in the process is to construct for every leaf τ a row of blocks R_τ and a column of blocks C_τ via

$$\mathsf{R}_\tau = \mathsf{A}(I_\tau,\,L_\tau), \qquad \text{and} \qquad \mathsf{C}_\tau = \mathsf{A}(L_\tau,\,I_\tau),$$

ALGORITHM 5 (reformatting inverse)

Postprocessing the terms computed Algorithm 3 to obtain an inverse in the standard HBS format.

loop over all levels, coarser to finer, $\ell = 0,\ 1,\ 2,\ \ldots, L-1$
  loop over all parent boxes τ on level ℓ,
    Let σ_1 and σ_2 denote the children of τ.
    Define the matrices $\mathsf{H}_{1,1}$, $\mathsf{B}_{\sigma_1,\sigma_2}$, $\mathsf{B}_{\sigma_2,\sigma_1}$, $\mathsf{H}_{2,2}$ so that
    $\mathsf{G}_\tau = \begin{bmatrix} \mathsf{H}_{1,1} & \mathsf{B}_{\sigma_1,\sigma_2} \\ \mathsf{B}_{\sigma_2,\sigma_1} & \mathsf{H}_{2,2} \end{bmatrix}$.
    $\mathsf{G}_{\sigma_1} \leftarrow \mathsf{G}_{\sigma_1} + \mathsf{E}_{\sigma_1}\,\mathsf{H}_{1,1}\,\mathsf{F}^*_{\sigma_1}$.
    $\mathsf{G}_{\sigma_2} \leftarrow \mathsf{G}_{\sigma_2} + \mathsf{E}_{\sigma_2}\,\mathsf{H}_{2,2}\,\mathsf{F}^*_{\sigma_2}$.
  end loop
end loop

loop over all leaf boxes τ
  $\mathsf{D}_\tau = \mathsf{G}_\tau$.
end loop

where L_τ the complement of the index vector I_τ within the full index set,

$$L_\tau = \{1,\, 2,\, 3,\, \ldots,\, N\} \backslash I_\tau.$$

The condition (2.33) implies that R_τ and C_τ each have rank at most k. We can therefore construct interpolatory decompositions

$$\mathsf{R}_\tau = \quad \mathsf{U}_\tau \, \mathsf{R}(J_\tau^{(\mathrm{row})}, :\,), \tag{2.50}$$

$$\mathsf{C}_\tau = \quad \mathsf{C}(:\,, J_\tau^{(\mathrm{col})})\, \mathsf{V}_\tau^*. \tag{2.51}$$

Now (2.49) holds if we set

$$\tilde{I}_\tau^{(\mathrm{row})} = I_\tau(J_\tau^{(\mathrm{row})}) \qquad \text{and} \qquad \tilde{I}_\tau^{(\mathrm{col})} = I_\tau(J_\tau^{(\mathrm{col})}).$$

Remark 16. *It is often useful to use the same basis matrices to span the ranges of both R_τ and C_τ^*. In particular, this can make a substantial difference in the stability of the inversion scheme described in Section 2.3. To accomplish this, form the matrix $\mathsf{X}_\tau = \begin{bmatrix}\mathsf{R}_\tau \mid \mathsf{C}_\tau^*\end{bmatrix}$ and then compute an interpolatory factorization*

$$\mathsf{X}_\tau = \mathsf{U}_\tau \, \mathsf{X}_\tau(J_\tau, :\,). \tag{2.52}$$

Then clearly $\mathsf{R}_\tau = \mathsf{U}_\tau \, \mathsf{R}(J_\tau, :\,)$ and $\mathsf{C}_\tau = \mathsf{C}(:\,, J_\tau)\, \mathsf{U}_\tau^*$. Enforcing symmetry may slightly increase the HSS-ranks, but typically in a very modest way.

Remark 17. *In applications, the matrices R_τ and C_τ are typically only approximately of rank k and the factorizations (2.50) and (2.51) are then required to hold only to within some preset tolerance ε. Techniques for computing rank-revealing partial factorizations of this type are described in detail in [46, 21].*

2.5.2 An accelerated compression scheme for leaf nodes

We will in this section demonstrate how to rapidly construct matrices U_τ, V_τ and index vectors $J_\tau^{(\mathrm{row})}$, $J_\tau^{(\mathrm{col})}$ such that (2.50) and (2.51) hold. We focus on the construction of U_τ and $J_\tau^{(\mathrm{row})}$ since the construction of V_τ and $J_\tau^{(\mathrm{col})}$ is analogous. While U_τ and $J_\tau^{(\mathrm{row})}$ can in principle

be constructed by directly computing an interpolatory decomposition of R_τ, this is in practice prohibitively expensive since R_τ is very large. The way to get around this is to exploit analytic properties of the kernel to construct a much smaller matrix $\mathsf{R}_\tau^{(\text{small})}$ whose columns span the column space of R_τ. Then we can cheaply form U_τ and $J_\tau^{(\text{row})}$ by factoring this smaller matrix. The process typically over-estimates the rank slightly (since the columns of $\mathsf{R}_\tau^{(\text{small})}$ will span a slightly larger space than the columns of R_τ) but this is more than compensated by the dramatic acceleration of the compression step.

To formalize matters, our goal is to construct a small matrix $\mathsf{R}_\tau^{(\text{small})}$ such that

$$\text{Ran}\big(\mathsf{R}_\tau\big) \subseteq \text{Ran}\big(\mathsf{R}_\tau^{(\text{small})}\big). \tag{2.53}$$

Then all we would need to do to compress τ is to construct the interpolatory decomposition

$$\mathsf{R}_\tau^{(\text{small})} = \mathsf{U}_\tau \, \mathsf{R}_\tau^{(\text{small})}(J_\tau^{(\text{row})}, :) \tag{2.54}$$

since (2.53) and (2.54) together imply (2.50).

When constructing the matrix $\mathsf{R}_\tau^{(\text{small})}$, we distinguish between near field interaction and far field interactions. The near field interactions cannot readily be compressed, but this is not a problem since they contribute a very small part of R_τ. To define what we mean by "near" and "far," we need to introduce some notation. Let Γ_τ denote the segment of Γ associated with the node τ, see Fig. 6.1. We enclose Γ_τ in a circle and then let $\Gamma_\tau^{(\text{proxy})}$ denote a circle with the same center but with a 50% larger radius. We now define $\Gamma_\tau^{(\text{far})}$ as the part of Γ outside of $\Gamma_\tau^{(\text{proxy})}$ and define $\Gamma_\tau^{(\text{near})}$ as the part of Γ inside $\Gamma_\tau^{(\text{proxy})}$ but disjoint from Γ_τ. In other words

$$\Gamma = \Gamma_\tau \cup \Gamma_\tau^{(\text{near})} \cup \Gamma_\tau^{(\text{far})}$$

forms a disjoint partitioning of Γ. We define $L_\tau^{(\text{near})}$ and $L_\tau^{(\text{far})}$ so that

$$\{1,\, 2,\, 3,\, \ldots,\, N\} = I_\tau \cup L_\tau^{(\text{near})} \cup L_\tau^{(\text{far})}$$

forms an analogous disjoint partitioning of the index set $\{1,\, 2,\, \ldots,\, N\}$. We now find that

$$\mathsf{R}_\tau = \Big[\mathsf{A}(I_\tau, L_\tau^{(\text{near})}) \mid \mathsf{A}(I_\tau, L_\tau^{(\text{far})})\Big] \, \Pi \tag{2.55}$$

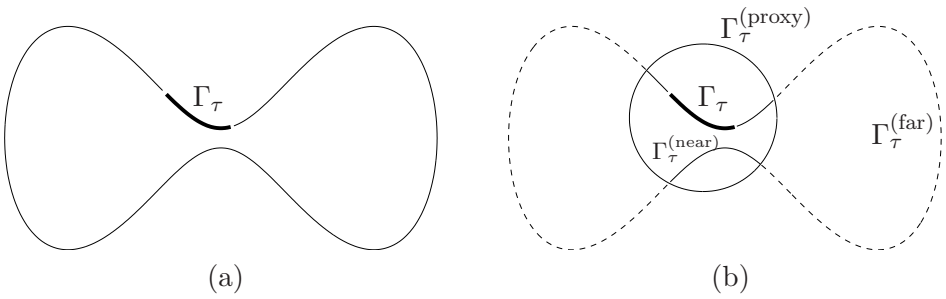


Figure 2.4: A contour Γ. (a) Γ_τ is drawn with a bold line. (b) The contour $\Gamma_\tau^{(\text{near})}$ is drawn with a thin solid line and $\Gamma_\tau^{(\text{far})}$ with a dashed line.

where Π is a permutation matrix. We will construct a matrix $\mathsf{R}_\tau^{(\text{proxy})}$ such that

$$\text{Ran}\big(\mathsf{A}(I_\tau, L_\tau^{(\text{far})})\big) \subseteq \text{Ran}\big(\mathsf{R}_\tau^{(\text{proxy})}\big), \tag{2.56}$$

and then we set

$$\mathsf{R}_\tau^{(\text{small})} = \Big[\mathsf{A}(I_\tau, L_\tau^{(\text{near})}) \mid \mathsf{R}_\tau^{(\text{proxy})}\Big]. \tag{2.57}$$

That (2.53) holds is now a consequence of (2.55), (2.56) and (2.57).

All that remains is to construct a small matrix $\mathsf{R}_\tau^{(\text{proxy})}$ such that (2.56) holds. We describe the process for the single layer potential associated with Laplace's equation (for generalizations, see Remarks 19, 20, 21, and 22). Since we use Nyström discretization, the matrix $\mathsf{A}(I_\tau, L_\tau^{(\text{far})})$ in this case represents evaluation on Γ_τ of the harmonic potential generated by a set of point charges on $\Gamma_\tau^{(\text{far})}$. We know from potential theory that any harmonic field generated by charges *outside* $\Gamma_\tau^{(\text{proxy})}$ can be generated by placing an equivalent charge distribution *on* $\Gamma_\tau^{(\text{proxy})}$. Since we only need to capture the field to within some preset precision ε, it is sufficient to place charges on a finite collection $\{\mathbf{z}_j\}_{j=1}^J$ of points on $\Gamma_\tau^{(\text{proxy})}$. In other words, we set

$$\mathsf{R}_\tau^{(\text{proxy})}(i,j) = \log|\mathbf{x}_i - \mathbf{z}_j|, \qquad i \in I_\tau,\ j \in \{1,\, 2,\, 3,\, \ldots,\, J\}.$$

The number of charges J that are needed on the external circle depends on the precision ε required. In fact $J = O(\log(1/\varepsilon))$ as $\varepsilon \to 0$. We have found that using $J = 50$ points is sufficient to attain ten digits of accuracy or better.

The construction of a small matrix $\mathsf{C}_\tau^{(\text{small})}$ that can be used to construct V_τ and $J_\tau^{(\text{col})}$ such that (2.51) holds is entirely analogous since the matrix C_τ^* is also a map of a charge distribution on $\Gamma_\tau^{(\text{far})}$ to a potential field on Γ_τ. The only caveat is that the rows of C_τ^* must be scaled by the quadrature weights used in the Nyström method.

Remark 18. *As an alternative to placing charges on the exterior circle, one could represent the harmonic field generated on* Γ_τ *by an expansion in the cylindrical harmonic functions* $\{r^j \cos(j\theta),\, r^j \sin(j\theta)\}_{j=0}^{J}$ *where* (r, θ) *are the polar coordinates of a point in the circle enclosing* Γ_τ*. The number of functions needed is about the same, but we found it easier to correctly weigh the two terms* $\mathsf{A}(I_\tau, L_\tau^{(\text{far})})$ *and* $\mathsf{R}_\tau^{(\text{proxy})}$ *when using proxy charges on the outer circle.*

Remark 19 (Extension to the double layer kernel)**.** *The procedure described directly generalizes to the double layer kernel associated with Laplace's equation. The compression of* R_τ *is exactly the same. The compression of* C_τ^* *is very similar, but now the target field is the normal derivative of the set of harmonic potentials that can be generated by sources outside* $\Gamma_\tau^{(\text{proxy})}$*.*

Remark 20 (Extension to Laplace's equation in $\mathbb{R}^3$)**.** *The scheme described generalizes immediately to BIEs defined on surfaces in* $\mathbb{R}^3$*. The circles must be replaced by spheres, and the complexity is no longer linear, but the method remains competitive at low and intermediate accuracies [43].*

Remark 21 (Extension to Helmholtz and other equations)**.** *The scheme described has been generalized to the single and double layer potentials associated with Helmholtz equation, see [61]. The only complication happens when the frequency is close to a resonant frequency of the proxy circle. This potential problem is avoided by placing both monopoles and dipoles on the proxy circle.*

Remark 22 (Extension to integral equations on the line)**.** *The acceleration gets particularly simple for integral equations on a line with smooth non-oscillatory kernels. In this case, the range of* $\mathsf{A}(I_\tau, L_\tau^{(\text{far})})$ *can typically be represented by a short expansion in a generic set of basis functions such as, e.g., Legendre polynomials.*

2.5.3 Compression of higher levels

The method described in Section 2.5.2 rapidly constructs the matrices U_τ, V_τ and the index vector $J_\tau^{(\text{row})}$, $J_\tau^{(\text{col})}$ for any leaf node τ. Using the notation introduced in Section 2.3.4, it computes the matrices $\underline{\mathsf{U}}^{(L)}$, $\underline{\mathsf{V}}^{(L)}$, and $\tilde{\mathsf{A}}^{(L)}$. It is important to note that when the interpolatory decomposition is used, $\tilde{\mathsf{A}}^{(L)}$ is in fact a submatrix of A, and is represented *implicitly* by specifying the relevant index vectors. To be precise, if τ and τ' are two nodes on level $L-1$, with children σ_1, σ_2 and σ_1', σ_2', respectively, then

$$\mathsf{A}_{\tau,\tau'} = \begin{bmatrix} \mathsf{A}(\tilde{I}_{\sigma_1}^{(\text{row})}, \tilde{I}_{\sigma_1'}^{(\text{col})}) & \mathsf{A}(\tilde{I}_{\sigma_1}^{(\text{row})}, \tilde{I}_{\sigma_2'}^{(\text{col})}) \\ \mathsf{A}(\tilde{I}_{\sigma_2}^{(\text{row})}, \tilde{I}_{\sigma_1'}^{(\text{col})}) & \mathsf{A}(\tilde{I}_{\sigma_2}^{(\text{row})}, \tilde{I}_{\sigma_2'}^{(\text{col})}) \end{bmatrix}.$$

The observation that $\tilde{\mathsf{A}}^{(L)}$ is a submatrix of A is critical. It implies that the matrices $\underline{\mathsf{U}}^{(L-1)}$, $\underline{\mathsf{V}}^{(L-1)}$, and $\tilde{\mathsf{A}}^{(L-1)}$ can be computed using the strategy of Section 2.5.2 *without any modifications.*

2.5.4 Approximation errors

As mentioned in Remark 17, factorizations such as (2.50), (2.51), (2.52), (2.54) are in practice only required to hold to within some preset tolerance. In a single-level scheme, it is straight-forward to choose a local tolerance in such a way that $||\mathsf{A} - \mathsf{A}_{\text{approx}}|| \leq \varepsilon$ holds to within some given global tolerance ε. In a multi-level scheme, it is rather difficult to predict how errors aggregate across levels, in particular when the basis matrices U_τ and V_τ are not orthonormal. As an empirical observation, we have found that such error propagation is typically very mild and the error $||\mathsf{A} - \mathsf{A}_{\text{approx}}||$ is very well predicted by the local tolerance in the interpolatory decomposition.

While there are as of yet no à priori error guarantees, it is often possible to produce highly accurate à posteriori error estimates. To this end, let $\mathbf{q} = \mathsf{A}^{-1}\,\mathbf{f}$ and $\mathbf{q}_{\text{approx}} = \mathsf{A}_{\text{approx}}^{-1}\,\mathbf{f}$ denote the exact and the approximate solution, respectively. Then

$$||\mathbf{q}_{\text{approx}} - \mathbf{q}|| = ||\mathsf{A}_{\text{approx}}^{-1}\,\mathsf{A}\,\mathbf{q} - \mathsf{A}_{\text{approx}}^{-1}\,\mathsf{A}_{\text{approx}}\,\mathbf{q}|| \leq ||\mathsf{A}_{\text{approx}}^{-1}||\,||\mathsf{A} - \mathsf{A}_{\text{approx}}||\,||\mathbf{q}||.$$

We can very rapidly and accurately estimate $||\mathsf{A}_{\text{approx}}^{-1}||$ via a power iteration since we have access to a fast matrix-vector multiply for $\mathsf{A}_{\text{approx}}^{-1}$. The factor $||\mathsf{A} - \mathsf{A}_{\text{approx}}||$ can similarly be estimated

whenever we have access to an independent fast matrix-vector multiplication for A. For all BIEs discussed in this chapter, the Fast Multipole Method can serve this function. If it is found that the factor $||\mathsf{A}_{\text{approx}}^{-1}||\,||\mathsf{A} - \mathsf{A}_{\text{approx}}||$ is larger than desired, the compression can be rerun with a smaller local tolerance. (The argument in this section assumes that the inversion is truly exact; a careful error analysis must also account for propagation of round-off errors.)

2.6 Numerical examples

In this section, we illustrate the performance of the direct solver on different boundary integral equations. We consider three classes of problems: single bodied domains, space filling boundary, and two-dimensional surface BIE-like equation.

For each example, a compressed representation of the coefficient matrix was computed via Matlab implementations of the compression scheme described in Section 2.5. Then Fortran 77 implementations of Algorithms 3 (inversion of an HBS matrix), 5 (conversion to standard HBS format), and 2 (matrix-vector multiply) were used to directly solve the respective linear systems. All codes were executed on a desktop computer with 2.8GHz Intel i7 processor and 2GB of RAM. The speed of the compression step will be improved significantly by moving to a Fortran implementation, but since this step is somewhat idiosyncratic to each specific problem, we believe that it is representative to report the times of an unoptimized Matlab code.

2.6.1 Single bodied domains

The performance of the direct solver was tested on linear systems arising upon the Nyström discretization of two BIEs.

The first BIE we consider is

$$\frac{1}{2}\,q(\mathbf{x}) + \int_{\Gamma} \frac{\mathbf{n}(\mathbf{x}') \cdot \big(\mathbf{x} - \mathbf{x}'\big)}{2\pi|\mathbf{x} - \mathbf{x}'|^2}\,q(\mathbf{x}')\,dl(\mathbf{x}') = f(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{2.58}$$

where $\mathbf{n}(\mathbf{y})$ is a unit normal to the contour Γ. This integral equation is a standard representation of the Laplace equation on a domain bordered by Γ when Dirichlet boundary data is specified. Three

geometries were considered:

- **Smooth star:** The star domain illustrated in Figure 2.5(a) is discretized via Gaussian quadrature as described in Section 2.1.3. In the experiment, we fix the size of the computational domain and increase the number of discretization points. This problem is artificial in the sense that the largest experiments use *far* more quadrature points than what is required for any reasonable level of accuracy. It was included simply to demonstrate the asymptotic scaling of the method.

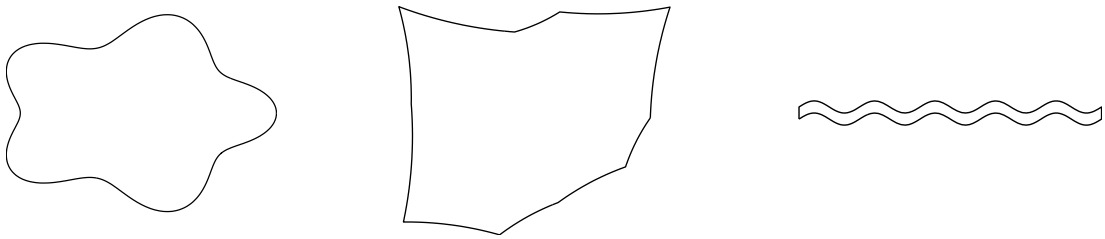
- **Star with corners:** The contour Γ consists of ten segments of circles with different radii as illustrated in Figure 2.5(b). We started with a discretization with 6 panels per segment and 17 Gaussian quadrature nodes per panel. Then grid refinement as described in [51, 12] (with a so-called "simply graded mesh") was used to increase the number of discretization points, cf. Remark 5.

- **Snake:** The contour Γ consists of two sine waves with amplitude 1 that are vertically separated by a distance of 0.2, see Figure 2.5(c). At the end points, two vertical straight lines connect the waves. Composite Gaussian quadrature was used with 25 nodes per panel, four panels on each vertical straight line (refined at the corners to achieve 10 digits of accuracy), and then we used 10 panels per wavelength. The width of the contour was increase from 2 full periods of the wave to 200, and the number of discretization points N was increased accordingly.

Next we consider a BIE associated with the single layer kernel for Helmholtz equation on the star domain (illustrated in Figure 2.5(a)), bordered by Γ. The BIE is:

$$q(\mathbf{x}) + \int_{\Gamma} H_0(k|\mathbf{x} - \mathbf{x}'|)\, q(\mathbf{x}')\, dl(\mathbf{x}') = f(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{2.59}$$

where H_0 is the Hankel function of the first kind of order zero and k is the wave number. We discretize the boundary Γ with a weighted trapezoidal rule [52]. The wave number k is chosen such



(a) (b) (c)

Figure 2.5: The contours Γ used in the numerical experiments for the BIE (2.58) in Section 5.5. (a) Smooth star. (b) Star with corners. (c) Snake.

that the the length of the domain increases from 1.8 wavelengths to 238.7 wavelengths long. Then number of discretization points N is increased to maintain 50 points per wavelength.

For these experiments, the local tolerance in the compression step was set to $\epsilon = 10^{-10}$. (In other words, the interpolatory factorization in (2.54) was required to produce a residual in the Frobenius norm no larger than ε.)

The times corresponding to each step in the direct solver for the BIEs (2.58) and (2.59) are reported in Figure 2.6. Notice that for the Helmholtz problem there is only slight deterioration in the performance as the wave number moves into the high frequency regime.

To assess the accuracy of the direct solver, the errors $\|\mathsf{A} - \mathsf{A}_{\text{approx}}\|$ and $\|\mathsf{I} - \mathsf{A}_{\text{approx}}^{-1}\mathsf{A}\|$ were estimated via a power iteration after each compression (the matrix A and its transpose were applied via the classical FMM [44] run at very high accuracy). The quantity $\|\mathsf{A}_{\text{approx}}^{-1}\|$ was also estimated. The results are reported in Figure 2.7. The quantities reported bound the overall error in the direct solver, see Section 2.5.4.

2.6.2 Space filling domain

Next, the performance of the direct solver was tested on the linear system arising upon the Nyström discretization of the BIE:

$$\frac{1}{2}\,q(\mathbf{x}) + \int_{\Gamma} \left(\frac{\mathbf{n}(\mathbf{x}') \cdot (\mathbf{x} - \mathbf{x}')}{2\pi |\mathbf{x} - \mathbf{x}'|^2} - \frac{1}{2\pi} \log(|\mathbf{x} - \mathbf{x}'|) \right) q(\mathbf{x}')\, dl(\mathbf{x}') = f(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{2.60}$$

where $\mathbf{n}(\mathbf{x})$ is a unit normal to the contour Γ at $\mathbf{x}$.

Equation (2.60) is a standard BIE representation of the Laplace equation when Dirichlet boundary data is specified on a multi-body domain where Γ is the union of the borders of all the bodies. The domain consider consists of a lattice of star domains that are placed such that the minimum horizontal distance between each star is approximately 0.25, as shown in Figure 2.8. The boundary of each star is discretized via the weighted trapezoidal rule [52] with a constant 50 points per star while the number of domains is increased.

For this experiment, the local tolerance in the compression step was set to $\epsilon = 10^{-6}$. The times corresponding to each step in the direct solver are reported in Figure 2.9. While the compression and inversion steps scale as $O(N^{3/2})$, the matrix-vector multiply scales linearly with the number of discretization points. To assess the accuracy of the direct solver, Table 2.1 reports the quantities $\|\mathsf{A} - \mathsf{A}_{\text{approx}}\|$, $\|\mathsf{A}^{-1}_{\text{approx}}\|$, and $\|\mathsf{I} - \mathsf{A}^{-1}_{\text{approx}}\mathsf{A}\|$.

N	$\|\mathsf{A}^{-1}_{\text{approx}}\|$	$\|\mathsf{I} - \mathsf{A}^{-1}_{\text{approx}}\mathsf{A}\|$	$\|\mathsf{A} - \mathsf{A}_{\text{approx}}\|$
800	3409.7	$1.1647e-5$	$6.721e-4$
3200	5139.9	$2.0279e-4$	$2.896e-4$
6400	9216.1	$4.1646e-4$	$7.398e-4$

Table 2.1: Error information for the space filling example.

2.6.3 Two dimensional surface problem

Finally, the performance of the method is tested on the linear system

$$\sum_{i \neq j}^{N} \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|} q_j = f_j, \tag{2.61}$$

where $\{\mathbf{x}_j\}_{j=1}^{N}$ are points in $\mathbb{R}^3$ resulting from the triangulation of torus domain depicted in Figure 2.10. While this is not an integral equation, the system (2.61) has similar rank properties as

the Nyström discretized two dimensional single layer kernel. Let $N = 6272$. Figure 2.11 illustrates the skeleton points in six levels of the HSS factorization of the matrix corresponding to (2.61). Notice that the skeleton points cluster around the boundary of the boxes.

For this experiment, the local tolerance in the compression step was set to $\epsilon = 10^{-6}$. The times corresponding to each step in the direct solver are reported in Figure 2.12(a). Again, the compression and inversion steps scale as $O(N^{3/2})$, while the matrix-vector scales linearly with the number of discretization points. To assess the accuracy of the direct solver, Figure 2.12(b) reports the quantities $\|\mathsf{A} - \mathsf{A}_{\text{approx}}\|$, and $\|\mathsf{I} - \mathsf{A}_{\text{approx}}^{-1}\mathsf{A}\|$. The quantity $\|\mathsf{A}_{\text{approx}}^{-1}\|$ remains a constant 1.4142 for all experiments.

2.7 Extensions and future work

Additional numerical examples illustrating the performance of the direct solver and similar direct solvers are reported in [61, 64, 43].

Currently, the error analysis is very rudimentary. Numerical experiments indicate that in most environments the method is stable and accurate, but this has not yet been demonstrated in any case other than that of symmetric positive definite matrices.

We believe that the method described here in combination with other methods presented in this manuscript provide the basis for a linear complexity inversion technique for surface integral equations. This is work currently in progress.

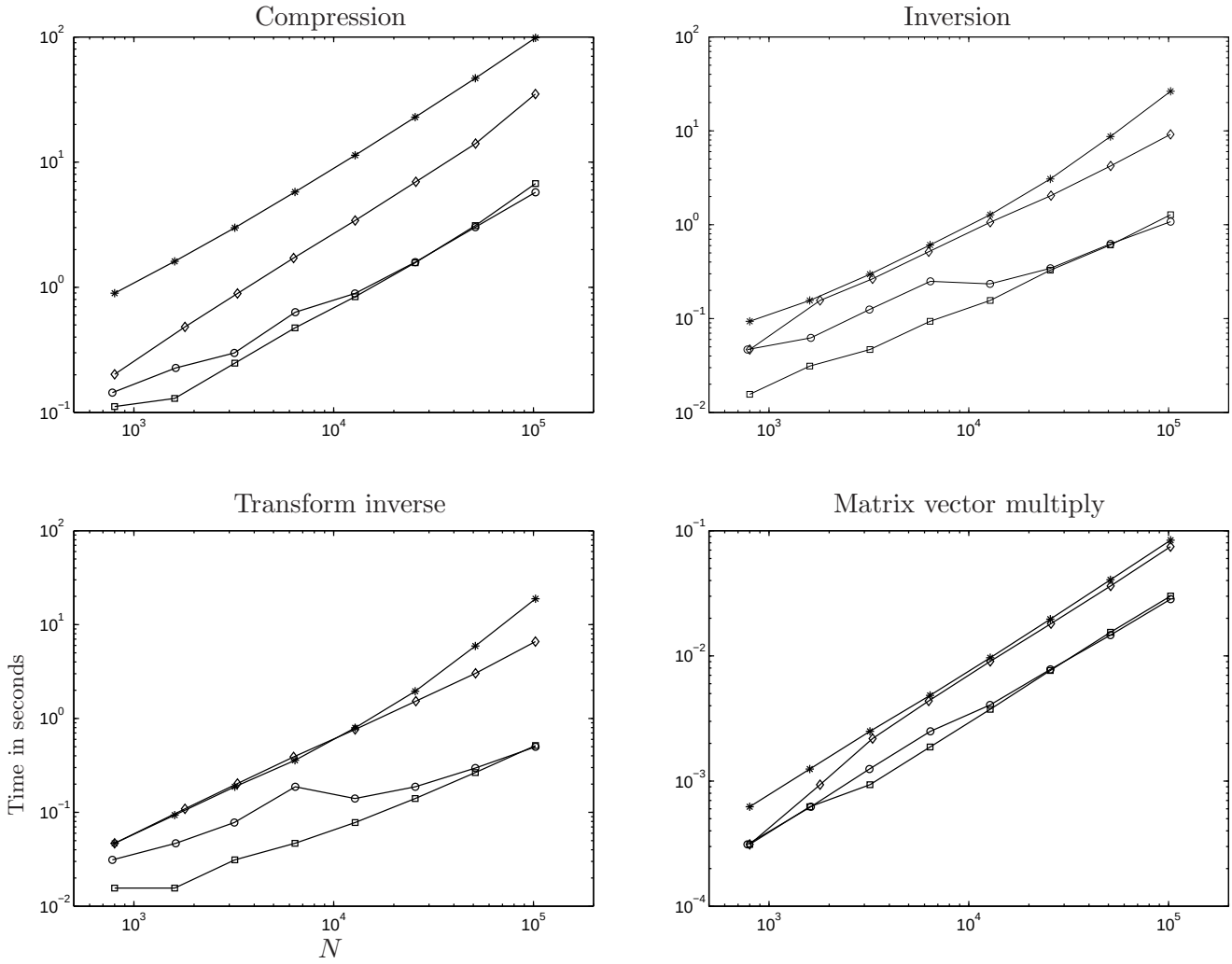


Figure 2.6: The four graphs give the times required for the four steps in the direct solver. Within each graph, the three lines correspond to the three different contours considered: □ – Smooth star, ○ – Star with corners, ◇ – Snake, ∗ – Helmholtz .

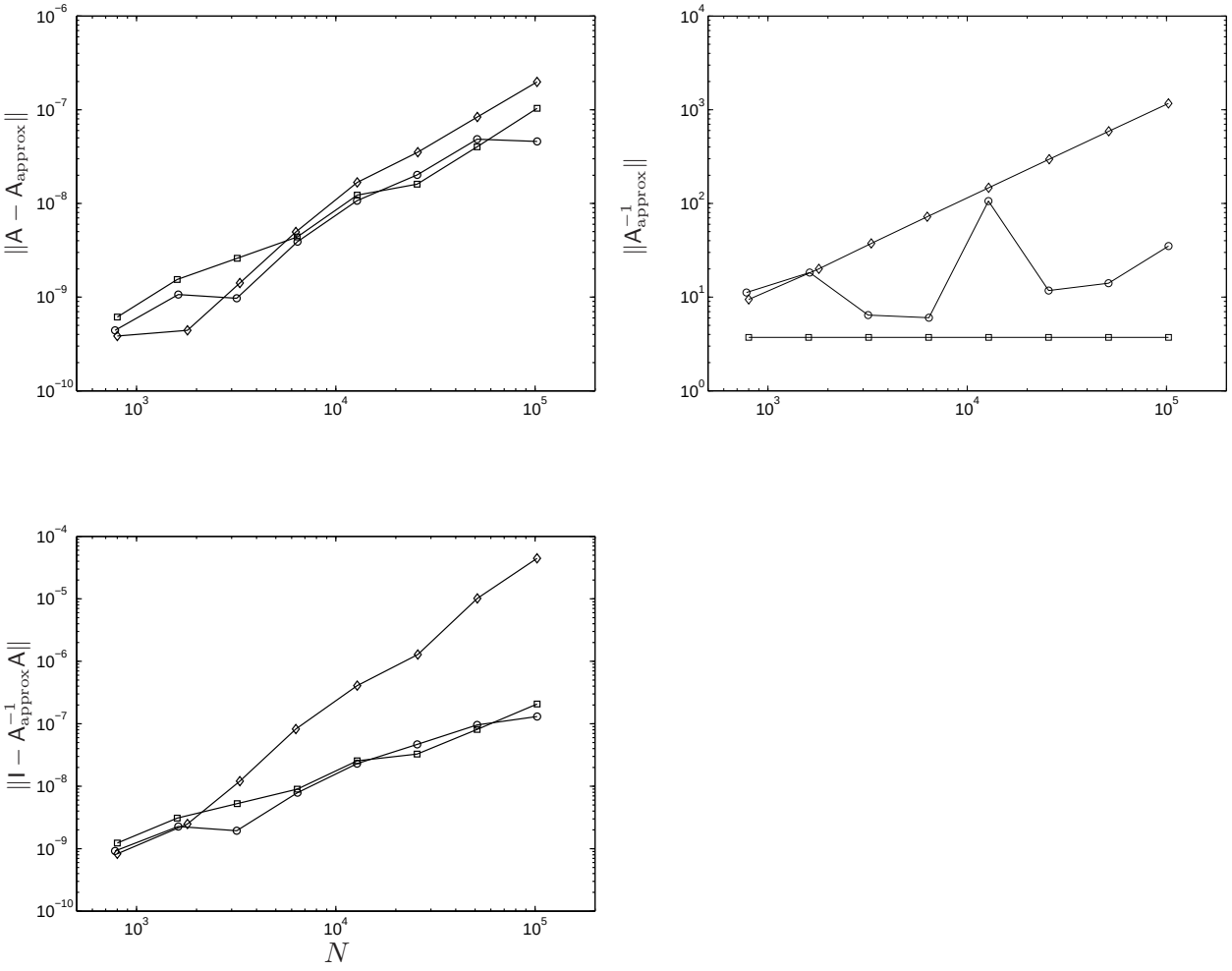


Figure 2.7: Error information for each of the domains: □ – smooth star, ∘ – star with corners, ◇ – snake.

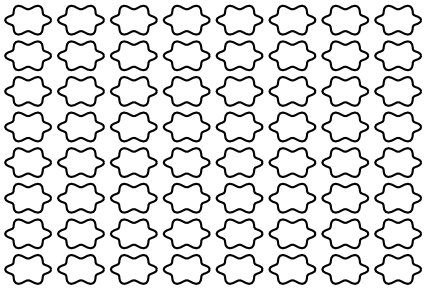


Figure 2.8: The contours which comprise Γ used in the numerical experiment for the BIE (2.60) in Section 5.5.

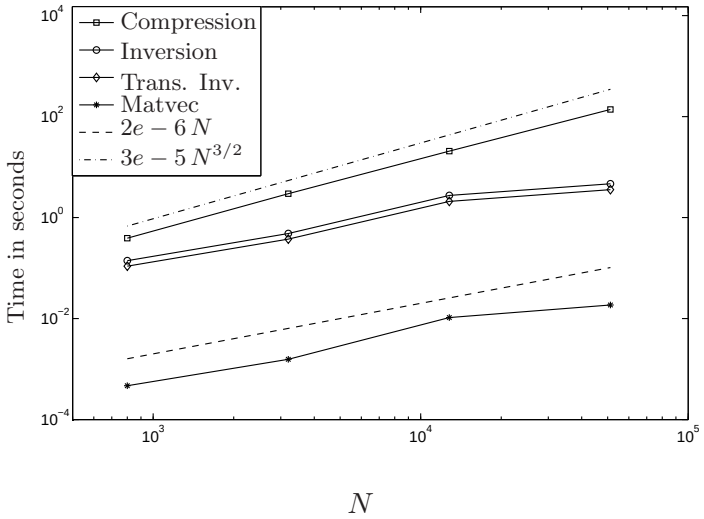


Figure 2.9: Performance of direct solver for the equation (2.60) with multiple bodies in the domain. (Trans. Inv. = Transform Inverse)

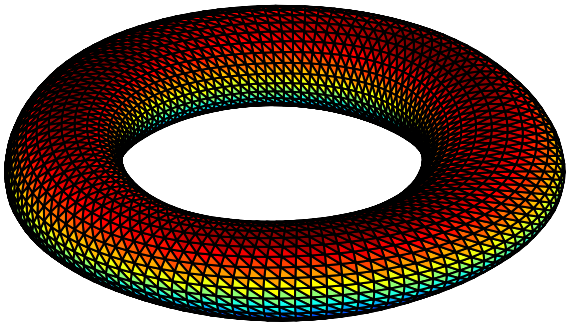
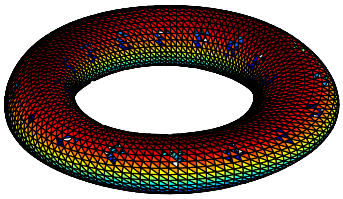
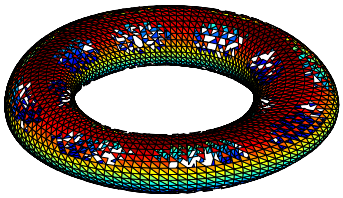


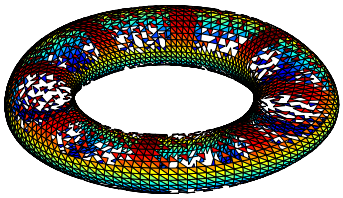
Figure 2.10: The surface which is discretized in the numerical experiments for (2.61).



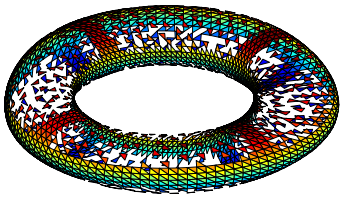
$l = 6$ $n_{\text{skel}} = 5839$



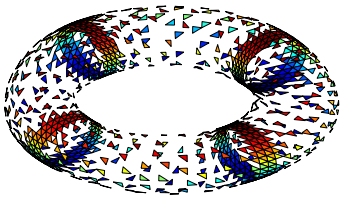
$l = 5$ $n_{\text{skel}} = 4549$



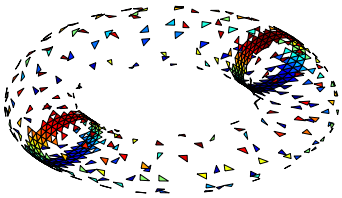
$l = 4$ $n_{\text{skel}} = 3524$



$l = 3$ $n_{\text{skel}} = 2789$



$l = 2$ $n_{\text{skel}} = 1142$



$l = 1$ $n_{\text{skel}} = 584$

Figure 2.11: Illustration of skeleton points for torus domain. Let l denote the level and n_{skel} denote the number skeleton points.

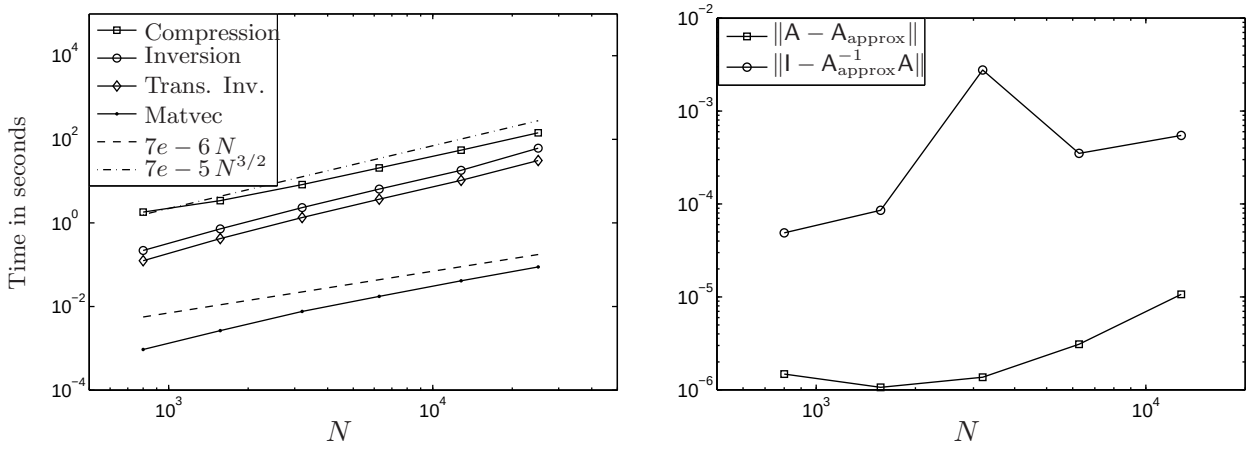


Figure 2.12: Performance of direct solver for the equation (2.61) on the torus domain. (Trans. Inv. = Transform Inverse)

Chapter 3

A direct solver with $O(N)$ complexity for finite difference matrices

In this chapter, we describe a fast direct solution technique for the linear system

$$\mathsf{A}\mathbf{x} = \mathbf{b} \tag{3.1}$$

that arises from a finite element or finite difference discretization of linear boundary value problems. For illustrative purposes, we consider the discrete Laplace operator on a $\sqrt{N} \times \sqrt{N}$ grid domain Ω, ie. the row of A corresponding the point $\mathbf{m}$ is given by

$$[\mathsf{A}u](\mathbf{m}) = \sum_{\mathbf{n} \in \mathbb{B}_m} \alpha_{\mathbf{m},\mathbf{n}}(u(\mathbf{m}) - u(\mathbf{n})), \tag{3.2}$$

where $\alpha_{\mathbf{m},\mathbf{n}}$ is a constant dependent on the discretization, and $\mathbb{B}_m$ is the set of lattice points neighboring $\mathbf{m}$ in Ω.

Example 1: The classic five point stencil is associated with $\alpha_{\mathbf{m},\mathbf{n}} = -1$, and has the following sparsity pattern for a 5×5 block matrix,

$$\mathsf{A} = \begin{bmatrix} \mathsf{B} & -\mathsf{I} & 0 & 0 & 0 \\ -\mathsf{I} & \mathsf{C} & -\mathsf{I} & 0 & 0 \\ 0 & -\mathsf{I} & \mathsf{C} & -\mathsf{I} & 0 \\ 0 & 0 & -\mathsf{I} & \mathsf{C} & -\mathsf{I} \\ 0 & 0 & 0 & -\mathsf{I} & \mathsf{B} \end{bmatrix}$$

where I is the identity matrix,

$$\mathsf{B} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 \\ 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \quad \text{and} \quad \mathsf{C} = \begin{bmatrix} 3 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 3 \end{bmatrix}.$$

Our approach is to accelerate the divide and conquer algorithm known as nested dissection [35]. The idea is to reorder the points in the domain in order to avoid operating on the zero elements in the matrix. The reordering of points allows for a hierarchical way of reducing the number of knowns in the domain. This results in the bulk of the computations being restricted to dense matrices that are much smaller than the original system. The overall complexity of nested dissection is $O(N^{3/2})$. We propose an accelerated version of the method that scales linearly. Other groups have been working on similar solution techniques, such as the Superfast multifrontal method [18, 70] and the $\mathcal{H}$-LU preconditioner [53].

This chapter begins with a basic description of a variation of the nested dissection algorithm. First the domain is partitioned into a sequence of nested boxes, see Section 3.1. Starting with the smallest boxes, the number of unknowns for each box is reduced by condensing the local problem to a problem defined only on the boundary of the box using the technique described in Section 3.2. The operator defined on the boundary of the box is known as the **boundary-to-boundary** operator. Hierarchically merging these operators (see Section 3.3), the boundary-to-boundary operator is found for the box Ω. Figure 3.2 gives a basic illustration of the algorithm. It turns out that the boundary-to-boundary operators are HSS matrices. In Section 3.4, we describe how we exploit this fact to improve the scaling of the nested dissection method. Finally, in Section 3.5, we illustrate the variety of problems where this method will have linear complexity.

3.1 Tree structure

First the domain is partitioned into what is typically called a quad-tree. In this section, we describe a simple technique for constructing such a tree decomposition.

Given an integer N_{leaf}, we partition the box domain Ω into 4^M boxes of equal size which contain no more than N_{leaf} points, where $M = \lfloor \sqrt{\frac{N}{N_{\text{leaf}}}} \rfloor$. These 4^M small boxes form the *leaves* of the tree. By merging the leaves by sets of fours into boxes with twice the side length, we form the 4^{M-1} boxes that make up the next level in the tree. This process is repeated until the box Ω is recovered. We call Ω the root of the tree.

3.2 The Schur complement problem

A technique for constructing the boundary-to-boundary operator for boxes on the leaf level is presented in this section.

Let $\hat{\Omega}$ denote a leaf box of Ω with boundary Γ and interior $\hat{\Omega}_{\text{i}}$ (so that $\hat{\Omega}$ is the disjoint union of Γ and $\hat{\Omega}_{\text{i}}$). Suppose that Γ has N_{b} nodes, and $\hat{\Omega}_{\text{i}}$ has N_{i} nodes. Consider the linear problem that arises from restricting (3.1) to $\hat{\Omega}$. By partitioning this restricted problem according to the two parts $\hat{\Omega} = \Gamma \cup \hat{\Omega}_{\text{i}}$, and assuming that the box is loaded only on the boundary (*i.e.* $f(j) = 0$ for $j \in \hat{\Omega}_{\text{i}}$), we obtain the equation

$$\begin{bmatrix} \mathsf{A}_{\text{b,b}} & \mathsf{A}_{\text{b,i}} \\ \mathsf{A}_{\text{i,b}} & \mathsf{A}_{\text{i,i}} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\text{b}} \\ \mathbf{x}_{\text{i}} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{\text{b}} \\ \mathbf{0} \end{bmatrix}. \tag{3.3}$$

If we are interested only in $\mathbf{x}$ restricted to the boundary ($\mathbf{x}_{\text{b}}$), it is given by the equation

$$\mathbf{x}_{\text{b}} = \mathsf{S}^{-1}\, \mathbf{f}_{\text{b}},$$

where S is the $N_{\text{b}} \times N_{\text{b}}$ matrix

$$\mathsf{S} = \mathsf{A}_{\text{b,b}} - \mathsf{A}_{\text{b,i}}\, \mathsf{A}_{\text{i,i}}^{-1}\, \mathsf{A}_{\text{i,b}}, \tag{3.4}$$

which we define as the *Schur complement* or boundary-to-boundary operator of $\hat{\Omega}$.

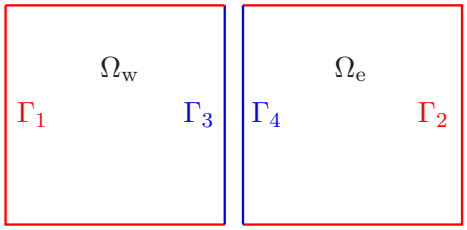


Figure 3.1: Labeling of nodes when merging two boxes.

3.3 Merging two Schur complements

In this section, we present a technique to further reduce the number of unknowns by merging the Schur complements for two touching boxes.

Suppose that $\hat{\Omega} = \Omega_w \cup \Omega_e$ ("west" and "east"), as shown in Figure 3.1. Further suppose the corresponding Schur complements S_w and S_e were previously found via the technique in Section 3.2. We seek the Schur complement S of $\hat{\Omega}$. This means we need to eliminate the "interior" points which consists solely of the points along the middle lines (marked in blue in the figure).

To eliminate these points, we first partition the boundary Γ_w into the subsets Γ_1 and Γ_3, and partition Γ_e into Γ_2 and Γ_4 as shown in Figure 3.1. Next, we partition the Schur complements S_w and S_e accordingly,

$$\mathsf{S}_w = \begin{bmatrix} \mathsf{S}_{11} & \mathsf{S}_{13} \\ \mathsf{S}_{31} & \mathsf{S}_{33} \end{bmatrix}, \qquad \text{and} \qquad \mathsf{S}_e = \begin{bmatrix} \mathsf{S}_{22} & \mathsf{S}_{24} \\ \mathsf{S}_{42} & \mathsf{S}_{44} \end{bmatrix}.$$

Supposing that the interior edges are unloaded, equation (3.1) restricted to $\hat{\Omega}$ now reads

$$\left[\begin{array}{cc|cc} \mathsf{S}_{11} & \mathsf{A}_{12} & \mathsf{S}_{13} & 0 \\ \mathsf{A}_{21} & \mathsf{S}_{22} & 0 & \mathsf{S}_{24} \\ \hline \mathsf{S}_{31} & 0 & \mathsf{S}_{33} & \mathsf{A}_{34} \\ 0 & \mathsf{S}_{24} & \mathsf{A}_{43} & \mathsf{S}_{44} \end{array}\right] \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{3.5}$$

where A_{ij} are the relevant sub-matrices of the original discrete Laplacian A.

From (3.5), one finds that the Schur complement of the large box is

$$\mathsf{S} = \begin{bmatrix} \mathsf{S}_{11} & \mathsf{A}_{12} \\ \mathsf{A}_{21} & \mathsf{S}_{22} \end{bmatrix} - \begin{bmatrix} \mathsf{S}_{13} & 0 \\ 0 & \mathsf{S}_{24} \end{bmatrix} \begin{bmatrix} \mathsf{S}_{33} & \mathsf{A}_{34} \\ \mathsf{A}_{43} & \mathsf{S}_{44} \end{bmatrix}^{-1} \begin{bmatrix} \mathsf{S}_{31} & 0 \\ 0 & \mathsf{S}_{42} \end{bmatrix}. \tag{3.6}$$

Roughly speaking, the computational cost of the nested dissection method is dictated by the cost at the highest level. Thus it is the inversion of the $2\sqrt{N} \times 2\sqrt{N}$ matrix

$$\mathsf{S}_{\text{mid}} = \begin{bmatrix} \mathsf{S}_{33} & \mathsf{A}_{34} \\ \mathsf{A}_{43} & \mathsf{S}_{44} \end{bmatrix} \tag{3.7}$$

that results in the scheme having complexity $O(N^{3/2})$.

3.4 Accelerated nested dissection

To improve the scaling, we develop a fast way of applying the inverse of S_{mid}. To illustrate the technique, consider the linear equation

$$\mathsf{S}_{\text{mid}} \begin{bmatrix} \mathsf{X}_1 \\ \mathsf{X}_2 \end{bmatrix} = \begin{bmatrix} \mathsf{S}_{33} & \mathsf{A}_{34} \\ \mathsf{A}_{43} & \mathsf{S}_{44} \end{bmatrix} = \begin{bmatrix} \mathsf{R}_1 \\ \mathsf{R}_2 \end{bmatrix}. \tag{3.8}$$

The solutions to this equation are

$$\mathsf{X}_2 = (\mathsf{S}_{44} - \mathsf{A}_{43}\mathsf{S}_{33}^{-1}\mathsf{A}_{34})^{-1}(\mathsf{R}_2 - \mathsf{A}_{43}\mathsf{S}_{33}^{-1}\mathsf{R}_1)$$

and

$$\mathsf{X}_1 = \mathsf{S}_{33}^{-1}\mathsf{R}_1 - \mathsf{S}_{33}^{-1}\mathsf{A}_{34}\mathsf{X}_2.$$

It turns out that the Schur complement matrices are HSS (See Chapter 2 for a detailed definition). This fact along with several other properties of the block matrices that comprise S_{mid} allow the matrices X_1 and X_2 to be computed rapidly.

These properties include:

- S_{33} and S_{44} are HSS matrices and thus can be inverted with linear computational complexity via Algorithm 3. The inverse can then be applied to vectors rapidly by Algorithm 4.

- A_{34} and A_{43} are anti-diagonal matrices. Thus the product $\mathsf{A}_{43}\mathsf{S}_{33}^{-1}\mathsf{A}_{34}$ is HSS and can be computed rapidly via Algorithm 6.

- $\mathsf{S}_{44} - \mathsf{A}_{43}\mathsf{S}_{33}^{-1}\mathsf{A}_{34}$ is the sum of two HSS matrices that happens to also be HSS. Hence, not only can the matrix sum be computed in linear time (by Algorithm 7) but the inverse can as well.

Additionally, the block matrices S_{31} and S_{42} are low-rank matrices. Therefore, we need only apply the inverse of S_{mid} to a small number of vectors. The result is a scheme whose computational complexity scales linearly with the number of points in the domain.

Remark 23. *Multiplying a matrix by two anti-diagonals effectively reverses the numbering of the matrix. Hence, we define the twin box of τ to be a box that is on the same level as τ and is the same distance from the midpoint but on the opposite side of the midpoint. For simplicity of presentation, Algorithm 6 assumes every box not on the root level has a twin box that is the same size.*

Remark 24. *Algorithm 7 assumes that the HSS matrices A_1 and A_2 are factorized according to the same tree structure.*

The efficiency of the method is further improved by using fast matrix algebra to compute the diagonal blocks of (3.6). Recall that the matrices S_{11} and S_{22} are HSS. In addition, the matrices S_{13} and S_{24} are low-rank. It turns out that $\mathsf{S}_{13}\,\mathsf{X}_1$ and $\mathsf{S}_{24}\,\mathsf{X}_2$ are low-rank HSS matrices. These matrices can be HSS factorized to have the same tree structure as S_{11} and S_{22}, respectively, via Algorithm 8. Then Algorithm 7 is used to compute the matrix addition.

Remark 25. *Appendix B details techniques for efficient storage and processing of the Schur complements between merge steps.*

3.5 Numerical Results

In this section, we explore the potential of the proposed scheme to solve elliptic boundary value problems in linear time. We consider problems of the form

$$\begin{cases} -\Delta u(\mathbf{x}) + b(\mathbf{x})u_x(\mathbf{x}) + c(\mathbf{x})u_y(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \Omega = [0,1]^2, \\ u(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Gamma, \end{cases} \tag{3.9}$$

where $b(\mathbf{x})$, $c(\mathbf{x})$, $f(\mathbf{x})$, and $g(\mathbf{x})$ are functions defined in Ω. We discretize equation (3.9) with the finite difference method associated with the five point stencil results in having to solve an $N \times N$ linear system, where N denotes the number of discretization points in Ω.

We consider four different choices of $b(\mathbf{x})$ and $c(\mathbf{x})$. They are

- **Laplacian:** Let $b(\mathbf{x}) = c(\mathbf{x}) = 0$. This corresponds to the system illustrated in Example 1.
- **Constant convection:** Let $c(\mathbf{x}) = 0$ and the convection in the x direction be constant. We consider $b(\mathbf{x}) = 100$ and $b(\mathbf{x}) = 1000$.
- **Divergence free diffusion-convection:** Let $b(\mathbf{x}) = 125\cos(4\pi y)$ and $c(\mathbf{x}) = 125\sin(4\pi x)$.
- **Diffusion-Convection with sources and sinks:** Let $b(\mathbf{x}) = 125\cos(4\pi x)$ and $c(\mathbf{x}) = 125\sin(4\pi y)$.

In addition, we also consider two numerical examples for the case of the random graph Laplacian. In the context of (3.9), this corresponds to $b(\mathbf{x}) = c(\mathbf{x}) = 0$ and letting the connectivity between vary between 1 and α where $\alpha = 2$, and 1000.

All experiments are run on a Dell desktop computer with 2.8GHz Intel i7 processor and 3GB of RAM. The method was implemented in Matlab. While this implementation is unoptimized, we believe it is sufficient for illustrating the the potential of the proposed technique. Additionally, for all experiments the tolerance of the HSS representation of the Schur complement matrix is set to 10^{-7}.

First we consider the matrix corresponding to the Laplacian operator. Ω is discretized with $n \times n$ points. Let $N = n^2$. Table 3.1 reports the time in seconds (T_{solve}) for constructing the global boundary-to-boundary operator, the time in seconds (T_{apply}) for applying the boundary-to-boundary operator to a vector, the amount of memory (M) in MB required to store the boundary-to-boundary operator and two error measures, e_1 and e_2. The value e_1 denotes the l^2-error in the vector $\mathsf{S}^{-1} r$ where r is a unit vector of random direction. The value e_2 denotes the l^2-error in the first column of S^{-1}.

Notice that while the time to construct the global Schur complement is linear with respect to N, the cost of apply the operator to a vector scales as $O(\sqrt{N})$.

For each problem, we fix the discretization to a 1024×1024 grid with leaf boxes containing 64 points. This means that the hierarchical tree has five levels. The accelerated nested dissection will scale linearly when the HSS ranks of the Schur complement operators S for each level does not grow with the size of S. Tables 3.2-3.8 report the ranks of the HSS blocks of size N_{B} for the different levels in the accelerated nested dissection method for the various problems. The results indicate that the method will not scale as desired when the physics of the underlying problem is ill-conditioned such as in the convection-diffusion problem with sources and sinks.

N	T_{solve} (sec)	T_{apply} (sec)	M (MB)	e_1	e_2
512^2	7.98	0.007	8.4	$5.56e-7$	$6.04e-7$
1024^2	26.49	0.014	18.6	$4.72e-7$	$4.98e-7$
2048^2	98.46	0.020	33.1	$2.89e-7$	$2.90e-7$
4096^2	435.8	0.039	65.6	-	-

Table 3.1: Times, errors and amount of memory required to build the global Schur complement for the discretized Poisson problem via the accelerated nested dissection method.

3.6 Concluding remarks

In this chapter, we presented a fast direct technique for solving the linear systems that arise from the finite element or finite difference discretization of elliptic boundary value problems.

	$N_B = 50$	$N_B = 100$	$N_B = 200$	$N_B = 400$	$N_B = 800$
Level 5	23.75	-	-	-	-
Level 4	23.89	27.50	-	-	-
Level 3	23.26	27.44	30.75	-	-
Level 2	22.82	26.63	30.44	33.50	-
Level 1	22.06	25.87	29.58	33.22	36.25

Table 3.2: HSS ranks for Laplacian.

Numerical results indicate that the method will scale linearly with N, the number of discretization points for a variety of problems. The cost of building the solver dominates the computational cost. However, once the solver is built, constructing the solution for multiple right-hand sides is essentially free. For problem involving approximately 16 million unknowns it takes about 7 minutes to build the solver, and 0.04 seconds to apply it to a right hand side.

In the worst case scenario, the method scales as $O(N^{1.5})$. This appears to happen when the physics of the underlying PDE is ill-conditioned. For these problems, it is advantageous to use the proposed method as preconditioner to accelerate the convergence of iterative solution techniques. The superfast multifrontal method [18, 70] and the $\mathcal{H}$-LU factorization [53] techniques are already being implemented in this capacity.

	$N_B = 50$	$N_B = 100$	$N_B = 200$	$N_B = 400$	$N_B = 800$
Level 5	23.32	-	-	-	-
Level 4	23.76	27	-	-	-
Level 3	23.26	27.16	30.38	-	-
Level 2	22.13	26.32	30.22	33.38	-
Level 1	22.03	25.64	29.26	32.78	35.75

Table 3.3: HSS ranks for Laplacian with random connectivity between 1 and 2.

	$N_B = 50$	$N_B = 100$	$N_B = 200$	$N_B = 400$	$N_B = 800$
Level 5	22.32	-	-	-	-
Level 4	22.78	26.21	-	-	-
Level 3	22.36	26.33	29.63	-	-
Level 2	21.54	25.71	29.53	32.63	-
Level 1	21.29	24.97	29.11	32.33	35.50

Table 3.4: HSS ranks for Laplacian with random connectivity between 1 and 2.

	$N_B = 50$	$N_B = 100$	$N_B = 200$	$N_B = 400$	$N_B = 800$
Level 5	23.50	-	-	-	-
Level 4	23.58	26.75	-	-	-
Level 3	22.94	26.36	29.31	-	-
Level 2	22.02	25.86	28.77	31.12	-
Level 1	21.83	24.89	27.78	30.33	33.75

Table 3.5: HSS ranks for $b(\mathbf{x}) = 100$.

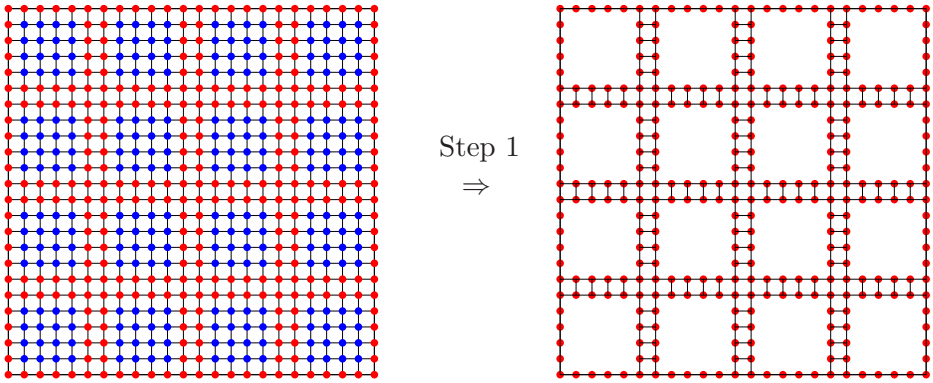
	$N_B = 50$	$N_B = 100$	$N_B = 200$	$N_B = 400$	$N_B = 800$
Level 5	33.25		-	-	-
Level 4	27.77	45.75	-	-	-
Level 3	22.05	28.88	45.50	-	-
Level 2	19.58	25.57	34.66	57.75	-
Level 1	17.97	22.15	29.89	43.00	74.75

Table 3.6: HSS ranks for $b(\mathbf{x}) = 1000$.

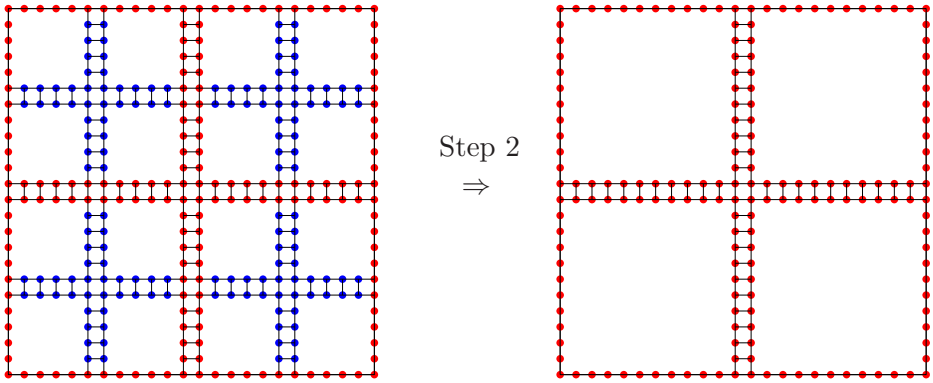
	$N_B = 50$	$N_B = 100$	$N_B = 200$	$N_B = 400$	$N_B = 800$
Level 5	23.27	-	-	-	-
Level 4	23.49	26.57	-	-	-
Level 3	22.89	26.47	28.87	-	-
Level 2	21.98	25.84	29.05	31.12	-
Level 1	21.66	24.84	28.10	30.77	34.75

Table 3.7: HSS ranks for divergence free convection diffusion.

Step 1: Partition the box into small boxes. For each box, identify the internal nodes (marked in blue) and eliminate them using formula (3.4).



Step 2: Merge 2×2 groups of small boxes to form larger boxes, and eliminate the interior nodes of the larger boxes (marked in blue) using either the process identified in Section 3.3.



Step 3, 4, 5, . . . : Repeat Step 2 on the coarser levels until all interior nodes have been eliminated.

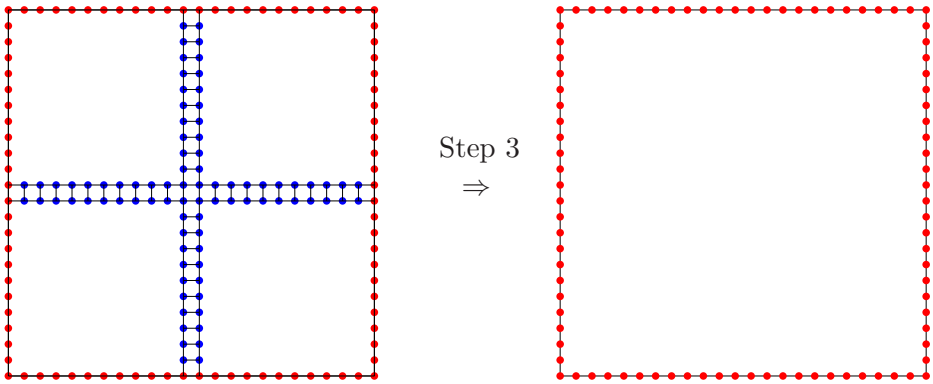


Figure 3.2: Hierarchical merging of boxes in a quad tree.

ALGORITHM 3 (inversion of an HBS matrix)

loop over all levels, finer to coarser, $\ell = L,\, L-1,\, \ldots, 1$
  loop over all boxes τ on level ℓ,
    if τ is a leaf node
      $\tilde{\mathsf{D}}_\tau = \mathsf{D}_\tau$
    else
      Let σ_1 and σ_2 denote the children of τ.
      $\tilde{\mathsf{D}}_\tau = \begin{bmatrix} \hat{\mathsf{D}}_{\sigma_1} & \mathsf{B}_{\sigma_1,\sigma_2} \\ \mathsf{B}_{\sigma_2,\sigma_1} & \hat{\mathsf{D}}_{\sigma_2} \end{bmatrix}$
    end if
    $\hat{\mathsf{D}}_\tau = \left(\mathsf{V}_\tau^* \, \tilde{\mathsf{D}}_\tau^{-1} \, \mathsf{U}_\tau\right)^{-1}$.
    $\mathsf{E}_\tau = \tilde{\mathsf{D}}_\tau^{-1} \, \mathsf{U}_\tau \, \hat{\mathsf{D}}_\tau$.
    $\mathsf{F}_\tau^* = \hat{\mathsf{D}}_\tau \, \mathsf{V}_\tau^* \, \tilde{\mathsf{D}}_\tau^{-1}$.
    $\mathsf{G}_\tau = \hat{\mathsf{D}}_\tau - \tilde{\mathsf{D}}_\tau^{-1} \, \mathsf{U}_\tau \, \hat{\mathsf{D}}_\tau \, \mathsf{V}_\tau^* \, \tilde{\mathsf{D}}_\tau^{-1}$.
  end loop
end loop
$\mathsf{G}_1 = \begin{bmatrix} \hat{\mathsf{D}}_2 & \mathsf{B}_{2,3} \\ \mathsf{B}_{3,2} & \hat{\mathsf{D}}_3 \end{bmatrix}^{-1}$.

	$N_{\mathrm{B}} = 50$	$N_{\mathrm{B}} = 100$	$N_{\mathrm{B}} = 200$	$N_{\mathrm{B}} = 400$	$N_{\mathrm{B}} = 800$
Level 5	23.25	-	-	-	-
Level 4	23.41	26.49	-	-	-
Level 3	22.68	26.00	28.15	-	-
Level 2	21.57	24.55	26.44	27.75	-
Level 1	20.97	23.35	24.73	25.44	24.25

Table 3.8: HSS ranks for convection-diffusion with sources and sinks.

ALGORITHM 4 (application of inverse)

Given a vector $\mathbf{u}$*, compute* $\mathbf{q} = \mathsf{A}^{-1}\,\mathbf{u}$ *using the compressed representation of* A^{-1} *resulting from Algorithm 3.*

loop over all leaf boxes τ
  $\hat{\mathbf{u}}_\tau = \mathsf{F}_\tau^*\,\mathbf{u}(I_\tau)$.
end loop

loop over all levels, finer to coarser, $\ell = L,\ L-1,\ \ldots, 1$
  loop over all parent boxes τ on level ℓ,
    Let σ_1 and σ_2 denote the children of τ.
    $\hat{\mathbf{u}}_\tau = \mathsf{F}_\tau^* \begin{bmatrix} \hat{\mathbf{u}}_{\sigma_1} \\ \hat{\mathbf{u}}_{\sigma_2} \end{bmatrix}$.
  end loop
end loop

$$\begin{bmatrix} \hat{\mathbf{q}}_2 \\ \hat{\mathbf{q}}_3 \end{bmatrix} = \hat{\mathsf{G}}_1 \begin{bmatrix} \hat{\mathbf{u}}_2 \\ \hat{\mathbf{u}}_3 \end{bmatrix}.$$

loop over all levels, coarser to finer, $\ell = 1,\ 2,\ \ldots,\ L-1$
  loop over all parent boxes τ on level ℓ
    Let σ_1 and σ_2 denote the children of τ.
    $\begin{bmatrix} \hat{\mathbf{q}}_{\sigma_1} \\ \hat{\mathbf{q}}_{\sigma_2} \end{bmatrix} = \mathsf{E}_\tau\,\hat{\mathbf{u}}_\tau + \mathsf{G}_\tau \begin{bmatrix} \hat{\mathbf{u}}_{\sigma_1} \\ \hat{\mathbf{u}}_{\sigma_2} \end{bmatrix}$.
  end loop
end loop

loop over all leaf boxes τ
  $\mathbf{q}(I_\tau) = \mathsf{E}_\tau\,\hat{\mathbf{q}}_\tau + \mathsf{G}_\tau\,\mathbf{u}(I_\tau)$.
end loop

ALGORITHM 6 (Multiplication by two anti-diagonal matrices)

Given two vectors $\mathbf{l}_1$ *and* $\mathbf{l}_2$ *and an HSS factorized matrix* $\hat{\mathsf{A}}$*, this algorithm computes* $\mathsf{A} = \text{antidiag}(\mathbf{l}_1)\,\hat{\mathsf{A}}\,\text{antidiag}(\mathbf{l}_2)$.

Create the list of twin boxes (see Remark 23).
loop over all levels, finer to coarser, $\ell = L,\, L-1,\, \ldots, 1$
    loop over all boxes τ on level ℓ,
        Let $\hat{\tau}$ denote the twin box of τ.
        if τ is a leaf node
            Let *ind* denote the indices associated with τ.
            $\tilde{\mathsf{U}}_\tau = \mathsf{U}_{\hat{\tau}}\,\text{antidiag}(\mathbf{l}_2(ind))$
            $\tilde{\mathsf{V}}_\tau = \text{antidiag}(\mathbf{l}_1(ind))\,\mathsf{V}_{\hat{\tau}}$
            $\tilde{\mathsf{D}}_\tau = \text{antidiag}(\mathbf{l}_1(ind))\,\mathsf{D}_{\hat{\tau}}\,\text{antidiag}(\mathbf{l}_2(ind))$
            $\tilde{\mathsf{B}}_\tau = \mathsf{B}_{\hat{\tau}}$
        else
            $\tilde{\mathsf{U}}_\tau = \mathsf{U}_{\hat{\tau}}$
            $\tilde{\mathsf{V}}_\tau = \mathsf{V}_{\hat{\tau}}$
            $\tilde{\mathsf{B}}_\tau = \mathsf{B}_{\hat{\tau}}$
        end if
    end loop
end loop
Recompress the matrix via Algorithm 9.

ALGORITHM 7 (Addition of two HSS matrices)

Given two HSS factorized matrices A_1 *and* A_2*, this algorithm computes* $\mathsf{A} = \mathsf{A}_1 + \mathsf{A}_2$*.*

loop over all levels, finer to coarser, $\ell = L,\, L-1,\, \ldots, 1$
  loop over all boxes τ on level ℓ,
    if τ is a leaf node
      $\tilde{\mathsf{D}}_\tau = \mathsf{D}_\tau^1 + \mathsf{D}_\tau^2$
      $\tilde{\mathsf{U}}_\tau = \begin{bmatrix}\mathsf{U}_\tau^1 \, \mathsf{U}_\tau^2\end{bmatrix}$
      $\tilde{\mathsf{V}}_\tau = \begin{bmatrix}\mathsf{V}_\tau^1 \, \mathsf{V}_\tau^2\end{bmatrix}$
    else
      Let σ_1 and σ_2 denote the children τ.

$$\tilde{\mathsf{U}}_\tau = \begin{bmatrix} \mathsf{U}_\tau^{1,\sigma_1} & 0 & 0 & 0 \\ 0 & \mathsf{U}_\tau^{2,\sigma_1} & 0 & 0 \\ 0 & 0 & \mathsf{U}_\tau^{1,\sigma_2} & 0 \\ 0 & 0 & 0 & \mathsf{U}_\tau^{2,\sigma_2} \end{bmatrix}$$

$$\tilde{\mathsf{V}}_\tau = \begin{bmatrix} \mathsf{V}_\tau^{1,\sigma_1} & 0 & 0 & 0 \\ 0 & \mathsf{V}_\tau^{2\sigma_1} & 0 & 0 \\ 0 & 0 & \mathsf{V}_\tau^{1,\sigma_2} & 0 \\ 0 & 0 & 0 & \mathsf{V}_\tau^{2,\sigma_2} \end{bmatrix}$$

    end if
    $\tilde{\mathsf{B}}_\tau = \begin{bmatrix} \mathsf{B}_\tau^1 & 0 \\ 0 & \mathsf{B}_\tau^2 \end{bmatrix}$
  end loop
end loop

Recompress the matrix via Algorithm 9.
Note: $\mathsf{U}_\tau^{1,\sigma_1}$ denotes the restriction of U_τ^1 acting on σ_1.

ALGORITHM 8 (HSS factorize a matrix given in QR-factorized form)

Given an HSS factorized matrix A *and the QR-factorization of a second matrix* $\mathsf{Q}\,\mathsf{R}$*, this algorithm computes the HSS factorization of* $\tilde{\mathsf{A}} = \mathsf{Q}\,\mathsf{R}$ *using the tree structure of* $\tilde{\mathsf{A}}$*.*

copy tree information from A
loop over all levels, finer to coarser, $\ell = L,\ L-1,\ \ldots,1$
  loop over all boxes τ on level ℓ,
    if τ is a leaf node
      Let *ind* denote the indices associated with τ
      $\mathsf{A}_{12} = \mathsf{Q}\,(ind, :\,)$
      $\mathsf{A}_{21} = \mathsf{R}\,(:\,, ind)^T$
      $\tilde{\mathsf{D}}_\tau = \mathsf{A}_{12}\,\mathsf{A}_{21}^T$
    else
      Let σ_1 and σ_2 denote the children τ.
      $\mathsf{A}_{12} = \begin{bmatrix} \mathsf{W}_{\sigma_1} \\ \mathsf{W}_{\sigma_2} \end{bmatrix}$
      $\mathsf{A}_{21} = \begin{bmatrix} \mathsf{P}_{\sigma_1} \\ \mathsf{P}_{\sigma_2} \end{bmatrix}$
      $\tilde{\mathsf{B}}_{\sigma_1} = \mathsf{W}_{\sigma_1}\,\mathsf{P}_{\sigma_2}^T$
      $\tilde{\mathsf{B}}_{\sigma_2} = \mathsf{W}_{\sigma_2}\,\mathsf{P}_{\sigma_1}^T$
    end if
    $[\mathsf{U}_\tau, \mathsf{W}_\tau] = \mathrm{qr}(\mathsf{A}_{12})$
    $[\mathsf{V}_\tau, \mathsf{P}_\tau] = \mathrm{qr}(\mathsf{A}_{21})$
  end loop
end loop
      $\tilde{\mathsf{B}}_2 = \mathsf{W}_2\,\mathsf{P}_3^T$
      $\tilde{\mathsf{B}}_3 = \mathsf{W}_3\,\mathsf{P}_2^T$

ALGORITHM 9 (Recompress an HSS matrix that has undergone some fast matrix algebra)

Given an HSS factorized matrix A *that has undergone some fast matrix algebra and a desired accuracy* ϵ*, this algorithm recompresses the matrix so that its factors are in a standard HSS form.*

loop over all levels, finer to coarser, $\ell = L,\, L-1,\, \ldots, 0$
  loop over all boxes τ on level ℓ,
    if τ is a leaf node
      $\tilde{\mathsf{D}}_\tau = \mathsf{D}_\tau$
    else
      Let σ_1 and σ_2 denote the children τ.
      $\begin{bmatrix} \tilde{\mathsf{U}}_{\sigma_1}, J^U_{\sigma_1} \end{bmatrix} = \mathrm{ID}(\mathsf{U}_{\sigma_1}, \epsilon)$
      $\begin{bmatrix} \tilde{\mathsf{U}}_{\sigma_2}, J^U_{\sigma_2} \end{bmatrix} = \mathrm{ID}(\mathsf{U}_{\sigma_2}, \epsilon)$
      $\begin{bmatrix} \tilde{\mathsf{V}}_{\sigma_1}, J^V_{\sigma_1} \end{bmatrix} = \mathrm{ID}(\mathsf{V}_{\sigma_1}, \epsilon)$
      $\begin{bmatrix} \tilde{\mathsf{V}}_{\sigma_2}, J^V_{\sigma_2} \end{bmatrix} = \mathrm{ID}(\mathsf{V}_{\sigma_2}, \epsilon)$
      $\tilde{\mathsf{B}}_{\sigma_1} = \mathsf{U}_{\sigma_1}(J^U_{\sigma_1}, :)\, \mathsf{B}_{\sigma_1}\, \left(\mathsf{V}_{\sigma_2}(J^V_{\sigma_2}, :)\right)^T$
      $\tilde{\mathsf{B}}_{\sigma_2} = \mathsf{U}_{\sigma_2}(J^U_{\sigma_2}, :)\, \mathsf{B}_{\sigma_2}\, \left(\mathsf{V}_{\sigma_1}(J^V_{\sigma_1}, :)\right)^T$
      if $\ell > 0$
        $\mathsf{U}_\tau = \begin{bmatrix} \mathsf{U}_{\sigma_1}(J^U_{\sigma_1}, :) & \mathsf{0} \\ \mathsf{0} & \mathsf{U}_{\sigma_2}(J^U_{\sigma_2}, :) \end{bmatrix} \mathsf{U}_\tau$
        $\mathsf{V}_\tau = \begin{bmatrix} \mathsf{V}_{\sigma_1}(J^V_{\sigma_1}, :) & \mathsf{0} \\ \mathsf{0} & \mathsf{V}_{\sigma_2}(J^V_{\sigma_2}, :) \end{bmatrix} \mathsf{V}_\tau$
      end if
    end if
  end loop
end loop

Chapter 4

A fast solver for Poisson problems on infinite regular lattices

This chapter describes an efficient technique for solving Poisson problems defined on the integer lattice $\mathbb{Z}^2$. For simplicity of presentation, we limit our attention to the equation

$$[\mathsf{A}u](\mathbf{m}) = f(\mathbf{m}), \qquad \mathbf{m} \in \mathbb{Z}^2, \tag{4.1}$$

where $f = f(\mathbf{m})$ and $u = u(\mathbf{m})$ are scalar valued functions on $\mathbb{Z}^2$, and where A is the so-called *discrete Laplace operator*

$$[\mathsf{A}\,u](\mathbf{m}) = 4u(\mathbf{m}) - u(\mathbf{m}+\mathbf{e}_1) - u(\mathbf{m}-\mathbf{e}_1) - u(\mathbf{m}+\mathbf{e}_2) - u(\mathbf{m}-\mathbf{e}_2), \qquad \mathbf{m} \in \mathbb{Z}^2. \tag{4.2}$$

In (5.1), $\mathbf{e}_1 = [1,0]$ and $\mathbf{e}_2 = [0,1]$ are the canonical basis vectors in $\mathbb{Z}^2$. If $f \in L^1(\mathbb{Z}^2)$ and $\sum_{\mathbf{m}\in\mathbb{Z}^2} |f(\mathbf{m})| = 0$, equation (4.1) is well-posed when coupled with a suitable decay condition for u, see [56] for details.

We are primarily interested in the situation where the given function f (the *source*) is supported at a finite number of points which we refer to as *source locations*, and where the function u (the *potential*) is sought at a finite number of points called *target locations*. While the solution technique is described for the equation (4.1) involving the specific operator (5.1), it may readily be extended to a broad range of lattice equations involving constant coefficient elliptic difference operators.

Variations of the equation (4.1) are perhaps best known as a set of equations associated with the discretization of elliptic partial differential equations. However, such equations also emerge in their own right as natural models in a broad range of applications: random walks [32], analyzing the

Ising model (in determining vibration modes of crystals), and many others in engineering mechanics including micro-structural models, macroscopic models, simulating fractures [69, 47] and as models of periodic truss and frame structures [24, 75, 56, 74].

Of particular interest in many of these applications is the situation where the lattice involves local deviations from perfect periodicity due to either broken links, or lattice inclusions. The fast technique described in this chapter can readily be modified to handle such situations, see Chapter 5. It may also be modified to handle equations defined on finite subsets of $\mathbb{Z}^2$, with appropriate conditions (Dirichlet / Neumann / periodic) prescribed on the boundary, see Chapter 5 or [36].

The technique described is a descendant of the Fast Multipole Method (FMM) [44, 42, 45], and, more specifically, of "kernel independent" FMMs [40, 63, 78]. The FMM was originally developed for solving the Poisson equation

$$-\Delta u(\mathbf{x}) = f(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^2, \tag{4.3}$$

which is the continuum analog of (4.1). The FMM exploits the fact that the analytic solution to (**??**) takes the form of a convolution

$$u(\mathbf{x}) = \int_{\mathbb{R}^2} \phi_{\text{cont}}(\mathbf{x} - \mathbf{y})\, f(\mathbf{y})\, d\mathbf{y}, \tag{4.4}$$

where ϕ_{cont} is the fundamental solution of the Laplace operator,

$$\phi_{\text{cont}}(\mathbf{x}) = -\frac{1}{2\pi} \log |\mathbf{x}|. \tag{4.5}$$

If the source function f corresponds to a number of point charges $\{q_j\}_{j=1}^N$ placed at locations $\{\mathbf{x}_j\}_{j=1}^N$, and if the potential u is sought at same set of locations, then the convolution (4.4) simplifies to the sum

$$u_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} \phi_{\text{cont}}(\mathbf{x}_i - \mathbf{x}_j)\, q_j, \qquad i = 1,\, 2,\, \ldots,\, N. \tag{4.6}$$

While direct evaluation of (4.6) requires $O(N^2)$ operations since the kernel is dense, the FMM constructs an approximation to the potentials $\{u_i\}_{i=1}^N$ in $O(N)$ operations. Any requested approximation error ε can be attained, with the constant of proportionality in the $O(N)$ estimate depending only logarithmically on ε.

In the same way that the FMM can be said to rely on the fact that the Poisson equation (4.3) has the explicit analytic solution (4.4), the techniques described in this chapter can be said to rely on the fact that the lattice Poisson equation (4.1) has an explicit analytic solution in the form

$$u(\mathbf{m}) = [\phi * f](\mathbf{m}) = \sum_{\mathbf{n}\in\mathbb{Z}^2} \phi(\mathbf{m}-\mathbf{n})\, f(\mathbf{n}). \tag{4.7}$$

where ϕ is a fundamental solution for the discrete Laplace operator (5.1). This fundamental solution is known analytically [27, 56, 60, 36] via the normalized Fourier integral

$$\phi(\mathbf{m}) = \frac{1}{(2\,\pi)^2} \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} \frac{\cos(t_1 m_1 + t_2 m_2) - 1}{4\,\sin^2(t_1/2) + 4\,\sin^2(t_2/2)}\, dt_1\, dt_2, \qquad \mathbf{m} = [m_1,\, m_2] \in \mathbb{Z}^2. \tag{4.8}$$

The derivation of (4.8) is presented in Section 5.1.3.2.

This chapter presents an adaptation of the original Fast Multipole Method that enables it to handle discrete kernels such as (4.8) and to exploit accelerations that are possible due the geometric restrictions present in the lattice case. The method extends directly to any problem that can be solved via convolution with a discrete fundamental solution. The technique for numerically evaluating (4.8) extends directly to other kernels, see Section 4.2.

While we are not aware of any previously published techniques for rapidly solving the free space problem (4.1) (or, equivalently, for evaluating (4.7)), there exist very fast solvers for the closely related case of lattice Poisson equations defined on rectangular subsets of $\mathbb{Z}^2$ with periodic boundary conditions. Such equations become diagonal when transformed to Fourier space, and may consequently be solved very rapidly via the FFT. The computational time T_{fft} required by such a method satisfies

$$T_{\text{fft}} \sim N_{\text{domain}} \log N_{\text{domain}} \qquad \text{as } N_{\text{domain}} \to \infty, \tag{4.9}$$

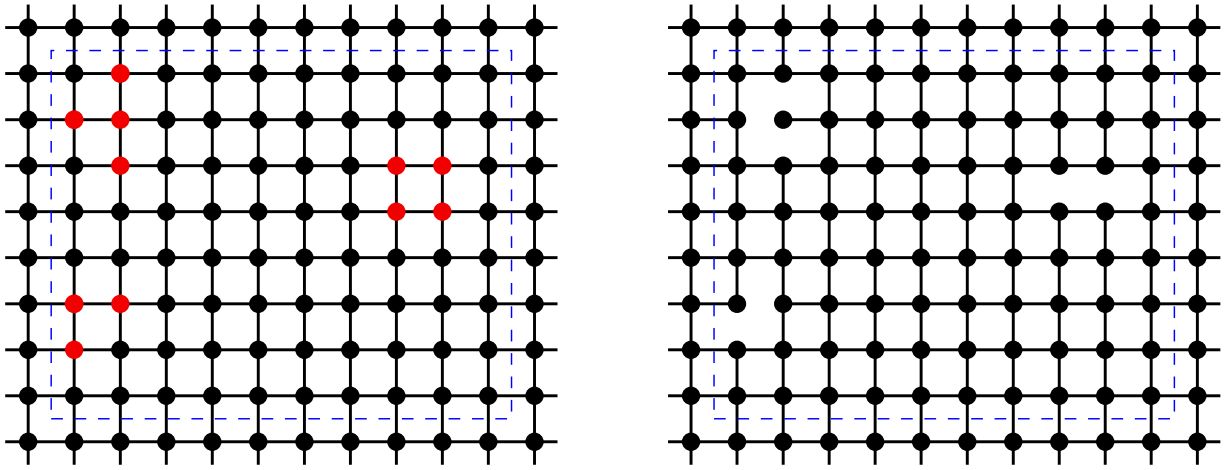
where N_{domain} denotes the number of lattice nodes in the smallest rectangular domain holding all source locations, and where the constant of proportionality is very small. Similar complexity, sometimes without the logarithmic factor, and with fewer restrictions on the boundary conditions, may also be achieved via multigrid methods [73].

The principal contribution of the present work is that the computational time T_{FMM} required by the method described here has asymptotic complexity

$$T_{\text{FMM}} \sim N_{\text{sources}}, \qquad \text{as } N_{\text{sources}} \to \infty, \tag{4.10}$$

where N_{sources} denote the number of lattice nodes that are loaded (assuming that the solution is sought only at the source points). In a situation where the source points are relatively densely distributed in a rectangle, we would have $N_{\text{domain}} \approx N_{\text{sources}}$ and there would be no point in using the new method (in fact, an FFT based method is in this case significantly faster since the constant of proportionality in (4.9) is smaller than that in (4.10)). However, if the source and target points are relatively sparsely distributed in the lattice, then the estimate (4.10) of the new method is clearly superior to that of (4.9) for an FFT based method. As demonstrated in Section 4.8, very significant gains in speed can be achieved. Perhaps even more importantly, much larger problems can be handled since an FFT based method requires that the potential on all N_{domain} nodes be held in memory.

Example: The distinction between N_{domain} in (4.9) and N_{sources} in (4.10) can be illustrated with the toy example shown in Figure 4.1. Figure 4.1(a) illustrates a part of an infinite lattice in which $N_{\text{source}} = 11$ nodes have been loaded. A rectangular domain covering these loads is marked with a blue dashed line and holds $N_{\text{domain}} = 80$ nodes. Clearly $N_{\text{sources}} = 11 \ll 80 = N_{\text{domain}}$. A solution strategy for (4.1) based on the FFT or multigrid would involve all N_{domain} nodes inside the rectangle. In contrast, the lattice fundamental solution allows the solution task to be reduced to evaluating the sum (4.7) which involves an $N_{\text{sources}} \times N_{\text{sources}}$ dense coefficient matrix. Figure 4.1(b) illustrates an infinite lattice in which 7 bars linking $N_{\text{sources}} = 11$ nodes have been removed and we consider the task of finding the potential $u = u(\mathbf{m})$ which satisfies (4.1), and also the boundary condition $\lim_{|\mathbf{m}|\to\infty} |u(\mathbf{m}) - m_1| = 0$. As described in Chapter 5, this problem can be reduced to a linear system of equations involving a dense coefficient matrix of size $N_{\text{sources}} \times N_{\text{sources}}$. This system can be solved via an iterative technique accelerated by the fast summation technique of this paper.



(a) (b)

Figure 4.1: (a) A subset of the infinite lattice $\mathbb{Z}^2$. The $N_{\text{sources}} = 11$ red nodes are loaded. The smallest rectangle holding all sources is marked with a dashed blue line. It has $N_{\text{domain}} = 80$ nodes. (b) An analogous situation in which 7 bars have been excised from the infinite lattice.

4.1 Review of fast summation techniques

In this section, we briefly outline the basic ideas behind the Fast Multipole Method, and then describe the modifications required to evaluate a lattice sum such as (4.7). Our presentation assumes some familiarity with Fast Multipole Methods; for an introduction, see, *e.g.*, [6, 42]. As a model problem, we consider the task of evaluating the sum

$$u_i = \sum_{j=1}^{N} \phi(\mathbf{x}_i - \mathbf{x}_j)\, q_j, \tag{4.11}$$

where $\{\mathbf{x}_i\}_{i=1}^N$ is a set of N points in the plane, where $\{q_i\}_{i=1}^N$ is a set of N given real numbers called *charges*, where $\{q_i\}_{i=1}^N$ is a set of N sought real numbers called *potentials*, and where $\phi : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ is a *kernel function.*

For simplicity, we consider in this review only the case where the sources are more or less uniformly distributed in a computational box Ω in the sense that Ω can be split into equi-sized small boxes, called *leaves*, in such a way that each small box holds about the same number of sources. We let N_{leaf} denote an upper bound for the number of sources held in any leaf. Then the

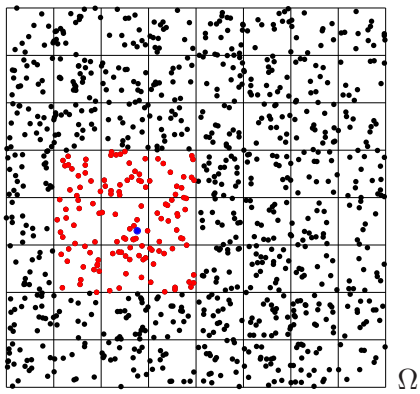


Figure 4.2: Geometry of the N-body problem in Section 4.1. Source i is blue, the sources in J_i^{near} as defined by (4.13) are red.

sum (5.33) can be split into two parts

$$u_i = u_i^{\text{near}} + u_i^{\text{far}},$$

where the *near-field* is defined by

$$u_i^{\text{near}} = \sum_{j \in J_i^{\text{near}} \setminus \{i\}} \phi(\mathbf{x}_i - \mathbf{x}_j)\, q_j, \qquad i = 1,\, 2,\, \ldots,\, N, \tag{4.12}$$

where $J_{\text{i}}^{\text{near}}$ is an index list marking all sources that lie either in the same box as charge i, or in a box that is directly adjacent to the box holding source i,

$$J_i^{\text{near}} = \{j : \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are located in the same leaf box or leaf boxes directly adjacent}\}. \tag{4.13}$$

The definition of J_i^{near} is illustrated in Figure 4.2. The *far-field* is then defined by

$$u_i^{\text{far}} = \sum_{j \notin J_i^{\text{near}}} \phi(\mathbf{x}_i - \mathbf{x}_j)\, q_j, \qquad i = 1,\, 2,\, \ldots,\, N, \tag{4.14}$$

The near-field (4.12) can now be directly evaluated at low cost since at most $9\, N_{\text{leaf}}$ sources are near any given source. In the lattice case, this step could potentially be rendered expensive by the fact that the kernel is known only via the Fourier integral (4.8) which is quite costly to evaluate via quadrature. We describe in Section 4.2 how this step may be accelerated by pre-computing and storing the values of $\phi(\mathbf{m})$ for all small values of $\mathbf{m}$ and then using an asymptotic expansion for large $\mathbf{m}$. We observe that the local evaluation gets particularly effective whenever the number of lattice cells along the side of any leaf box is bounded by some fixed number L of moderate size (say $L \leq 1000$). In this case, there is in the lattice situation only $16\, L^2$ possible relative positions of two charges that are near each other which means that evaluation of the kernel for the near-field calculations amounts to simply a table lookup. (In fact, due to symmetries, only $2\, L^2$ values need to be stored.)

The far-field (4.14) is as in the classical FMM evaluated via the computation of so called *multipole expansions* and *incoming expansions*. These in turn are constructed via a hierarchical procedure on a quad-tree such as the one shown in Figure 4.5. With the development of so called

kernel independent FMMs, the multipole expansions of the original FMM were superseded by more general representations valid for a broad range of kernels. The bulk of this chapter consists of a description of such a kernel independent FMM, adapted to exploit geometrical restrictions imposed in the lattice case. Section 4.3 reviews a technique for compactly representing charges and potentials, and Section 4.4 describes how it can be adapted to the particular case of lattice equations. Section 4.5 introduces notation for handling quad-trees, Section 4.6 describes the so called *translation operators*, then the full lattice FMM is described in Section 4.7.

4.2 Evaluation of the lattice fundamental solution

The numerical evaluation of the function ϕ in (4.8) requires some care since the integrand has a singularity at the origin and gets highly oscillatory when $|\mathbf{m}|$ is large. The latter issue can be handled quite easily since a highly accurate asymptotic expansion of $\phi(\mathbf{m})$ as $|\mathbf{m}| \to \infty$ is known, see Section 4.2.1. When $|\mathbf{m}|$ is small, quadrature and Richardson extrapolation may be used to compute $\phi(\mathbf{m})$ to very high accuracy, see Section 4.2.2. We note that in this regime where $|\mathbf{m}|$ is small, computational speed is of secondary importance since there are only a small number of possible values of $|\mathbf{m}|$, the corresponding values of $\phi(\mathbf{m})$ can be pre-computed and tabulated.

4.2.1 Evaluation of fundamental solution for $|\mathbf{m}|$ large

It has been established (see *e.g.* [27, 28, 55, 56, 60]) that as $|\mathbf{m}| \to \infty$, the fundamental solution ϕ defined by (4.8) has the asymptotic expansion

$$\phi(\mathbf{m}) = -\frac{1}{2\pi}\left(\log|\mathbf{m}| + \gamma + \frac{\log 8}{2}\right) + \frac{1}{24\pi}\frac{m_1^4 - 6m_1^2m_2^2 + m_2^4}{|\mathbf{m}|^6} + \frac{1}{480\pi}\frac{43m_1^8 - 772m_1^6m_2^2 + 1570m_1^4m_2^4 - 772m_1^2m_2^6 + 43m_2^8}{|\mathbf{m}|^{12}} + O(1/|\mathbf{m}|^6). \quad (4.15)$$

The number γ is the Euler constant ($\gamma = 0.577206\cdots$).

For $|\mathbf{m}|$ large, we approximate ϕ by dropping the $O(1/|\mathbf{m}|^6)$ term off the asymptotic expansion. We found that for $|\mathbf{m}| > 30$ the expansion (4.15) is accurate to at least 10^{-12}.

The asymptotic expansion (4.15) is valid for the simple square lattice only. However, there is a simple process for constructing analogous expansions for fundamental solutions associated with a very broad class of constant coefficient elliptic difference operators [60]. The process can be automated and executed using symbolic software such as Maple [56].

4.2.2 Evaluation of fundamental solution for |m| small

When $|\mathbf{m}|$ is small enough that the asymptotic expansion provides insufficient accuracy, we approximate the integral (4.8) using a two-step quadrature procedure: First, the domain $[-\pi,\,\pi]^2$ is split into $n \times n$ equisized boxes where n is an odd number chosen so that each box holds about one oscillation of the integrand (in other words, $n \approx |\mathbf{m}|$). For each box not containing the origin, the integral is approximated using a Cartesian Gaussian quadrature with 20×20 nodes. This leaves us with the task of evaluating the integral

$$g(a) = \frac{1}{(2\,\pi)^2} \int_{-a}^{a} \int_{-a}^{a} \frac{\cos(t_1 m_1 + t_2 m_2) - 1}{4\,\sin^2(t_1/2) + 4\,\sin^2(t_2/2)}\, dt_1\, dt_2,$$

where $a = \pi/n$ denotes the size of the center box. Now observe that

$$g(a) = \sum_{n=0}^{\infty} \left(g\left(\frac{a}{2^n}\right) - g\left(\frac{a}{2^{n+1}}\right)\right) = \sum_{n=0}^{\infty} \frac{1}{(2\,\pi)^2} \int_{\Omega_n} \frac{\cos(t_1 m_1 + t_2 m_2) - 1}{4\,\sin^2(t_1/2) + 4\,\sin^2(t_2/2)}\, dA, \tag{4.16}$$

where

$$\Omega_n = [2^{-n}\,a,\, 2^{-n}\,a]^2 \backslash [2^{-n-1}\,a,\, 2^{-n-1}\,a]^2, \qquad n = 1,\, 2,\, 3,\, \ldots$$

is a sequence of annular domains whose union is the square $[-a,\,a]^2$. All integrals in (4.16) involve non-singular integrands, and can easily be evaluated via Gaussian quadratures. (We split each Ω_n into eight rectangular regions and use a 20×20 point Gaussian quadrature on each.) Using Richardson extrapolation to accelerate the convergence, it turns out that only about 14 terms are needed to evaluate the sum (4.16) to a precision of 10^{-14}.

Remark 26. *The particular integral (4.8) can be evaluated via a short-cut since it is possible to evaluate the integral over t_1 analytically, and then use quadrature only for the resulting (non-singular) integral over t_2, see [56]. Similar tricks are likely possible in many situations involving*

mono-atomic lattices. However, we prefer to not rely on this approach since it does not readily generalize to vector valued problems (such as those associated with mechanical lattice problems) or multi-atomic lattices.

4.3 Outgoing and incoming expansions

In this section, we present techniques for efficiently approximating the far-field u_i^{far} to some desired finite precision ε. The parameter ε can be tuned to balance the computational cost verses the accuracy. In the numerical examples reported in Section 4.8, $\varepsilon = 10^{-10}$.

4.3.1 Interaction ranks

An essential component of the classical FMM is an efficient technique for representing potentials and source distributions via "expansions" of different kinds. To illustrate the concept, let us consider a simplified problem in which a number of sources are placed in a "source box" Ω_τ, and the potential induced by these sources is to be evaluated at a number of locations in a "target box" Ω_σ. The orientation of the boxes is shown in Figure 4.3. To be precise, we suppose that sources $\{q_j^\tau\}_{j=1}^N$ are placed at locations $\{\mathbf{x}_j^\tau\}_{j=1}^N \subset \Omega_\tau$, and that we seek the potentials $\{u_i^\sigma\}_{i=1}^M$ induced at some locations $\{\mathbf{x}_i^\sigma\}_{i=1}^M \subset \Omega_\sigma$,

$$u_i^\sigma = \sum_{j=1}^{N} \Phi(\mathbf{x}_i^\sigma, \mathbf{x}_j^\tau)\, q_j^\tau, \qquad i = 1, 2, \ldots, M. \tag{4.17}$$

In this review of the classical FMM, the kernel Φ is defined by

$$\Phi(\mathbf{x}, \mathbf{y}) = \phi_{\text{cont}}(\mathbf{x} - \mathbf{y}) = -\frac{1}{2\pi} \log |\mathbf{x} - \mathbf{y}|,$$

where ϕ_{cont} is the fundamental solution of the Laplace equation. For convenience, we write (4.17) as a matrix-vector product

$$\mathbf{u}^\sigma = \mathsf{A}^{\sigma,\tau}\, \mathbf{q}^\tau, \tag{4.18}$$

where $\mathbf{u}^\sigma = [u_i^\sigma]_{i=1}^M$ and $\mathbf{q}^\tau = [q_j^\tau]_{j=1}^N$, and where $\mathsf{A}^{\sigma,\tau}$ is the $M \times N$ matrix with entries

$$\mathsf{A}_{ij}^{\sigma,\tau} = \Phi(\mathbf{x}_i^\sigma, \mathbf{x}_j^\tau).$$

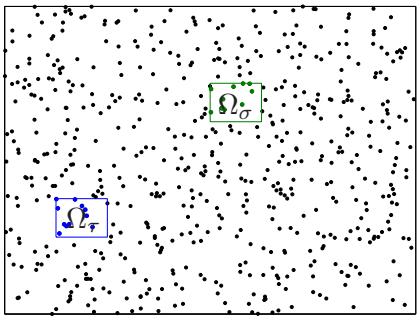


Figure 4.3: Illustration of source box Ω_τ and target box Ω_σ.

A key observation underlying the FMM is that to any finite precision ε, the rank of a matrix such as $\mathsf{A}^{\sigma,\tau}$ is bounded independently of the numbers M and N of targets and sources in the two boxes. In fact, the ε-rank P of $\mathsf{A}^{\sigma,\tau}$ satisfies

$$P \lesssim \log(1/\varepsilon), \quad \text{as } \varepsilon \to 0.$$

The constant of proportionality depends on the geometry of the boxes, but is typically very modest. As a consequence of this rank deficiency, it is possible to factor the matrix $\mathsf{A}^{\sigma,\tau}$, say

$$\begin{array}{ccccc} \mathsf{A}^{\sigma,\tau} & \approx & \mathsf{B} & \mathsf{C}, \\ M \times N & & M \times P & P \times N \end{array} \tag{4.19}$$

and then to evaluate the potential $\mathbf{u}^\tau$ in two steps:

$$\mathbf{v} = \mathsf{C}\,\mathbf{q}^\tau, \qquad \mathbf{u} \approx \mathsf{B}\,\mathbf{v}. \tag{4.20}$$

The cost of evaluating $\mathbf{u}$ via (4.20) is $O((M+N)\,P)$, which should be compared to the $O(M\,N)$ cost of evaluating $\mathbf{u}$ via (4.18).

4.3.2 Formal definitions of outgoing and incoming expansions

In the classical FMM, a "multipole expansion" for a box is a short vector from which the potential caused by all charges in the box can be evaluated; it can be viewed as a compressed representation of all the charges inside the box. In this section, we introduce the "outgoing expansion"

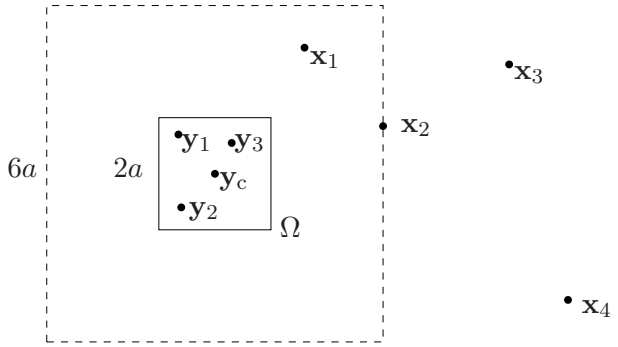


Figure 4.4: Illustration of *well-separated points*. Any point on or outside of the dashed square is *well-separated* from Ω. Consequently, the points $\mathbf{x}_2$, $\mathbf{x}_3$, and $\mathbf{x}_4$ are well-separated from Ω, but $\mathbf{x}_1$ is not.

as a generalization of this idea that allows representations other than classical multipole expansions to be incorporated. The "incoming expansion" is analogously introduced to generalize the concept of a "local expansion."

Well-separated boxes: Let Ω be a box with side length $2a$ and center $\mathbf{c}$ as shown in Figure 4.4. We say that a point $\mathbf{x}$ is *well-separated* from Ω if it lies outside the square of side length $6a$ centered at $\mathbf{c}$. We say that two boxes Ω and Ω' are *well-separated* if every point in Ω' is well-separated from Ω, and vice versa.

Outgoing expansion: Let Ω be a box containing a set of sources. We say that a vector $\hat{\mathbf{q}}$ is an *outgoing expansion* for Ω if the potential caused by the sources in Ω can be reconstructed from $\hat{\mathbf{q}}$ to within precision ε at any point that is well-separated from Ω.

Incoming expansion: Let Ω be a box in which a potential has been induced by a set of sources located at points that are well-separated from Ω. We say that a vector $\hat{\mathbf{u}}$ is an *incoming expansion* for Ω if u can be reconstructed from $\hat{\mathbf{u}}$ to within precision ε.

4.3.3 Charge basis

The cost of computing a factorization such as (4.19) using a generic linear algebraic technique such as QR is $O(M\,N\,P)$, which would negate any savings obtained when evaluating the matrix-

vector product (unless a very large number of matrix-vector products involving the same source and target locations is required). Fortunately, it is possible in many environments to construct such factorizations much faster. The classical FMM uses multipole expansions. As an alternative, an approach based on so-called "proxy charges" has recently been developed [78]. It has been demonstrated [63, 65] that for any given box Ω, it is possible to find a set of locations $\hat{\mathbf{Y}} = \{\hat{\mathbf{y}}_p\}_{p=1}^{P} \subset \Omega$ with the property that sources placed at these points can to high accuracy replicate any potential caused by a source distribution in Ω. The number of points P required is given in Table 4.1. To be precise, given any set of points $\mathbf{Y} = \{\mathbf{y}_j\}_{j=1}^{N} \subset \Omega$ and any sources $\mathbf{q} = \{q_j\}_{j=1}^{N}$, we can find "equivalent charges" $\hat{\mathbf{q}} = \{\hat{q}_p\}_{p=1}^{P}$ such that

$$\sum_{j=1}^{N} \phi_{\text{cont}}(\mathbf{x} - \mathbf{y}_j)\, q_j \approx \sum_{p=1}^{P} \phi_{\text{cont}}(\mathbf{x} - \hat{\mathbf{y}}_p)\, \hat{q}_p, \tag{4.21}$$

whenever $\mathbf{x}$ is well-separated from Ω. The approximation (4.21) holds to some preset (relative) precision ε. Moreover, the map from $\mathbf{q}$ to $\hat{\mathbf{q}}$ is linear, and there exists a matrix $\mathsf{T}_{\text{ofs}} = \mathsf{T}_{\text{ofs}}(\hat{\mathbf{Y}}, \mathbf{Y})$ such that

$$\hat{\mathbf{q}} = \mathsf{T}_{\text{ofs}}\, \mathbf{q}, \tag{4.22}$$

where "ofs" is an abbreviation of "outgoing [expansion] from sources."

	ϵ			
l	10^{-6}	10^{-8}	10^{-10}	10^{-13}
1/32	19	27	37	49
1/16	19	27	36	49
1/4	19	28	37	49
1/2	21	29	37	51

Table 4.1: The number of points P required to replicate the field to accuracy ϵ for a box Ω with side length l.

We say that the points $\{\hat{\mathbf{y}}_p\}_{p=1}^{P}$ form an *outgoing skeleton* for Ω, and that the vector $\hat{\mathbf{q}}$ is an outgoing expansion of Ω.

In addition, we can find an *incoming skeleton* $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_p\}_{p=1}^{P} \subset \Omega$ with the property that any incoming potential in Ω can be interpolated from its values on the incoming skeleton. To be precise,

suppose that $U = U(\mathbf{x})$ is a potential caused by sources that are well-separated from Ω, and that $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^M$ is an arbitrary set of points in Ω. Then there exists a matrix $\mathsf{T}_{\text{tfi}} = \mathsf{T}_{\text{tfi}}(\mathbf{X}, \hat{\mathbf{X}})$ ('tfi" stands for "targets from incoming [expansion]") such that

$$\mathbf{u} = \mathsf{T}_{\text{tfi}}\, \hat{\mathbf{u}},$$

where

$$\mathbf{u} = [U(\mathbf{x}_i)]_{i=1}^M, \qquad \text{and} \qquad \hat{\mathbf{u}} = [U(\hat{\mathbf{x}}_p)]_{p=1}^P.$$

When the kernel Φ is symmetric in the sense that $\Phi(\mathbf{x} - \mathbf{y}) = \Phi(\mathbf{y} - \mathbf{x})$ for all $\mathbf{x}$ and $\mathbf{y}$, any outgoing skeleton is also an incoming skeleton,

$$\hat{\mathbf{X}} = \hat{\mathbf{Y}}.$$

Moreover, if the target points equal the source points so that $\mathbf{X} = \mathbf{Y}$, then

$$\mathsf{T}_{\text{tfi}} = (\mathsf{T}_{\text{ofs}})^*.$$

Applied to the situation described in Section 4.3.1, where a set of sources were placed in a source box Ω_τ, and we sought to evaluate the potential induced at a set of target points in a box Ω_σ, the claims of this section can be summarized by saying that $\mathsf{A}^{\sigma,\tau}$ admits an approximate factorization

$$\begin{array}{ccccccc} \mathsf{A}^{\sigma,\tau} & \approx & \mathsf{T}^{\sigma}_{\text{tfi}} & & \mathsf{T}^{\sigma,\tau}_{\text{ifo}} & & \mathsf{T}^{\tau}_{\text{ofs}} \\ M \times N & & M \times P & & P \times P & & P \times N \end{array}$$

where the middle factor is simply a subsampling of the original kernel function

$$\mathsf{T}^{\sigma,\tau}_{\text{ifo},pq} = \Phi(\hat{\mathbf{x}}^{\sigma}_p, \hat{\mathbf{x}}^{\tau}_q).$$

Remark 27. *For solving multiple problems involving different source and load distributions that involve the same kernel, one set of skeleton points may be used for all problems by choosing the skeleton points to lie on the boundary of Ω_σ and Ω_τ. The interpolation matrices T_{tfi} and T_{ofs} need be constructed for each unique set of source and load distributions using the techniques from [21]. In Section 4.4, we describe this generalization of the skeletonization process in more detail for the lattice fundamental solution.*

4.4 Constructing charge bases for the lattice fundamental solution

In this section, we describe how to construct the charge bases for the lattice fundamental solution defined by (5.31).

From potential theory, we know that to capture the interaction between a set of source points $\{\mathbf{m}_j^\tau\}_{j=1}^N$ in box Ω_τ and all points far from Ω_τ, it is enough to capture the interaction between the source points and a set of "proxy" points F that lie densely on the boundary of a box that is concentric to Ω_τ and has a boundary that is well-separated from Ω_τ.

We choose the skeleton points to be a subset of the set of all points Y that lie on the boundary of box τ. Either rank revealing QR factorization [46] or factorization techniques from [21] are applied to the matrix $\mathsf{A}^{F,Y}$ (whose entries are given by $\mathsf{A}_{i,j}^{F,Y} = \Phi(\mathbf{m}_i^F - \mathbf{m}_j^Y)$) to determine the rank P and which P points make up the set of skeleton points $\hat{Y}$ of Ω_τ.

Using the skeleton points and the techniques from [21], we find the $P \times N$ matrix T_{ofs} such that

$$\|\mathsf{A}^{F,\tau} - \mathsf{A}^{F,\hat{Y}}\mathsf{T}_{\text{ofs}}\| < \epsilon. \tag{4.23}$$

We use a similar technique to find the incoming skeleton points and the translation operator T_{tfi}.

Remark 28. *Because of the smoothness of the kernel, it is not required to use all the points on the boundary of the well-separated box as "proxy" points. We found it is enough to take 40 points per edge to approximate the far field with accuracy* 10^{-10}.

4.5 Tree structure

The separation of variables in the kernel that was described in Section 4.3 is all that is needed to effectively evaluate a potential field whenever the set of target locations is well-separated from the set of source locations. When the two sets coincide, we need to tessellate the box containing them into smaller boxes, and use the expansion only for interactions between boxes that are well-separated. In this section, we describe the simplest such tessellation.

Suppose that we are given a set of points $\{\mathbf{x}_i\}_{i=1}^N$ in a box Ω. Given an integer N_{leaf}, we pick the smallest integer L such that when the box Ω is split into 4^L equisized smaller boxes, no box holds more than N_{leaf} points. These 4^L equisized small boxes form the *leaves* of the tree. We merge the leaves by sets of four to form 4^{L-1} boxes of twice the side-length, and then continue merging by pairs until we recover the original box Ω, which we call the *root*.

The set consisting of all boxes of the same size forms what we call a *level*. We label the levels using the integer $\ell = 0, 1, 2, \ldots, L$, with $\ell = 0$ denoting the root, and $\ell = L$ denoting the leaves. See Figure 4.5.

Definition 1. *Let τ be a box in a hierarchical tree.*

- *The parent of τ is the box on the next coarser level that contains τ.*
- *The children of τ is the set $\mathcal{L}_\tau^{\text{child}}$ of boxes whose parent is τ.*
- *The neighbors of τ is the set $\mathcal{L}_\tau^{\text{nei}}$ of boxes that are on the same level as τ and are directly adjacent to it.*
- *The interaction list of τ is the set $\mathcal{L}_\tau^{\text{int}}$ of all boxes σ such that:*

  (1) σ and τ are on the same level.

  (2) σ and τ are not directly adjacent.

  (3) The parents of σ and τ are directly adjacent.

Example: For the tree shown in Figure 4.5, we have, *e.g.*,

$$\mathcal{L}_{14}^{\text{child}} = \{54, 55, 56, 57\},$$

$$\mathcal{L}_{23}^{\text{nei}} = \{22, 24, 25, 26, 28\},$$

$$\mathcal{L}_{59}^{\text{nei}} = \{36, 37, 48, 58, 60, 61, 70, 72\},$$

$$\mathcal{L}_{7}^{\text{int}} = \{11, 13, 14\colon 21\},$$

$$\mathcal{L}_{37}^{\text{int}} = \{22\colon 29, 30\colon 33, 38\colon 41, 47, 49, 54\colon 57, 60, 61, 71, 72, 73\}.$$

Level $\ell = 0$

Box $\tau = 1$

Level $\ell = 1$

Box $\tau = 3$	Box $\tau = 5$
Box $\tau = 2$	Box $\tau = 4$

Level $\ell = 2$

$\tau = 11$	$\tau = 13$	$\tau = 19$	$\tau = 21$
$\tau = 10$	$\tau = 12$	$\tau = 18$	$\tau = 20$
$\tau = 7$	$\tau = 9$	$\tau = 15$	$\tau = 17$
$\tau = 6$	$\tau = 8$	$\tau = 14$	$\tau = 16$

Level $\ell = 3$

35	37	51	53	67	69	83	85
34	36	50	52	66	68	82	84
31	33	47	49	63	65	79	81
30	32	46	48	62	64	78	80
27	29	43	45	59	61	75	77
26	28	42	44	58	60	74	76
23	25	39	41	55	57	71	73
22	24	38	40	54	56	70	72

Figure 4.5: A binary tree with 4 levels of uniform refinement.

For the moment, we are assuming that the given point distribution is sufficiently uniform that all the leaves hold roughly the same number of points. In this case,

$$L \sim \log \frac{N}{N_{\text{leaf}}}.$$

For non-uniform distributions of points, a uniform subdivision of Ω into 4^L boxes of equal length would be inefficient since many of the leaves would hold few or no points. In such cases, adaptive subdivisions should be used [14].

4.6 Translation operators

In the FMM, five different so called *translation operators* that construct or translate outgoing or incoming expansions are required. We will, in this section, describe how to construct them, but we first list which operators we need:

$\mathsf{T}^{\tau}_{\text{ofs}}$ *The outgoing from sources translation operator:* Let τ denote a box holding a set of sources whose values are listed in the vector $\mathbf{q}^{\tau}$. The outgoing expansion $\hat{\mathbf{q}}^{\tau}$ of τ is then constructed via

$$\hat{\mathbf{q}}^{\tau} = \mathsf{T}^{\tau}_{\text{ofs}}\, \mathbf{q}^{\tau}.$$

$\mathsf{T}^{\tau,\sigma}_{\text{ofo}}$ *The outgoing from outgoing translation operator:* Suppose that a child σ of a box τ holds a source distribution represented by the outgoing expansion $\hat{\mathbf{q}}^{\sigma}$. The far-field caused by these sources can equivalently be represented by an outgoing representation $\hat{\mathbf{q}}^{\tau}$ of the parent, constructed via

$$\hat{\mathbf{q}}^{\tau} = \mathsf{T}^{\tau,\sigma}_{\text{ofo}}\, \hat{\mathbf{q}}^{\sigma}.$$

$\mathsf{T}^{\tau,\sigma}_{\text{ifo}}$ *The incoming from outgoing translation operator:* Suppose that τ and σ are two well-separated boxes, and that σ holds a source distribution represented by an outgoing expansion $\hat{\mathbf{q}}^{\sigma}$. Then the field in τ caused by these sources can be represented by an incoming expansion

$\hat{\mathbf{u}}^\tau$ that is constructed via

$$\hat{\mathbf{u}}^\tau = \mathsf{T}^{\tau,\sigma}_{\text{ifo}}\, \hat{\mathbf{q}}^\sigma.$$

$\mathsf{T}^{\tau,\sigma}_{\text{ifi}}$ *The incoming from incoming translation operator:* Suppose that τ is the parent of a box σ. Suppose further that the incoming expansion $\hat{\mathbf{u}}^\tau$ represents a potential in τ caused by sources that are all well-separated from τ. Then these sources are also well-separated from σ, and the potential in σ can be represented via an incoming expansion $\hat{\mathbf{u}}^\sigma$ given by

$$\hat{\mathbf{u}}^\sigma = \mathsf{T}^{\tau,\sigma}_{\text{ifi}}\, \hat{\mathbf{u}}^\sigma.$$

$\mathsf{T}^{\tau}_{\text{tfi}}$ *The targets from incoming translation operator:* Suppose that τ is a box whose incoming potential is represented via the incoming representation $\hat{\mathbf{u}}^\tau$. Then the potential at the actual target points are constructed via

$$\mathbf{u}^\tau = \mathsf{T}^{\tau}_{\text{tfi}}\, \hat{\mathbf{u}}^\tau.$$

Techniques for constructing the matrix T^{τ}_{ofs} were described in Section 4.4. Since in our case, the kernel is symmetric (*i.e.* $\phi(\mathbf{x} - \mathbf{y}) = \phi(\mathbf{y} - \mathbf{x})$ for all $\mathbf{x}$ and $\mathbf{y}$), these techniques immediately give us the targets-from-incoming translation operator as well, since

$$\mathsf{T}^{\tau}_{\text{tfi}} = \left(\mathsf{T}^{\tau}_{\text{ofs}}\right)^*.$$

We next observe that when charge bases are used, the outgoing-to-incoming translation operator is simply a sampling of the kernel function,

$$\mathsf{T}^{\tau,\sigma}_{\text{ifo},pq} = \phi(\hat{\mathbf{x}}^\tau_p - \hat{\mathbf{x}}^\sigma_q), \qquad p,\, q = 1,\, 2,\, 3,\, \ldots,\, P,$$

where $\{\hat{\mathbf{x}}^\tau_i\}_{i=1}^P$ and $\{\hat{\mathbf{x}}^\sigma_j\}_{j=1}^P$ are the locations of the skeleton points of τ and σ, respectively.

All that remains is to construct $\mathsf{T}^{\tau,\sigma}_{\text{ofo}}$ and $\mathsf{T}^{\sigma,\tau}_{\text{ifi}}$. In fact, since the kernel is symmetric,

$$\mathsf{T}^{\sigma,\tau}_{\text{ifi}} = \left(\mathsf{T}^{\sigma,\tau}_{\text{ofo}}\right)^*,$$

and all that actually remains is to construct the matrices $\mathsf{T}_{\text{ofo}}^{\tau,\sigma}$. To this end, let $\{\sigma_i\}_{i=1}^{l}$ denote children of box τ. The construction of $\mathsf{T}_{\text{ofo}}^{\tau,\sigma_i}$ closely resembles the construction of the T_{ofs} operator described in Section 4.4. Instead of choosing the skeleton points from the set of all points on the boundary of τ as was done in the construction of T_{ofs}, we choose the skeleton points for τ to be a subset of the skeleton points of its children, $Y = [\hat{Y}^{\sigma_1}, \ldots, \hat{Y}^{\sigma_l}]$. As in Section 4.4, we define a set of "proxy" points F that are well-separated from τ and use a factorization technique such as rank revealing QR to determine which points in Y make up the set of skeleton points $\hat{Y}$. Using the techniques from [21], we find the interpolation matrix S such that

$$\|\mathsf{A}^{P,Y} - \mathsf{A}^{P,\hat{Y}}\mathsf{S}\| < \epsilon.$$

The translation operator $\mathsf{T}_{\text{ofo}}^{\tau,\sigma_1}$ is then defined via $\mathsf{T}_{\text{ofo}}^{\tau,\sigma_1} = \mathsf{S}(:, 1 : k_1)$ where k_1 is the number of skeleton points of σ_1, $\mathsf{T}_{\text{ofo}}^{\tau,\sigma_2} = \mathsf{S}(:, (k_1+1) : (k_1+k_2))$ where k_2 is the number of skeleton points of σ_2, etc.

4.7 A lattice Fast Multipole Method

While the classical FMM derives so-called "translation operators" based on asymptotic expansions of the kernel function, the method of we propose determines these operators computationally. In this regard, it is similar to "kernel independent FMMs" such as [1, 54, 78]. Since the kernel is translation invariant, the computations need be carried out only for a single box on each level. Thus the construction of the translation operators is very inexpensive (less than linear complexity).

4.7.1 Precomputing skeletons and translation operators

For each level l, we define a "model" box which is centered at the origin and has the same size as the boxes on level l. The skeleton points and the translation operators are found with respect to the model box.

To illustrate the concept, suppose that we are given a source f that is non-zero set of points $\{\mathbf{m}_i\}_{i=1}^{N}$ in a box Ω. We seek the potential at the source points.

The pre-computation consist of the following steps:

(1) Divide Ω into the tree structure as described in Section 4.5.

(2) Construct the lists described in Section 4.5.

(3) Construct the skeleton points, T_{ofo}, and T_{ifi} translation operators. At the lowest level L, we construct the skeleton points for the level L model box using the procedure described in Section 4.4. For each level $i < L$, we take four copies of the skeleton points for level $i+1$ shifting them so that each copy makes up one quadrant of the model box for level i. The skeleton points and the translation operators T_{ofo} and T_{ifi} are constructed using the technique described in Section 4.6.

(4) Construct the T_{ifo} translation operators. For each level $i > 1$, we construct the T_{ifo} translation operators for the model box. We assume that the model box is completely surrounded with boxes such that the interaction list has the maximum number of boxes possible which is 42. Let $\hat{Y}$ be the outgoing skeleton points and $\hat{X}$ be the incoming skeleton points for the model box on level i. For each $j \leq 42$, we shift $\hat{X}$ to be centered at the j^{th} possible location for a box on the interaction list and define

$$\tilde{\mathsf{T}}^{j}_{\text{ifo}} = \mathsf{A}^{\hat{X},\hat{Y}} \tag{4.24}$$

Remark 29. *In computing the sum, described in Section 4.7.2, it is easy to use the pre-computed translation operators. For example, given a box τ that has a box σ on the interaction list, we identify which j location σ is in relative to τ and define $\mathsf{T}^{\sigma,\tau} = \tilde{\mathsf{T}}^{j}_{\text{ifo}}$.*

Remark 30. *For leaf boxes of size less than 8×8 on level l, we utilize the fact that there are a finite number of points inside the box that are also in $\mathbb{Z}^2$ and construct the translation operator $\mathsf{T}^{l}_{\text{ofs}}$ for the model box assuming the source points are dense. For each box τ on level l with N^{τ} sources, we construct an index vector J^{τ} that notes the locations of the sources $\{\mathbf{m}^{\tau}_{j}\}_{j=1}^{N^{\tau}}$ in the dense lattice. We define $\mathsf{T}^{\tau}_{\text{ofs}} = \mathsf{T}^{l}_{\text{ofs}}(:, J^{\tau})$. The translation operator $\mathsf{T}^{\tau}_{\text{tfi}}$ is constructed in a similar manner.*

4.7.2 Application

We have now assembled the tools for computing the sum (4.7) through two passes through the hierarchical tree; one upwards, and one downwards.

(1) Sweep over all leaf boxes τ. For each box, construct its outgoing representation from the values of the sources inside it:

$$\hat{\mathbf{q}}^{\tau} = \mathsf{T}^{\tau}_{\text{ofs}}\, \mathbf{q}(J^{\tau}).$$

(2) Sweep over all non-leaf boxes τ, going from finer to coarser levels. Merge the outgoing expansions of the children to construct the outgoing expansion for τ,

$$\hat{\mathbf{q}}^{\tau} = \sum_{\sigma \in \mathcal{L}^{\tau}_{\text{children}}} \mathsf{T}^{\tau,\sigma}_{\text{ofo}} \hat{\mathbf{q}}^{\sigma}.$$

(3) Loop over all boxes τ. For each box, collect the contributions to its incoming expansion from boxes in its interaction list:

$$\hat{\mathbf{u}}^{\tau} = \sum_{\sigma \in \mathcal{L}^{\tau}_{\text{int}}} \mathsf{T}^{\tau,\sigma}_{\text{ifo}}\, \hat{\mathbf{q}}^{\sigma}.$$

(4) Loop over all parent boxes τ, going from coarser levels to finer. For each box τ, loop over all children σ of τ, and broadcast the the incoming expansion of τ to the incoming expansions of σ:

$$\hat{\mathbf{u}}^{\sigma} = \hat{\mathbf{u}}^{\sigma} + \mathsf{T}^{\sigma,\tau}_{\text{ifi}}\, \hat{\mathbf{u}}^{\tau}.$$

(5) Sweep over all leaf nodes τ. For each node, form the potential $\mathbf{u}^{\tau}$ by evaluating the incoming representation and directly adding the contributions from the sources inside τ and in all boxes that are not well-separated from τ:

$$\mathbf{u}^{\tau} = \mathbf{u}(J^{\tau}) = \mathsf{T}^{\tau}_{\text{tfi}}\, \hat{\mathbf{u}}^{\tau} + \mathsf{A}(J^{\tau}, J^{\tau})\, \mathbf{q}(J^{\tau}) + \sum_{\sigma \in \mathcal{L}^{\tau}_{\text{nei}}} \mathsf{A}(J^{\tau}, J^{\sigma})\, \mathbf{q}(J^{\sigma}).$$

4.7.3 Asymptotic complexity of the proposed scheme

Since the kernel (4.8) is separable, the cost of computing the skeleton points and the translation operators on any level of the quad-tree is $O(P\,M\,|F|)$ where P is the number of skeleton points, M is the number of points on the boundary the box, and $|F|$ is the number of well-separated proxy nodes used [21]. The cost of solving the least squares problem (4.23) to find the matrix T_{ofs} for a leaf box is $O(P^2\,|F| + N\,P\,|F|)$ where N is the number of loaded points in the box. Hence, the total complexity of the lattice FMM is $O(N_{\text{source}})$.

Also notice that the memory needed to store the precomputed information is $O(N_{\text{source}})$.

4.8 Numerical examples

In this section, we show that the lattice FMM speed compares favorably to FFT based techniques except for situations where the source points populate the majority of some computational box. We also show that the memory required to use the lattice FMM is linear with respect to the number of source terms.

All experiments are run on a Dell desktop computer with 2GB of RAM and an Intel Pentium 4 3.4GHz dual processor. The method was run at a requested relative precision of 10^{-10}. The techniques were implemented rather crudely in Matlab, which means that significant further gains in speed should be achievable.

We consider the lattice Poisson problem

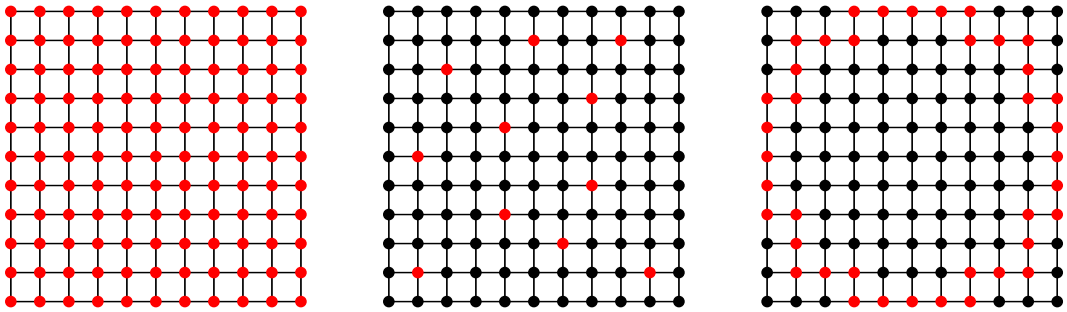
$$[\mathsf{A}u](\mathbf{m}) = f(\mathbf{m}), \tag{4.25}$$

where the points where $f(\mathbf{m})$ is non-zero are confined to an $n \times n$ square subdomain Ω of $\mathbb{Z}^2$. The FFT produces a slightly different solution than the lattice FMM since it enforces periodic boundary conditions, but this is not important for our purposes. We suppose throughout that n is a power of two to make the comparison as favorable to the FFT as possible. We let T_{fft} denote the time required by the FFT, and T_{FMM} the time for the FMM.

In the first experiment, we suppose that every node in the lattice is loaded, see Figure 4.6(a), so that $N_{\text{source}} = N_{\text{domain}} = n^2$. In this case, we expect

$$T_{\text{fft}} \sim n^2 \log(n), \qquad \text{and} \qquad T_{\text{FMM}} \sim n^2,$$

and the purpose of the numerical experiment is simply to see how the constants of proportionality compare. Figure 4.7(a) provides the answer. We see that the FMM is slower by roughly one order of magnitude.



(a) Dense (b) Random (c) Embedded Circle

Figure 4.6: Illustration of load distributions for the three experiments. Red dots are the source points.

In the next three experiments, we suppose that f is only sparsely supported in the domain Ω, so that $N_{\text{source}} \ll N_{\text{domain}}$. In this case, we expect

$$T_{\text{fft}} \sim n^2 \log(n), \qquad \text{and} \qquad T_{\text{FMM}} \sim N_{\text{source}}.$$

In the second experiment, we suppose that n loads distributed according to a uniform random distribution throughout the domain, see Figure 4.6(b). Figure 4.8(a) provides the measured times. It confirms our expectation that T_{FMM} does not depend on N_{domain}, and indeed, that the FMM can handle a situation with $n = 10^6$ loaded nodes in a domain involving $N_{\text{domain}} = 10^{12}$ lattice nodes. Figure 4.8(b) illustrates the memory (in KB) per source point (M/N_{source}) required for storing the pre-computation information. It confirms our expectation that the memory (in KB) required for storing the pre-computation information depends linearly with respect to N_{source}.

In the third experiment, we distribute the load on a circle inscribed in the square Ω, see Figure 4.6(c), in such a way that $N_{\text{source}} = \alpha\, n$ nodes are loaded, for $\alpha = 1,\ 1/4,\ 1/16,\ 1/64$. Figure 4.9(a) provides the time measurements and again confirms our expectation that the T_{FMM} is not dependent on N_{domain}.

In the final experiment, we fix the domain to be sized 2048×2048 and increase the number of body loads distributed according to a uniform distribution. Figure 4.10(a) provides the time measurements in comparison with the FFT. It illustrates that for sources occupying less than 0.39% of the domain (corresponding to $16,384$ sources) the lattice FMM is the faster method.

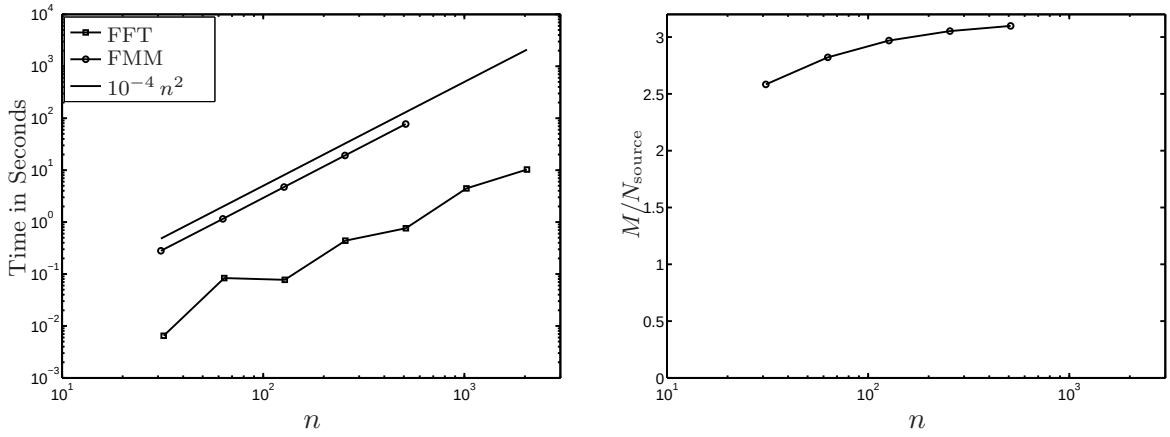


Figure 4.7: Computational profile for a dense source distribution in a $n \times n$ domain. Computational times using the lattice FMM and FFT are reported (a). The memory M (in KB) per source point (M/N_{source}) used in storing the precomputed information for the lattice FMM are reported (b).

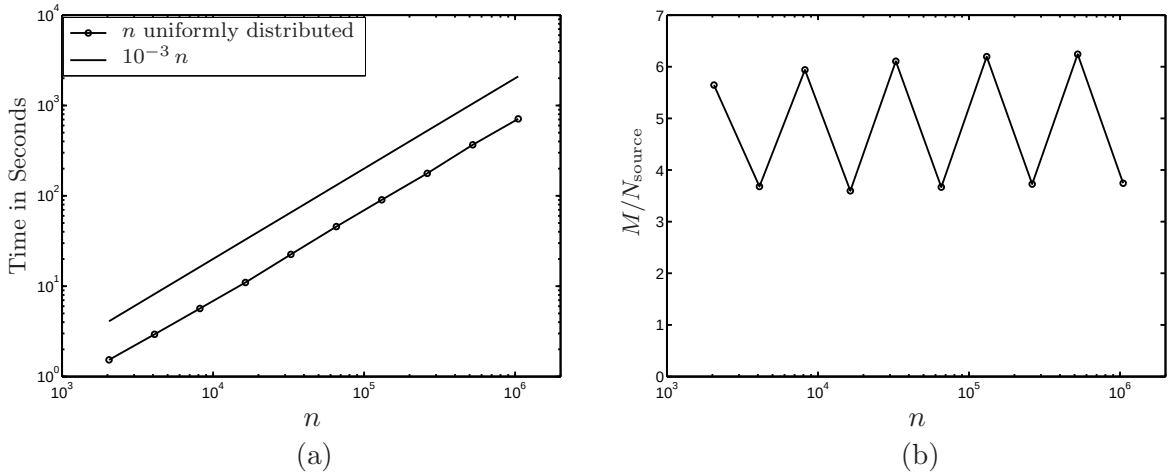
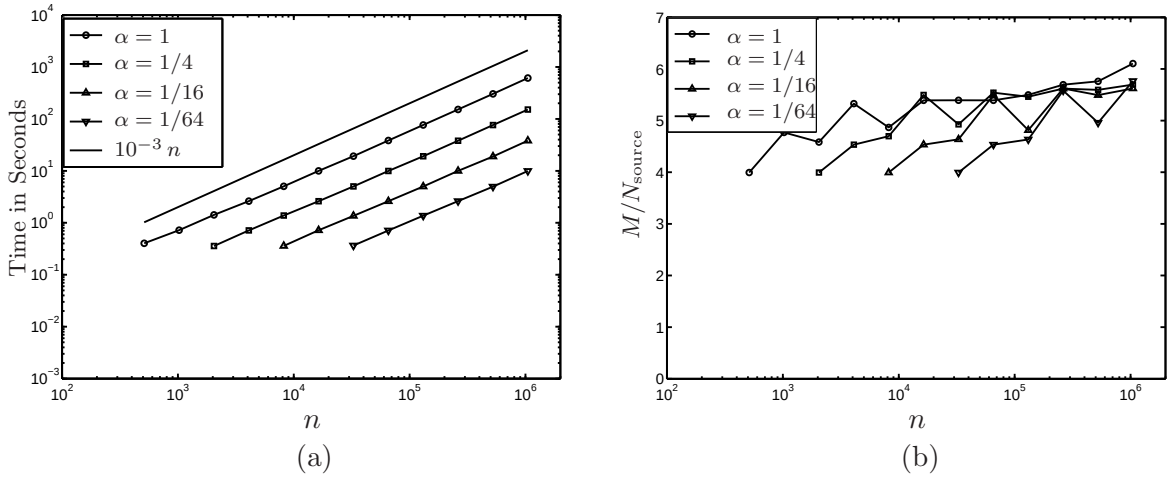
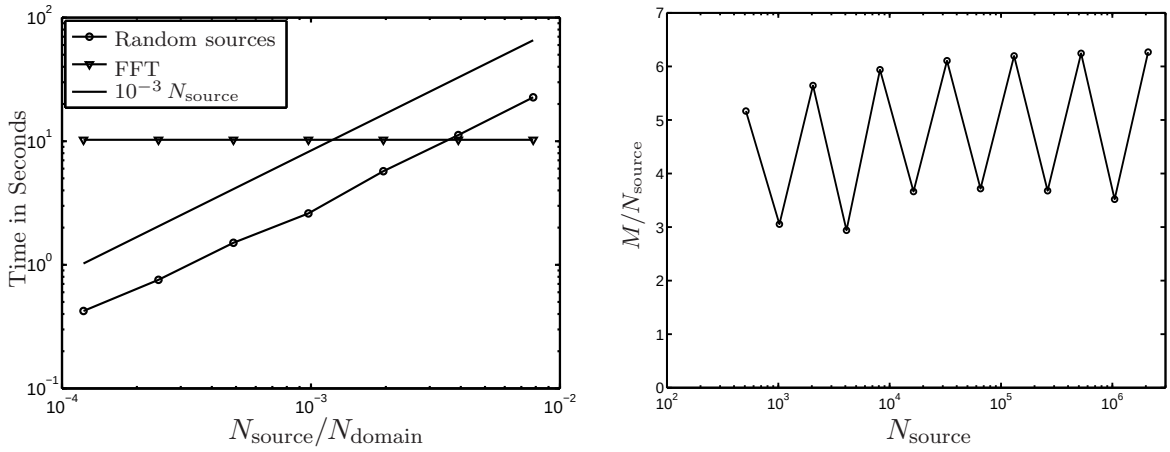


Figure 4.8: Computational profile for a n source points distributed via uniform random distribution in a $n \times n$ domain. Computational times using the lattice FMM are reported (a). The memory M (in KB) per source point (M/N_{source}) used in storing the precomputed information for the lattice FMM are reported (b).



(a) (b)

Figure 4.9: Computational profile for αn sources lie on a circle embedded in an $n \times n$ domain. Computational times using the lattice FMM are reported (a). The memory M (in KB) per source point (M/N_{source}) used in storing the precomputed information for the lattice FMM are reported (b).



(a) (b)

Figure 4.10: Computational profile for a fixed 2048×2048 lattice domain. Computational times using the lattice FMM and the FFT are reported (a). The memory M (in KB) per source point (M/N_{source}) used in storing the precomputed information for the lattice FMM are reported (b).

4.9 Concluding remarks

The chapter presents a kernel independent FMM for solving Poisson problems defined on the integer lattice $\mathbb{Z}^2$. For simplicity of presentation, we focused on equations involving the discrete Laplace operator. Techniques for evaluating the corresponding lattice fundamental solutions are presented. The complexity of the proposed method is $O(N_{\text{source}})$ where N_{source} is the number of locations in $\mathbb{Z}^2$ subjected to body loads.

Numerical experiments demonstrate that for problems where the body loads are sparsely distributed in a computational box the proposed method is faster and more robust than the FFT. For instance, it was demonstrated that using a standard desktop PC, a lattice Poisson equation on a lattice with $N_{\text{domain}} = 10^{12}$ nodes, of which $N_{\text{source}} = 10^6$ were loaded, was solved to ten digits of accuracy in three minutes. It should be noted that this problem is about six orders of magnitude larger than the largest Poisson problem that can be handled via the FFT. Also, it was demonstrated for a lattice Poisson problem in a domain with $N_{\text{domain}} = 4,194,304$ nodes, the lattice FMM is faster than the FFT when the number of loaded points is less than $N_{\text{source}} = 16,384$.

Additionally, the lattice FMM is a key tool for other solution techniques on lattice domains. In particular, the fast direct solution technique for lattice boundary value problems with inclusions presented in the next chapter exist in large part due the existence of the lattice FMM.

Chapter 5

Fast direct solution techniques for solving elliptic difference equations defined on lattices

This chapter describes efficient techniques for solving elliptic difference equations defined either on the integer lattices $\mathbb{Z}^2$ or $\mathbb{Z}^3$, or on finite sub-domains of those lattices. The techniques are applicable to a wide range of difference equations, but to keep the presentation simple, we focus on problems in two dimensions involving the well known *discrete Laplace operator* A which for $\mathbf{m} \in \mathbb{Z}^2$ takes the form

$$[\mathsf{A}\, u](\mathbf{m}) = 4u(\mathbf{m}) - u(\mathbf{m} + \mathbf{e}_1) - u(\mathbf{m} - \mathbf{e}_1) - u(\mathbf{m} + \mathbf{e}_2) - u(\mathbf{m} - \mathbf{e}_2). \tag{5.1}$$

In (5.1), $\mathbf{e}_1 = [1, 0]$ and $\mathbf{e}_2 = [0, 1]$ are the canonical basis vectors in $\mathbb{Z}^2$, and $u = u(\mathbf{m})$ is a real valued function on $\mathbb{Z}^2$. We will briefly review techniques for the free space equation

$$[\mathsf{A}u](\mathbf{m}) = f(\mathbf{m}), \qquad \mathbf{m} \in \mathbb{Z}^2 \tag{5.2}$$

and then describe techniques for boundary value problems of the form

$$\begin{cases} [\mathsf{A}u](\mathbf{m}) &= 0, & \mathbf{m} \in \Omega, \\ u(\mathbf{m}) &= g(\mathbf{m}), & \mathbf{m} \in \Gamma. \end{cases} \tag{5.3}$$

In (5.3), Ω is a subset of $\mathbb{Z}^2$ with boundary Γ. A precise definition of what we mean by the *boundary* of a lattice domain is given in Section 5.3. We typically refer to the data function f as a *source,* and the unknown function u as a *potential.*

Equations of the forms (5.2) and (5.3) are perhaps best known as equations arising upon finite difference discretizations of Poisson's and Laplace's equations, but they arise naturally in a

wide range of applications. To name just a few examples, equations (5.2) or (5.3) or variations of them appear directly in models of random walks [32], in analyzing the Ising model, in determining vibration modes of crystals, and in modeling QCD [23, 67]. Additional examples arise in engineering mechanics as micro-structural models, macroscopic models, simulating fractures [69, 47] and as models of periodic truss and frame structures [24, 75, 56, 74].

The techniques described in this chapter can be viewed as an adaptation to the discrete case of a set of analytical and numerical methods for efficiently solving the corresponding continuum equations. For instance, equation (5.2) has a continuum analog in the free space Laplace equation

$$[-\Delta\, u](\mathbf{x}) = f(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^2, \tag{5.4}$$

coupled with suitable decay conditions at infinity. The analytic solution of (5.4) is

$$u(\mathbf{x}) = [\Phi * f](\mathbf{x}) = \int_{\mathbb{R}^2} \Phi(\mathbf{x}-\mathbf{y})\, f(\mathbf{y})\, dA(\mathbf{y}), \tag{5.5}$$

were Φ is the fundamental solution of the Laplace operator,

$$\Phi(\mathbf{x}) = -\frac{1}{2\pi} \log|\mathbf{x}|.$$

When f is compactly supported, the integral in (5.5) can be discretized using appropriate quadratures, and the resulting finite sum can be evaluated rapidly via, *e.g.*, the Fast Multipole Method (FMM). In the discrete case, it turns out to be possible to define a *lattice fundamental function* ϕ such that the exact solution of (5.2) is

$$u(\mathbf{m}) = [\phi * f](\mathbf{m}) = \sum_{\mathbf{n}\in\mathbb{Z}^2} \phi(\mathbf{m}-\mathbf{n})\, f(\mathbf{n}). \tag{5.6}$$

The function ϕ cannot be expressed directly in terms of elementary functions, but can easily be evaluated numerically from its Fourier representation

$$\phi(\mathbf{m}) = \frac{1}{(2\,\pi)^2} \int_{[-\pi,\,\pi]^2} \frac{\cos(\mathbf{m}\cdot\mathbf{t}) - 1}{4\sin^2(t_1/2) + 4\sin^2(t_2/2)}\, dA(\mathbf{t}), \qquad \mathbf{m} \in \mathbb{Z}^2,$$

as presented in Section 4.2. In Section 5.1, we recall that when f is supported on a finite number N_{source} of source points, the sum (5.6) can be evaluated rapidly via a lattice version of the FMM.

Supposing that the potential is required only at the source points, the computational cost $T_{\text{freespace}}$ of the scheme satisfies

$$T_{\text{freespace}} \sim N_{\text{source}}. \tag{5.7}$$

The techniques for the free space problem on a perfectly periodic infinite lattice can easily be modified to handle local deviations from periodicity. Specifically, we describe in Section 5.2 techniques for solving the equation

$$[(\mathsf{A}+\mathsf{B})u](\mathbf{m}) = f(\mathbf{m}), \qquad \mathbf{m} \in \mathbb{Z}^2, \tag{5.8}$$

where B is a local operator acting on some finite subset $\Omega_{\text{inc}} \subset \mathbb{Z}^2$. An equation such as (5.8) can be used to model a lattice in which a finite number of bars have been added or removed, or have had their conductivities changed, see Figure 5.1. By convolving the equation (5.8) by the lattice fundamental solution ϕ, an equation defined on the finite subset Ω_{inc} is obtained. Moreover, this equation can using fast methods be solved in time T_{inc}, where T_{inc} scales linearly with the number of points N_{inc} in the inclusion,

$$T_{\text{inc}} \sim N_{\text{inc}} + N_{\text{source}}. \tag{5.9}$$

For boundary value problems such as (5.3), the techniques proposed in this manuscript are lattice analogs of Boundary Integral Equation (BIE) methods for solving elliptic partial differential equations. To illustrate, let us consider a Laplace boundary value problem with Dirichlet boundary conditions that is a continuum analog of (5.3):

$$\begin{cases} [-\Delta\, u](\mathbf{x}) = \ 0, & \mathbf{x} \in \Omega, \\ \qquad u(\mathbf{x}) = \ g(\mathbf{x}), & \mathbf{x} \in \Gamma. \end{cases} \tag{5.10}$$

It is possible to reduce (5.10) to an equation on Γ by first representing the solution u as a *double layer potential*

$$u(\mathbf{x}) = \int_{\Gamma} D(\mathbf{x},\mathbf{y})\, \sigma(\mathbf{y})\, d\ell(\mathbf{y}), \tag{5.11}$$

where D is the so called *double layer kernel*,

$$D(\mathbf{x},\mathbf{y}) = \frac{\partial}{\partial \mathbf{n}(\mathbf{y})}\Phi(\mathbf{x}-\mathbf{y}) = \mathbf{n}(\mathbf{y}) \cdot \nabla_{\mathbf{y}}\Phi(\mathbf{x}-\mathbf{y}) = \frac{\mathbf{n}(\mathbf{y}) \cdot (\mathbf{x}-\mathbf{y})}{2\,\pi\,|\mathbf{x}-\mathbf{y}|^2},$$

where $\mathbf{n}(\mathbf{y})$ is the unit normal vector of Γ at $\mathbf{y}$. The boundary density function σ in (5.10) is determined by calculating the limit of $u(\mathbf{x})$ as $\mathbf{x}$ approaches the boundary Γ and equating the result with the Dirichlet condition in (5.10). This results in the second kind Fredholm equation

$$\frac{1}{2}\sigma(\mathbf{x}) + \int_{\Gamma} D(\mathbf{x},\mathbf{y})\,\sigma(\mathbf{y})\,d\ell(\mathbf{y}) = g(\mathbf{x}), \qquad \mathbf{x} \in \Gamma. \tag{5.12}$$

We find that a solution to (5.10) can now be obtained by solving the equation (5.12). The reformulation offers several advantages, including a reduction in dimensionality, and a well conditioned equation [3]. An apparent disadvantage of using (5.12) instead of (5.10) as a foundation for numerical work is that (5.12) leads to dense systems upon discretization. This potential drawback can again be overcome by the FMM. The principal contribution of this chapter is to demonstrate that existing fast numerical methods developed for continuum problems can be modified to work for lattice equations and lead to highly efficient solvers. It has been demonstrated [59, 56] that the solution of (5.3) can be written

$$u(\mathbf{m}) = \sum_{n\in\Gamma} d(\mathbf{m},\mathbf{n})\,\sigma(\mathbf{n}), \tag{5.13}$$

where d is a discrete analog of the continuum double layer potential (5.12). (Equation (5.56) provides the precise definition.) An equation for σ is obtained by inserting (5.13) into the boundary condition in (5.3) which results in the boundary equation

$$\sum_{n\in\Gamma} d(\mathbf{m},\mathbf{n})\,\sigma(\mathbf{n}) = g(\mathbf{m}), \qquad \mathbf{m} \in \Gamma. \tag{5.14}$$

It was shown in [59] that (5.14) is very well conditioned (as the continuum analog would indicate) and we demonstrate in Section 5.3 that it can be solved in time

$$T_{\text{bvp}} \sim N_{\text{boundary}}, \tag{5.15}$$

where N_{boundary} denotes the number of points in Γ.

The techniques for handling (i) body loads, (ii) deviations from periodicity, and (iii) boundary conditions, can all be combined. We demonstrate in Section 5.4 that an equation of the form

$$\begin{cases} [(\mathsf{A}+\mathsf{B})u](\mathbf{m}) &= f(\mathbf{m}), \qquad \mathbf{m} \in \Omega, \\ u(\mathbf{m}) &= g(\mathbf{m}), \qquad \mathbf{m} \in \Gamma, \end{cases} \tag{5.16}$$

can be solved in time T_{combined} that satisfies

$$T_{\text{combined}} \sim N_{\text{source}} + N_{\text{inc}} + N_{\text{boundary}}. \tag{5.17}$$

The core point of this chapter is that the time requirements (5.7), (5.9), (5.15), and (5.17) are in a strong sense optimal: The computational time depends linearly on the amount of actual input data. In contrast, the conventional approach to solve, *e.g.*, the boundary value problem (5.3) would be to form and solve a linear system of size $N_{\text{domain}} \times N_{\text{domain}}$, where N_{domain} denotes the number of points in Ω. This system is sparse, and can often be solved in time proportional to the number of degrees of freedom (using, *e.g.*, multigrid). However, the time $T_{\text{conventional}}$ required even for such a linear complexity solver would satisfy

$$T_{\text{conventional}} \sim N_{\text{domain}}, \tag{5.18}$$

which is far worse that (5.15) since one would typically have

$$N_{\text{domain}} \sim N_{\text{boundary}}^2 \qquad \text{when } \Omega \subset \mathbb{Z}^2,$$
$$N_{\text{domain}} \sim N_{\text{boundary}}^{3/2} \qquad \text{when } \Omega \subset \mathbb{Z}^3.$$

In fairness, it must be noted that the constant of proportionality in the estimate (5.18) for conventional methods is typically lower than that in (5.15) for the methods proposed here. In particular, there exist very fast methods based on the FFT which exploit the fact that a constant coefficient difference equation on a regular grid is diagonal in Fourier space. The time T_{fft} required by such a method satisfies

$$T_{\text{fft}} \sim N_{\text{domain}} \log N_{\text{domain}},$$

but with a very small constant. A limitation of FFT based methods is that they intrinsically require the computational domain to be a rectangle with periodic boundary conditions. However, they are so fast that even for problems involving other boundary conditions, it often makes sense to either modify the mathematical model to conform to the available numerical tool, or to implement various "correction" techniques. In contrast, the methods proposed in this chapter have the advantage of being able to naturally handle domains of different shapes and with different boundary

conditions (Dirichlet, Neumann, decay at infinity, periodic or quasi-periodic, *etc*), in addition to their advantage of having complexity $O(N_{\text{boundary}})$ rather than $O(N_{\text{domain}})$.

Some indication of the problem size at which it becomes advantageous to switch to the methods proposed in this chapter is given by the numerical examples reported in Section 5.5. The switching point depends on the computational environment, but typically occurs for lattices with between 10^4 and 10^6 nodes.

5.1 Techniques for the free space problem

This section describes a fast numerical method for solving a lattice analog of a free space Poisson equation.

5.1.1 Problem definition

With A defined by (5.1), the free space lattice Poisson equation for a given source function f takes the form

$$[\mathsf{A}\, u](\mathbf{m}) = f(\mathbf{m}), \qquad \mathbf{m} \in \mathbb{Z}^2. \tag{5.19}$$

We assume for simplicity that f has compact support. In this case, the equation (5.19) has a unique solution that tends to zero at infinity whenever

$$\sum_{\mathbf{m}\in\mathbb{Z}^2} f(\mathbf{m}) = 0. \tag{5.20}$$

If (5.20) does not hold, then we require u to grow at most logarithmically at infinity,

$$\sup_{\mathbf{m}\in\mathbb{Z}^2} \frac{|u(\mathbf{m})|}{\log(2 + |\mathbf{m}|)} < \infty, \tag{5.21}$$

which ensures that (5.19) has a unique solution (up to a shift by a constant), see [56].

5.1.2 Continuum analog

A continuum analog of (5.19) is the Poisson equation

$$[-\Delta u](\mathbf{x}) = f(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^2,$$

whose analytic solution (under suitable decay conditions on f and u) is

$$u(\mathbf{x}) = [\Phi * f](\mathbf{x}) = \int_{\mathbb{R}^2} \Phi(\mathbf{x} - \mathbf{y})\, f(\mathbf{y})\, dA(\mathbf{y}),$$

where Φ is the fundamental solution of the Laplace operator,

$$\Phi(\mathbf{x}) = -\frac{1}{2\pi} \log |\mathbf{x}|.$$

5.1.3 The lattice fundamental solution

As mentioned in the introduction, it is possible to construct a fundamental solution ϕ to the discrete Laplacian such that a solution to (5.19) and (5.21) is provided by the convolution

$$u(\mathbf{m}) = [\phi * f](\mathbf{m}) = \sum_{\mathbf{n} \in \mathbb{Z}^2} \phi(\mathbf{m} - \mathbf{n})\, f(\mathbf{n}). \tag{5.22}$$

In this section, we define ϕ, sketch how to construct it, and describe its asymptotic expansion at infinity.

5.1.3.1 Definition

The lattice fundamental solution is defined as the unique function ϕ that satisfies the three conditions

$$\sup_{\mathbf{m} \in \mathbb{Z}^2} \frac{|\phi(\mathbf{m})|}{\log(2 + |\mathbf{m}|)} < \infty, \tag{5.23}$$

$$\phi(\mathbf{0}) = 0, \tag{5.24}$$

$$\mathsf{A}\,\phi = \delta, \tag{5.25}$$

where the function δ is defined by

$$\delta(\mathbf{m}) = \begin{cases} 1, & \mathbf{m} = 0, \\ 0, & \mathbf{m} \neq 0. \end{cases} \tag{5.26}$$

For future reference, we define the solution operator of (5.19) and (5.21) as the operator G defined via

$$[\mathsf{G}\, f](\mathbf{m}) = [\phi * f](\mathbf{m}) = \sum_{n \in \mathbb{Z}^2} \phi(\mathbf{m} - \mathbf{n})\, f(\mathbf{n}). \tag{5.27}$$

Then the solution to (5.19) and (5.21) is simply $u = \mathsf{G}\, f$.

5.1.3.2 Analytic formula

In order to construct an analytic formula for ϕ, we introduce the discrete Fourier transform F via

$$\hat{u}(\mathbf{t}) = [F\,u](\mathbf{t}) = \sum_{\mathbf{m}\in\mathbb{Z}^2} e^{i\,\mathbf{m}\cdot\mathbf{t}}\,u(\mathbf{m}), \qquad \mathbf{t}\in[-\pi,\,\pi]^2. \tag{5.28}$$

The inverse transform is given by

$$u(\mathbf{m}) = [F^*\,\hat{u}](\mathbf{m}) = \frac{1}{(2\pi)^2}\int_{[-\pi,\,\pi]^2} e^{-i\,\mathbf{m}\cdot\mathbf{t}}\,\hat{u}(\mathbf{t})\,dA(\mathbf{t}), \qquad \mathbf{m}\in\mathbb{Z}^2. \tag{5.29}$$

With (5.28) and (5.29), the discrete Laplace operator has the Fourier representation

$$[F\,\mathsf{A}\,F^*\,\hat{u}](\mathbf{t}) = 4\,\hat{u}(\mathbf{t}) - e^{i\,t_1}\,\hat{u}(\mathbf{t}) - e^{-i\,t_1}\,\hat{u}(\mathbf{t}) - e^{i\,t_2}\,\hat{u}(\mathbf{t}) - e^{-i\,t_2}\,\hat{u}(\mathbf{t}) = \sigma(\mathbf{t})\,\hat{u}(\mathbf{t}),$$

where the *symbol* σ of A is given by

$$\sigma(\mathbf{t}) = 4 - e^{i\,t_1} - e^{-i\,t_1} - e^{i\,t_2} - e^{-i\,t_2} = 4\,\sin^2\frac{t_1}{2} + 4\,\sin^2\frac{t_2}{2}.$$

Applying F to both sides of (5.25), we get the equation

$$\sigma(\mathbf{t})\,\hat{\phi}(\mathbf{t}) = 1.$$

It seems that ϕ should now be obtained by simply solving for $\hat{\phi}$ and applying the inverse Fourier transform. This would lead to the formula

$$\phi(\mathbf{m}) = \frac{1}{(2\,\pi)^2}\int_{[-\pi,\,\pi]^2} e^{-i\,\mathbf{m}\cdot\mathbf{t}}\,\frac{1}{\sigma(\mathbf{t})}\,dA(\mathbf{t}). \tag{5.30}$$

However, the integrand in (5.30) is strongly singular, and must be renormalized. The result is the formula

$$\phi(\mathbf{m}) = \frac{1}{(2\,\pi)^2}\int_{[-\pi,\,\pi]^2} \frac{e^{-i\,\mathbf{m}\cdot\mathbf{t}} - 1}{\sigma(\mathbf{t})}\,dA(\mathbf{t}). \tag{5.31}$$

See [56, 60] for details.

5.1.3.3 Asymptotic expansion

It has been established (see *e.g.* [56, 60, 27, 28, 55]) that as $|\mathbf{m}| \to \infty$, the fundamental solution ϕ defined by (5.31) has the asymptotic expansion

$$\phi(\mathbf{m}) = -\frac{1}{2\pi}\left(\log|\mathbf{m}| + \gamma + \frac{\log 8}{2}\right) + \frac{1}{24\pi}\frac{m_1^4 - 6m_1^2m_2^2 + m_2^4}{|\mathbf{m}|^6} + \frac{1}{480\pi}\frac{43m_1^8 - 772m_1^6m_2^2 + 1570m_1^4m_2^4 - 772m_1^2m_2^6 + 43m_2^8}{|\mathbf{m}|^{12}} + O(1/|\mathbf{m}|^6). \quad (5.32)$$

The number γ is the Euler constant ($\gamma = 0.577206\cdots$).

5.1.4 Fast evaluation of convolutions via the Fast Multipole Method

In this section we describe certain modifications to the classical Fast Multipole Method (FMM) [44, 45] that allow the rapid evaluation of the lattice potential due to a set of sources $\{f_i\}_{i=1}^{N_{\text{source}}}$, placed at points $\{\mathbf{n}_i\}_{i=1}^{N_{\text{source}}} \subset \mathbb{Z}^2$. Assuming that we seek the potential at the locations of the sources, we need to evaluated the sums

$$u_i = \sum_{j=1}^{N_{\text{source}}} f_j\,\phi(\mathbf{n}_i - \mathbf{n}_j), \qquad i = 1,\, 2,\, \ldots,\, N_{\text{source}}. \quad (5.33)$$

Our first step is to split the sum into a near-field and a far-field term. We say that two sources i and j are *near* if $|\mathbf{n}_i - \mathbf{n}_j|_\infty \le L$, where $|\cdot|_\infty$ denotes the ℓ^∞ norm on $\mathbb{Z}^2$, and where L is an adjustable parameter. The sum (5.33) then splits into two parts

$$u_i = u_i^{\text{near}} + u_i^{\text{far}},$$

where the *near-field* is defined by

$$u_i^{\text{near}} = \sum_{j\,:\,|\mathbf{n}_i - \mathbf{n}_j|_\infty \le L} f_j\,\phi(\mathbf{n}_i - \mathbf{n}_j), \qquad i = 1,\, 2,\, \ldots,\, N_{\text{source}}, \quad (5.34)$$

and the *far-field* is defined by

$$u_i^{\text{far}} = \sum_{j\,:\,|\mathbf{n}_i - \mathbf{n}_j|_\infty > L} f_j\,\phi(\mathbf{n}_i - \mathbf{n}_j), \qquad i = 1,\, 2,\, \ldots,\, N_{\text{source}}. \quad (5.35)$$

We first observe that for each lattice node there are only $4L^2 + 4L$ other nodes in its near-field. The values of the lattice fundamental solution for these possible interactions can be pre-computed by directly evaluating the integral (5.31) and storing the results (due to symmetries, only $(L + 1)(L + 2)/2$ values are actually needed). The near-field contribution can therefore be evaluated to floating-point precision using, at worst, $O(N\,L^2)$ operations.

For the far-field contribution, we approximate the sum (5.35) by replacing the kernel ϕ by an approximation ϕ_{far} obtained by omitting the $O(|\mathbf{m}|^{-6})$ term in (5.32). The introduced error is controlled by the parameter L; we found that by choosing $L = 30$, the relative error incurred was less than 10^{-12}. The resulting sum

$$u_i^{\text{far}} \approx \sum_{j\,:\,|\mathbf{n}_i - \mathbf{n}_j|_\infty > L} f_j\,\phi_{\text{far}}(\mathbf{n}_i - \mathbf{n}_j), \qquad i = 1,\,2,\,\ldots,\,N_{\text{source}}, \tag{5.36}$$

is then amenable to the either the classical FMM [44, 45], or more recent kernel-independent variations [40, 63, 78]. We chose to implement a two-dimensional version of the method of [63] since it was readily available. For a problem on $\mathbb{Z}^3$, we would expect the FFT-accelerated method of [78] to perform better.

5.2 Techniques for lattices with inclusions

This section describes techniques for solving a Poisson equation similar to (5.19) but with the twist that parts of the lattice may be perturbed from perfect periodicity. The mathematical model we consider can handle both the removal of links from the lattice, and the inclusion of additional ones; the only essential assumptions are that the perturbation be linear, and that only finitely many lattice nodes are affected.

5.2.1 Problem definition

We consider a perturbed lattice equation

$$[(\mathsf{A} + \mathsf{B})\,u](\mathbf{m}) = f(\mathbf{m}), \qquad \mathbf{m} \in \mathbb{Z}^2 \tag{5.37}$$

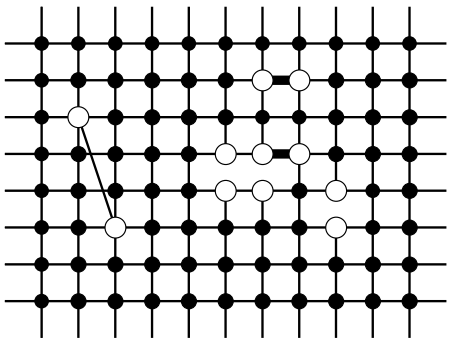


Figure 5.1: A piece of an infinite lattice with some deviations from perfect periodicity. One bar has been added, three bars have been removed, and two bars have been strengthened. The set Ω_{inc} of effected nodes has 11 elements, which are marked with white circles.

along with the decay condition (5.21) where B is a perturbation to the discrete Laplace operator arising from local deviations from perfect periodicity. Specifically, we assume that B is such that $\mathsf{A}+\mathsf{B}$ remains coercive (when coupled with the decay condition (5.21)), and also that B is "local" in the sense that there exists a finite set $\Omega_{\text{inc}} \subset \mathbb{Z}^2$ such that:

- $\mathsf{B}u = 0$ when u is such that $u(\mathbf{m}) = 0$ for all $\mathbf{m} \in \Omega_{\text{inc}}$.
- For any u, $[\mathsf{B}\,u](\mathbf{m}) = 0$ when $\mathbf{m} \notin \Omega_{\text{inc}}$.

The two conditions amount to an assumption that B is a block diagonal operator supported on the block corresponding to Ω_{inc}.

Example: Let us consider a lattice that is perturbed by adding J bars to the lattice. Letting r_j denote the conductivity of the j'th added bar, and letting $\mathbf{m}_j^+$ and $\mathbf{m}_j^-$ denote the nodes that the bar connects, the perturbation B takes the form

$$[\mathsf{B}\,u](\mathbf{m}) = \sum_{j\,:\,\mathbf{m}=\mathbf{m}_j^+} r_j\left(u(\mathbf{m}_j^+) - u(\mathbf{m}_j^-)\right) + \sum_{j\,:\,\mathbf{m}=\mathbf{m}_j^-} r_j\left(u(\mathbf{m}_j^-) - u(\mathbf{m}_j^+)\right). \tag{5.38}$$

In this case,

$$\Omega_{\text{inc}} = \bigcup_{j=1}^{J} \{\mathbf{m}_j^+,\, \mathbf{m}_j^-\}. \tag{5.39}$$

If all the numbers r_j are non-negative, then the operator B defined by (5.38) is non-negative as well, and (5.37) coupled with (5.21) has a unique solution for any f. The operator $\mathsf{A}+\mathsf{B}$ may be coercive

even when some of the numbers r_j are negative. For instance, if the pairs of nodes $\{\mathbf{m}_j^+, \mathbf{m}_j^-\}_{j=1}^J$ are connected in the original lattice, then by setting $r_j = -1$ for all j, the equations (5.37), (5.38), (5.21) model the lattice obtained by cutting the connections between the pairs $\{\mathbf{m}_j^+, \mathbf{m}_j^-\}$. Whether $\mathsf{A} + \mathsf{B}$ remains coercive now depends on the topology of the lattice after the cuts — essentially on whether all nodes remain connected.

5.2.2 Continuum analog

Equation (5.37) can be viewed as a discrete version of the perturbed free space Poisson equation

$$-\Delta u(\mathbf{x}) - \nabla \cdot \big(b(\mathbf{x})\, \nabla u(\mathbf{x})\big) = f(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^2, \tag{5.40}$$

where $b(\mathbf{x})$ is a function that is non-zero only in some bounded domain Ω_{inc}. Now convolving (5.40) by Φ, we obtain the new equation

$$u(\mathbf{x}) - \int_{\mathbb{R}^2} \Phi(\mathbf{x}-\mathbf{y})\, \nabla \cdot \big(b(\mathbf{y})\, \nabla u(\mathbf{y})\big)\, dA(\mathbf{y}) = [\Phi * f](\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^2. \tag{5.41}$$

Since $b(\mathbf{y}) = 0$ whenever $\mathbf{y} \notin \Omega_{\text{inc}}$, we can restrict the domain of integration in (5.41) to Ω_{inc}, and obtain an integral equation for u of the form

$$u(\mathbf{x}) - \int_{\Omega_{\text{inc}}} \Phi(\mathbf{x}-\mathbf{y})\, \nabla \cdot \big(b(\mathbf{y})\, \nabla u(\mathbf{y})\big)\, dA(\mathbf{y}) = [\Phi * f](\mathbf{x}), \qquad \mathbf{x} \in \Omega_{\text{inc}}. \tag{5.42}$$

Our gain is to have converted the equation (5.40), which involved an unbounded operator on an infinite domain, to (5.42), which involves a bounded operator on a finite domain. The integral operator in equation (5.42) is in general not compact, but well-posedness can often be assured via a simple positivity or perturbation argument.

5.2.3 A local lattice equation

To convert the equation (5.37) to an equation on the finite domain Ω_{inc}, we follow the template established by the continuum case in Section 5.2.2 and convolve (5.37) by the lattice fundamental solution ϕ. This yields the equation

$$\big[(\mathsf{I} + \mathsf{G}\,\mathsf{B})\, u\big](\mathbf{m}) = [\mathsf{G}\, f](\mathbf{m}), \qquad \mathbf{m} \in \Omega_{\text{inc}}. \tag{5.43}$$

It is in many environments convenient to post-multiply (5.43) by B again, which results in the equation

$$\big(\mathsf{I} + \mathsf{B}\,\mathsf{G}\big)\,\mu = \mathsf{B}\,\mathsf{G}\,f, \tag{5.44}$$

where the new unknown variable is $\mu = \mathsf{B}\,u$. Once (5.44) has been solved for μ, the full solution u is given by the formula

$$u = \mathsf{G}\,\big(f - \mu\big). \tag{5.45}$$

5.2.4 Numerical methods

We have found that the equation (5.44) can easily be solved using an iterative solver such as GMRES. For large scale problems, application of the operator G can be accelerated using the lattice FMM described in Section 5.1.4. The lattice FMM can also be used to rapidly evaluate the potential via (5.45) once μ has been determined from (5.44).

5.3 Techniques for lattice boundary value problems

In this section we describe techniques for solving the lattice equilibrium equations on a finite lattice with prescribed boundary conditions. The techniques can be modified to a wide range of different boundary conditions (see [59]) but for concreteness, we restrict attention to the basic Dirichlet and Neumann boundary conditions.

5.3.1 Problem definition

5.3.1.1 Definition of a lattice domain and boundary

Let $\bar{\Omega}$ denote a finite subset of $\mathbb{Z}^2$. We define the *interior* of $\bar{\Omega}$ as the set Ω of nodes whose four neighbors are all contained in $\bar{\Omega}$, and the *boundary* Γ as the remaining nodes, $\Gamma = \bar{\Omega}\backslash\Omega$. Figure 5.2(a) illustrates these definitions. For simplicity, we assume that Ω forms a connected lattice.

In addition to defining the boundary of the domain, we also need to define exterior and interior boundary flux operators, analogous to normal derivatives in the continuum case. To this

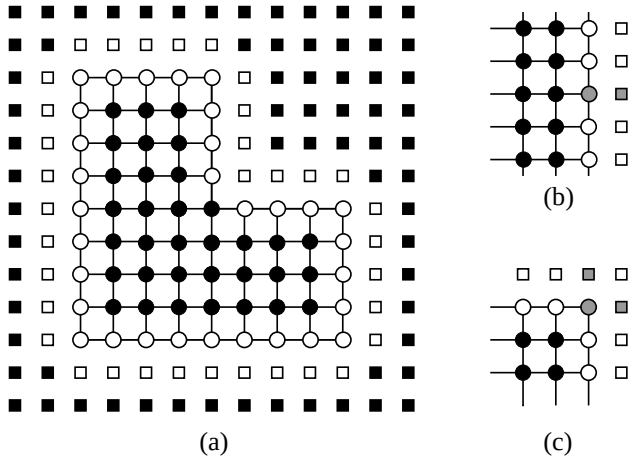


Figure 5.2: (a) An example of a lattice domain, $\bar{\Omega} = \Gamma \cup \Omega$. The black circles form the interior Ω and the white circles form the boundary Γ. (b) Illustration of the set $\mathbb{D}_m$ (the grey square) for a boundary node m (grey circle) along a straight edge. (c) Illustration of $\mathbb{D}_m$ for a corner node.

end, we let for a given boundary node $\mathbf{n} \in \Gamma$ the set $\mathbb{D}_{\mathbf{n}}$ denote the set of all points in $\mathbb{Z}^2$ that connect to $\mathbf{n}$, but are not contained in $\bar{\Omega}$. We let $\mathbb{E}_{\mathbf{n}}$ denote the remaining nodes connecting to $\mathbf{n}$ so that $\mathbb{D}_{\mathbf{n}} \cup \mathbb{E}_{\mathbf{n}}$ forms a disjoint union of the four nodes connecting to $\mathbf{n}$, see Figure 5.2(b,c). We can now define an *exterior difference operator* ∂ via

$$[\partial\, u](\mathbf{n}) = \sum_{\mathbf{k}\in\mathbb{D}_{\mathbf{n}}} \big(u(\mathbf{k}) - u(\mathbf{n})\big), \tag{5.46}$$

and an *interior difference operator* $\bar{\partial}$ via

$$[\bar{\partial}\, u](\mathbf{n}) = \sum_{\mathbf{k}\in\mathbb{E}_{\mathbf{n}}} \big(u(\mathbf{n}) - u(\mathbf{k})\big). \tag{5.47}$$

5.3.1.2 The Dirichlet problem

A Boundary Value Problem (BVP) with Dirichlet boundary conditions takes the form

$$\begin{cases} [\mathsf{A}\, u](\mathbf{m}) = 0, & \mathbf{m} \in \Omega, \\ \quad u(\mathbf{m}) = g(\mathbf{m}), & \mathbf{m} \in \Gamma. \end{cases} \tag{5.48}$$

It can be demonstrated that (5.48) has a unique solution for any boundary load g, see *e.g.* [56].

5.3.1.3 The Neumann problem

This problem corresponds physically to prescribing the boundary fluxes, rather than the temperatures. Mathematically, we specify the values of the interior difference operator, which results in the equation

$$\begin{cases} [\mathsf{A}\, u](\mathbf{m}) = 0, & \mathbf{m} \in \Omega, \\ [\bar{\partial}\, u](\mathbf{m}) = g(\mathbf{m}), & \mathbf{m} \in \Gamma. \end{cases} \tag{5.49}$$

When

$$\sum_{\mathbf{m}\in\Gamma} g(\mathbf{m}) = 0, \tag{5.50}$$

equation (5.49) has a solution that is unique up to a constant.

5.3.2 The continuum analog

The continuum analogs of (5.48) and (5.49) are of course

$$\begin{cases} [-\Delta u](\mathbf{x}) = & 0, & \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = & g(\mathbf{x}), & \mathbf{x} \in \Gamma, \end{cases} \tag{5.51}$$

and

$$\begin{cases} [-\Delta u](\mathbf{x}) = & 0, & \mathbf{x} \in \Omega, \\ \frac{\partial u(\mathbf{x})}{\partial n} = & g(\mathbf{x}), & \mathbf{x} \in \Gamma. \end{cases} \tag{5.52}$$

It is well known from classical potential theory that the solution to (5.51) admits the representation

$$u(\mathbf{x}) = \int_\Gamma D(\mathbf{x}, \mathbf{y})\, \sigma(\mathbf{y})\, d\ell(\mathbf{y}),$$

where D is the so called *double layer kernel*

$$D(\mathbf{x}, \mathbf{y}) = \frac{\partial}{\partial \mathbf{n}(\mathbf{y})} \Phi(\mathbf{x} - \mathbf{y}) = \mathbf{n}(\mathbf{y}) \cdot \nabla_{\mathbf{y}} \Phi(\mathbf{x} - \mathbf{y}) = \frac{\mathbf{n}(\mathbf{y}) \cdot (\mathbf{x} - \mathbf{y})}{2\,\pi\,|\mathbf{x} - \mathbf{y}|^2}, \tag{5.53}$$

and σ is a boundary potential that can be determined by solving the second kind Fredholm equation

$$\frac{1}{2}\sigma(\mathbf{x}) + \int_\Gamma D(\mathbf{x}, \mathbf{y})\, \sigma(\mathbf{y})\, d\ell(\mathbf{y}) = g(\mathbf{x}), \qquad \mathbf{x} \in \Gamma.$$

Likewise, any solution to (5.52) admits a representation (up to addition of a constant)

$$u(\mathbf{x}) = \int_\Gamma S(\mathbf{x}, \mathbf{y})\, \sigma(\mathbf{y})\, d\ell(\mathbf{y}),$$

where S is the so called *single layer kernel*

$$S(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x} - \mathbf{y})$$

and σ is a boundary potential that can be determined by solving the second kind Fredholm equation

$$-\frac{1}{2}\sigma(\mathbf{x}) + \int_\Gamma D^*(\mathbf{x}, \mathbf{y})\, \sigma(\mathbf{y})\, d\ell(\mathbf{y}) = g(\mathbf{x}), \qquad \mathbf{x} \in \Gamma \tag{5.54}$$

where $D^*(\mathbf{x}, \mathbf{y})$ is the dual kernel of (5.53),

$$D^*(\mathbf{x}, \mathbf{y}) = \frac{\partial}{\partial \mathbf{n}(\mathbf{x})} \Phi(\mathbf{x} - \mathbf{y}) = \mathbf{n}(\mathbf{x}) \cdot \nabla_{\mathbf{x}} \Phi(\mathbf{x} - \mathbf{y}) = -\frac{\mathbf{n}(\mathbf{x}) \cdot (\mathbf{x} - \mathbf{y})}{2\,\pi\,|\mathbf{x} - \mathbf{y}|^2}. \tag{5.55}$$

We observe that the operator on the left hand side of (5.54) has a one-dimensional null-space, and that its range has co-dimension one (corresponding to the fact that g must integrate to zero). This causes very little difficulty in the construction of numerical methods based on (5.54) since the range is known analytically.

5.3.3 Lattice boundary equations

Inspired by the continuum case, a lattice analog of the double layer potential was proposed in [56, 59] (and was briefly mentioned in the introduction to the chapter in equation (5.13)). It is obtained by differentiating the fundamental solution ϕ using the external difference operator ∂ defined in (5.46),

$$d(\mathbf{m}, \mathbf{n}) = \partial_{\mathbf{n}} \phi(\mathbf{m} - \mathbf{n}) = \sum_{\mathbf{k} \in \mathbb{D}_{\mathbf{n}}} \big[\phi(\mathbf{m} - \mathbf{k}) - \phi(\mathbf{m} - \mathbf{n})\big]. \tag{5.56}$$

(The subscript $\mathbf{n}$ in $\partial_{\mathbf{n}}$ simply indicates that the difference operator is acting on the variable $\mathbf{n}$.) We define the corresponding operator D via

$$[\mathsf{D}\, q](\mathbf{m}) = \sum_{n \in \Gamma} d(\mathbf{m}, \mathbf{n})\, q(\mathbf{n}). \tag{5.57}$$

It can be shown that any solution u of (5.48) admits the representation

$$u = \mathsf{D}\, q \tag{5.58}$$

for some boundary charge distribution q. An equation for q is obtained by simply restricting (5.58) to Γ:

$$\sum_{n \in \Gamma} d(\mathbf{m}, \mathbf{n})\, q(\mathbf{n}) = g(\mathbf{m}), \qquad \mathbf{m} \in \Gamma. \tag{5.59}$$

Both theoretical and numerical results in [59] indicate that the equation (5.59) is typically a well conditioned equation, with the spectrum of D resembling that of a second kind Fredholm operator. (Remark 31 presents a calculation that perhaps makes this claim more intuitively plausible.)

Turning next to the Neumann equation (5.49), it can be shown that up to addition of a constant, any solution u admits a representation via a single layer potential

$$u(\mathbf{m}) = \sum_{n \in \Gamma} s(\mathbf{m}, \mathbf{n})\, q(\mathbf{n}), \tag{5.60}$$

where the *single layer potential* s is defined simply via

$$s(\mathbf{m}, \mathbf{n}) = \phi(\mathbf{m} - \mathbf{n}).$$

Inserting (5.60) into (5.49), we find that q must satisfy the equation

$$\sum_{n \in \Gamma} d^*(\mathbf{m}, \mathbf{n})\, q(\mathbf{n}) = g(\mathbf{m}), \qquad \mathbf{m} \in \Gamma, \tag{5.61}$$

where d^* is the kernel

$$d^*(\mathbf{m}, \mathbf{n}) = \bar{\partial}_{\mathbf{m}} s(\mathbf{m}, \mathbf{n}).$$

Again, both theoretical and experimental results indicate that the spectral properties of the system matrix in (5.61) resemble those of its continuum analog (5.54): precisely one singular value is zero, and the remaining ones are clustered relatively closely.

Remark 31. *The operators in equations (5.59) and (5.61) are in a certain sense complementary. To explicate this relationship, let us note that for a boundary node* $\mathbf{n}$ *we have, for any lattice function* u*,*

$$[\mathsf{A}\, u](\mathbf{n}) = \sum_{\mathbf{k} \in \mathbb{D}_{\mathbf{n}} \cup \mathbb{E}_{\mathbf{n}}} \big(u(\mathbf{n}) - u(\mathbf{k})\big) = [\bar{\partial}\, u](\mathbf{n}) - [\partial\, u](\mathbf{n}).$$

Consequently, the kernel of (5.59) satisfies

$$[\partial_{\mathbf{n}} \phi](\mathbf{m} - \mathbf{n}) = [(\bar{\partial}_{\mathbf{n}} - \mathsf{A}_{\mathbf{n}})\phi](\mathbf{m} - \mathbf{n}) = [\bar{\partial}_{\mathbf{n}}\, \phi](\mathbf{m} - \mathbf{n}) - \delta(\mathbf{m} - \mathbf{n}). \tag{5.62}$$

Inserting (5.62) into (5.59) we obtain the equation

$$g(\mathbf{m}) = -q(\mathbf{m}) + \sum_{\mathbf{n} \in \Gamma} \big(\bar{\partial}_{\mathbf{n}}\, \phi(\mathbf{m} - \mathbf{n})\big)\, q(\mathbf{n}), \qquad \mathbf{m} \in \Gamma. \tag{5.63}$$

Analogously, equation (5.61) is equivalent to the equation

$$g(\mathbf{m}) = q(\mathbf{m}) + \sum_{\mathbf{n} \in \Gamma} \big(\partial_{\mathbf{m}}\, \phi(\mathbf{m} - \mathbf{n})\big)\, q(\mathbf{n}), \qquad \mathbf{m} \in \Gamma. \tag{5.64}$$

A benefit of the formulations (5.63) and (5.64) is that they perhaps make it easier to intuit that (5.59) and (5.61) behave qualitatively like second kind Fredholm equations.

5.3.4 Numerical methods

The procedure for solving the Dirichlet problem (5.48) that was outlined in Section 5.3.3 consists of two steps: First (5.59) is solved for the boundary load q, and then the actual potential u is evaluated from q via the sum (5.58). The second step is executed numerically by simply applying the fast summation technique of Section 5.1.4. The linear solve in the first step can be solved using an iterative solver accelerated with the fast summation technique of Section 5.1.4. In numerical experiments, we found that the iterative solver (in our case GMRES) converged rapidly, as one would expect given that the coefficient matrix in (5.59) is extremely well conditioned [59]. However, one can do even better. It turns out that the system matrix in (5.59) is in fact a *Hierarchically Semi-Separable* matrix [18, 19, 20, 61], which means that not only can the matrix be applied to vectors in $O(N_{\text{boundary}})$ time, but it is possible to directly compute an approximation to its inverse in linear time, see Chapter 2. We implemented the scheme of [61] and found that a matrix of size $102,400 \times 102,400$ (corresponding to a lattice with about 6.5×10^8 nodes) could be inverted in 1 minute, and applied to a vector in 1 second. The computational time of course depends on the requested accuracy, and the numbers reported refer to a computation whose relative accuracy was 10^{-10}. See Section 5.5 for details.

The Neumann problem (5.49) can be solved numerically using procedures entirely analogous to those described for the Dirichlet problem. The Neumann problem involves the additional complication that the system matrix in (5.61) is singular. However, its range is known analytically (it is the set of functions that sum to zero), so the system can easily be modified to obtain a non-singular well-conditioned equation.

5.4 Combined techniques

5.4.1 Problem statement

In this section, we consider an equation on a bounded domain Ω for a problem that involves both a body load, and deviations from perfect periodicity. Letting B denote a non-negative operator

supported on some subset Ω_{inc} of Ω, such an equation takes the form

$$\begin{cases} [(\mathsf{A}+\mathsf{B})\,u](\mathbf{m}) = f(\mathbf{m}), & \mathbf{m} \in \Omega, \\ u(\mathbf{m}) = g(\mathbf{m}), & \mathbf{m} \in \Gamma, \end{cases} \tag{5.65}$$

where f is a given body load, and g is a given Dirichlet boundary condition. We will convert equation (5.65) to an equation defined on the smaller set $\Gamma \cup \Omega_{\text{inc}}$. The technique is exact and works for any non-negative operator B. However, it is particularly well suited to the case where Ω_{inc} is in some sense a "small" subset of Ω, and f is either zero, or also supported on a "small" subset.

5.4.2 Reformulation of the difference equation

We look for a solution of the form

$$u = \mathsf{G}\,f - \mathsf{G}\,\mu + \mathsf{D}\,q, \tag{5.66}$$

where D is the double layer operator defined in (5.57), where q is a function on Γ that is to be determined, and μ is the (as yet unknown) function

$$\mu = \mathsf{B}\,u. \tag{5.67}$$

It is immediately clear that if (5.66) and (5.67) are satisfied, then for $\mathbf{m} \in \Omega$,

$$[\mathsf{A}u](\mathbf{m}) = [\mathsf{A}\,\mathsf{G}\,f](\mathbf{m}) - [\mathsf{A}\,\mathsf{G}\,\mu](\mathbf{m}) + [\mathsf{A}\,\mathsf{D}\,q](\mathbf{m}) = f(\mathbf{m}) - \mu(\mathbf{m}) + 0 = f(\mathbf{m}) - [\mathsf{B}\,u](\mathbf{m}),$$

so the difference equation in (5.65) is satisfied. To enforce the boundary condition, we require μ and q to satisfy

$$g(\mathbf{m}) = [\mathsf{G}\,f](\mathbf{m}) - [\mathsf{G}\,\mu](\mathbf{m}) + [\mathsf{D}\,q](\mathbf{m}), \qquad \mathbf{m} \in \Gamma. \tag{5.68}$$

It remains to convert (5.66) to an equation that does not involve u. This is easily done by applying B to (5.66) and restricting the resulting equation to Ω_{inc},

$$\mu(\mathbf{m}) = [\mathsf{B}\,\mathsf{G}\,f](\mathbf{m}) - [\mathsf{B}\,\mathsf{G}\,\mu](\mathbf{m}) + [\mathsf{B}\,\mathsf{D}\,q](\mathbf{m}), \qquad \mathbf{m} \in \Omega_{\text{inc}}. \tag{5.69}$$

Combining (5.68) and (5.69), we obtain the system

$$\begin{bmatrix} \mathsf{D}_\Gamma & -\mathsf{G}_{(\Gamma\leftarrow\Omega_{\text{inc}})} \\ -\mathsf{B}_{\Omega_{\text{inc}}}\mathsf{D}_{(\Omega_{\text{inc}}\leftarrow\Gamma)} & (\mathsf{I}+\mathsf{B}_{\Omega_{\text{inc}}}\mathsf{G}_{\Omega_{\text{inc}}}) \end{bmatrix} \begin{bmatrix} q \\ \mu \end{bmatrix} = \begin{bmatrix} g-\mathsf{G}_{(\Gamma\leftarrow\Omega)}f \\ \mathsf{B}_{\Omega_{\text{inc}}}\mathsf{G}_{(\Omega_{\text{inc}}\leftarrow\Omega)}f \end{bmatrix}. \tag{5.70}$$

In (5.70) we added subscripts to the operators to indicate their range and domain. For instance, D_Γ is an operator mapping a function on Γ to a function on Γ while $\mathsf{D}_{(\Omega_{\text{inc}}\leftarrow\Gamma)}$ is an operator mapping a function on Γ to one defined on Ω_{inc}.

5.4.3 Numerical methods

All matrices in the linear system (5.70) are amenable to fast schemes for applying a matrix to a vector, and in the examples we have investigated, GMRES converges to a solution reasonably fast. Empirically, we found that the efficiency of the method can be improved if we exploit the fact that an approximation to D_Γ^{-1} can be computed (see Section 5.3.4) and directly eliminate the vector q from the system. This results in the equation

$$\tilde{\mathsf{A}}\mu = h \tag{5.71}$$

where

$$\tilde{\mathsf{A}} = \left[-\mathsf{B}_{\Omega_{\text{inc}}}\mathsf{D}_{(\Omega_{\text{inc}}\leftarrow\Gamma)}\mathsf{D}_\Gamma^{-1}\mathsf{G}_{(\Gamma\leftarrow\Omega_{\text{inc}})} + (\mathsf{I}+\mathsf{B}_{\Omega_{\text{inc}}}\mathsf{G}_{\Omega_{\text{inc}}})\right]$$

and

$$h = \mathsf{B}_{\Omega_{\text{inc}}}\mathsf{G}_{(\Omega_{\text{inc}}\leftarrow\Omega)}f + \mathsf{B}_{\Omega_{\text{inc}}}\mathsf{D}_{(\Omega_{\text{inc}}\leftarrow\Gamma)}\mathsf{D}_\Gamma^{-1}(g-\mathsf{G}_{(\Gamma\leftarrow\Omega)}f).$$

Once (5.71) has been solved, the vector q is retrieved via the formula

$$q = \mathsf{D}_\Gamma^{-1}(g-\mathsf{G}_{(\Gamma\leftarrow\Omega)}f + \mathsf{G}_{(\Gamma\leftarrow\Omega_{\text{inc}})}\mu),$$

and then u can be obtained from (5.66).

5.5 Numerical results

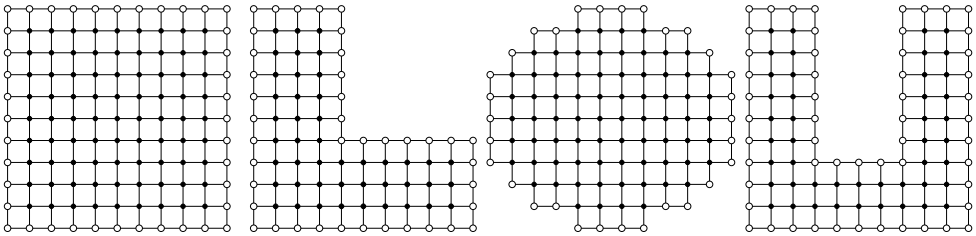
In this section, we illustrate the robustness of the proposed methodology.

All experiments are run on a Dell desktop computer with 2GB of RAM and an Intel Pentium 4 3.4GHz dual processor. The methods were run at a requested relative precision of 10^{-10}. The techniques were implemented rather crudely in Matlab, which means that significant further gains in speed should be achievable.

Numerical results for the lattice Fast Multiple method are reported in Chapter 4.

5.5.1 Numerical results for finite lattices

In this section, we investigate the techniques described in Section 5.3.4, as applied to solving the lattice Dirichlet problem (5.48) on the four domains illustrated in Figure 5.3. The *ellipses* have an aspect ratio of 0.75, and the *U-shapes* are scaled so that their thickness is one quarter the length of the long side.



(a) Square (b) L-shape (c) Ellipse (d) U-shape

Figure 5.3: Finite lattice geometries

For each domain, we recast the BVP (5.48) as a boundary equation of the form (5.59), and then solved (5.59) using the direct solvers described in Section 5.3.4. A direct solver of this kind involves three steps: (1) Create a compressed representation of the operator in a "data-sparse" format. (2) Compute an approximation to the inverse of the compressed operator. (3) Apply the inverse to the source vector. The times required for each of these three steps are shown in Figure 5.4. We make three observations:

- All steps in the computation scale linearly with the number of points N_{boundary} on the boundary of the domain.

- The constant of proportionality is small. Specifically, a lattice with $N_{\text{domain}} \approx 6.5 \times 10^8$

nodes can be processed in one minute. Once the inverse of the boundary operator has been computed, additional solves take only one second.

- Among the three steps of the direct solver, the compression takes by far the longest, and we expect that much could be gained by improving on the crude method we implemented.

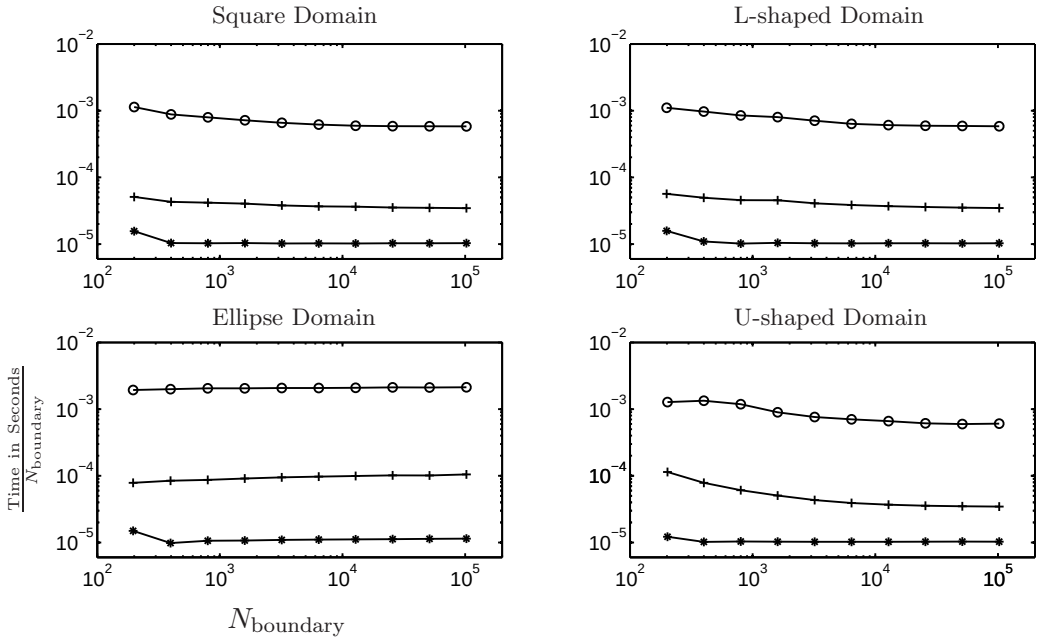
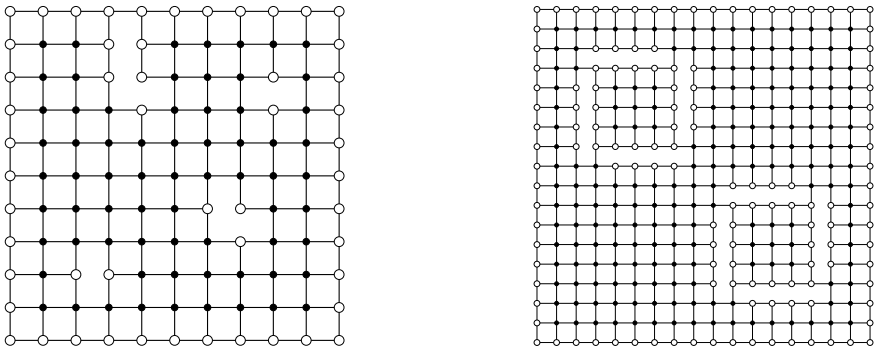


Figure 5.4: Times for inversion of the double layer operator. Labels: -o- Compression, -+- Inversion, -*- Application

5.5.2 Numerical results for finite lattices with inclusions

The experiments reported in this section are included to illustrate the conditioning of the linear systems (5.70) and (5.71) that model a finite lattice with local deviations from perfect periodicity. As an examples of such lattices, we consider a sequence of square 79×79 lattices in which p percent of the internal nodes (chosen at random) had been cut, see Figure 5.5(a). As $p \to 0$, the condition numbers of (5.70) and (5.71) approach the condition number for the unperturbed boundary equation (5.59), which is excellent (typically less than 10, see [59]). The interesting question is what happens as the lattice gets pushed further away from perfect periodicity.



(a) (b)

Figure 5.5: Lattice domains with inclusions

Figure 5.6(a) shows the condition numbers of (5.70) and (5.71) as functions of p. Figure 5.6(b) shows the closely related graphs illustrating of the number of iterations in GMRES required to attain a residual of less than 10^{-10}, again as functions of p. We first observe that the conditioning remains entirely acceptable for all values of p that we tested. However, between the two formulations, the Schur complement provides significantly better performance.

As a final example, we considered a square with 79×79 nodes, in which two smaller squares were partially separated from the main body of the lattice, as shown in Figure 5.5(b). The separation was accomplished by cutting p percent of the links (chosen at random) on each side of the small squares. We ran experiments all the way up to $p = 100$, at which point the interior of each square is connected to the rest of the lattice by only one link at each corner. The potential was grounded at the boundary (*i.e.* a homogeneous Dirichlet boundary condition was enforced), and two point sources were placed at the center of each small square. The solution technique described in Section 5.4.3 handled every value of p with ease. The potential fields associated with some values of p are shown in Figure 5.7. We remark that for large values of p, the physics of the underlying problems is quite ill-conditioned, and that the problems considered would be hard for previously existing methods.

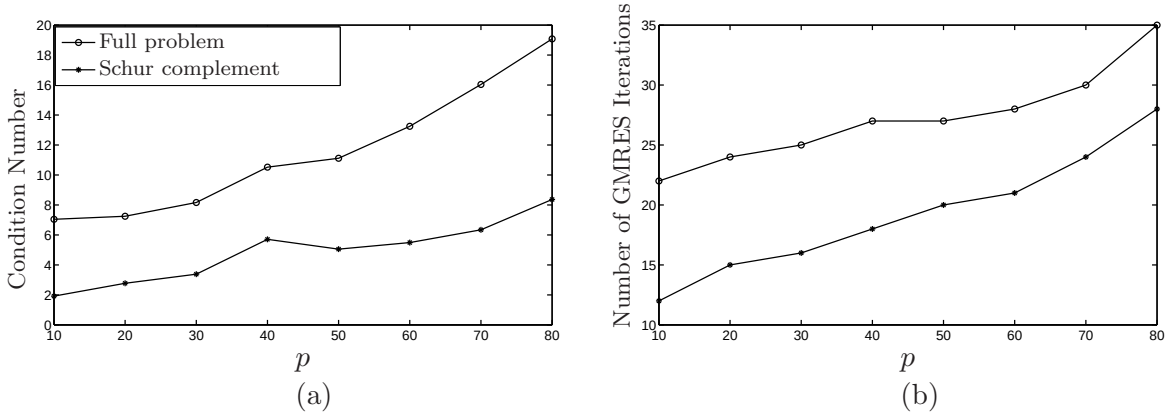


Figure 5.6: Finite lattice with random inclusions increasing from 20% to 80% of the domain. Square size is 79×79 Labels: -o- Full problem, -+- Schur complement problem

5.6 Generalization to other lattice operators

While this manuscript has focused exclusively on lattice equations involving the discrete Laplace operator (5.1) acting on the square lattice $\mathbb{Z}^2$ (or subsets thereof), the methods can straightforwardly be generalized in several ways.

The extension to other lattice operators on $\mathbb{Z}^2$ is particularly simple. The methods for the free-space problem, and for lattices with inclusions, apply directly once a fundamental solution for the operator under consideration has been constructed. Techniques for constructing such fundamental solutions are described in [60]. The techniques for handling boundary conditions in Section 5.3 also generalize, with the only caveat that for difference operators that involve more than the eight nearest neighbors of any lattice node, the boundary of a lattice domain must be extended to a boundary *layer* of nodes, sufficiently wide that the nodes inside the layer do not communicate directly with the nodes on the outside.

The extension to operators on more general mono-atomic or multi-atomic periodic lattices in two dimensions (such as triangular and hexagonal) is also relatively straightforward [56].

The extension to lattices in three dimensions is in principle not hard either, although in this case iterative methods must be used to solve the boundary equations that generalize (5.59), since we do not currently have methods for computing the inverse of such a boundary operator in linear

time. Moreover, the FMM that we used would probably not perform well in three dimensions, and we would recommend the use of the method of [78].

5.7 Conclusions

The chapter describes efficient techniques for solving elliptic difference equations on lattice domains. For simplicity of presentation, the paper focuses on lattice equations involving the discrete Laplace operator (5.1) acting on the square lattice $\mathbb{Z}^2$, or subsets thereof. Discrete analogs to boundary integral equations are proposed. These equations are amenable to fast solvers such as the Fast Multipole Method. Techniques are introduced for problems involving inclusions or local deviations from perfect periodicity. The complexity of the proposed method is $O(N_{\text{boundary}} + N_{\text{source}} + N_{\text{inc}})$ where N_{boundary} is the number of nodes on the boundary of the domain, N_{source} is the number of nodes subjected to body loads, and N_{inc} is the number of nodes that deviate from perfect periodicity.

Numerical experiments that demonstrate the robustness, versatility, and speed of the methods were presented. For instance, it was demonstrated that using lattice equivalents of boundary integral equations along with fast methods for dense matrices, it is possible to solve a boundary value problem on a lattice with $6.5 \cdot 10^9$ nodes (for which $N_{\text{boundary}} = 25,\, 600$) in 1 minute for the first solve; again using a standard desktop PC. The solution was accurate to ten digits. Once the first problem has been solved, additional right hand sides (that is, problems specifying other values on the boundary) can be handled in 1 second.

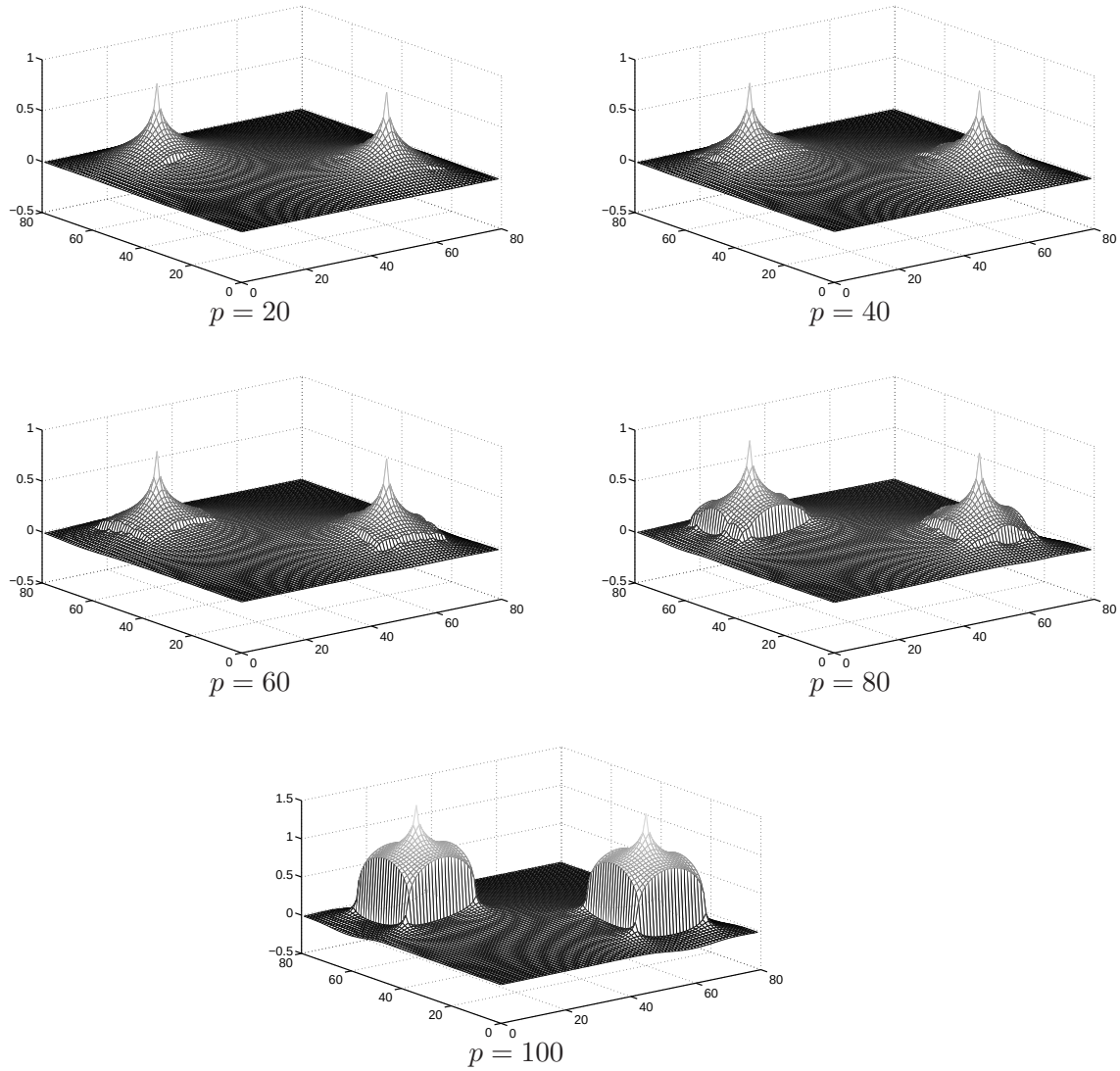


Figure 5.7: Potential for finite lattice with 2 random square inclusions and homogeneous boundary conditions. Lattice domain size is 79×79.

Chapter 6

A high-order discretization scheme for elliptic partial differential equations

This chapter considers problems of the form

$$\begin{cases} -\Delta\phi(\mathbf{x}) + b(\mathbf{x})\phi(\mathbf{x}) = 0, & \mathbf{x} \in \Omega, \\ \frac{\partial\phi(\mathbf{x})}{\partial\nu} = g(\mathbf{x}), & \mathbf{x} \in \Gamma, \end{cases} \tag{6.1}$$

where $\Omega \subset \mathbb{R}^2$ and $\Gamma = \partial\Omega$, $b(\mathbf{x}) \in C^\infty(\mathbb{R}^2)$, and $g(\mathbf{x})$ is some given boundary data. A classic solution approach is to discretize the PDE with a finite element or spectral element method. These methods lead to large sparse systems that are typically solved via iterative techniques such as GMRES or multigrid aided by a problem specific preconditioner for rapid convergence.

The method we propose builds the solution operator known as the Neumann-to-Dirichlet operator in a hierarchical fashion. While we believe the discretization is valid for arbitrary domains, we take Ω to be a square domain in this preliminary work. The main idea is to compute the Neumann-to-Dirichlet operator via a least squares solve for a large number of small square domains whose union is Ω. Having computed these to high accuracy, the global solution operator can be found by a sequence of merge procedures which involve the inversion of small dense matrices. The overall cost is $O(N^{1.5})$ where N is the total number of points on the boundary of the small boxes. For many problems, the Neumann-to-Dirichlet is an HSS matrix. Matrices of this form are well suited for fast matrix algebra including fast inversion. Utilizing these techniques reduces the computational cost to linear.

Section 6.1 begins the chapter by describing how the domain is partitioned. We choose the discretization points to be the one dimensional Gaussian quadrature points along the boundary of

each subdomain. Next, we formally define the Neumann-to-Dirichlet operator and, in Section 6.2.2, explain how to construct the operator cheaply in each of the small squares. By using the fluxes through the boundaries as unknowns, in Section 6.2.3, we combine local equations to form a global system. By using a nested dissection approach, we hierarchically eliminate the interior unknowns, in Section 6.3. In Section 6.4, we illustrate the potential of the method on several preliminary examples.

6.1 Discretization

First, Ω is tessellated into a large collection of small boxes called *leaf boxes*. Let I_{leaf} denote the set of all leaf boxes and let $\{\Gamma_i\}_{i \in I_{\text{leaf}}}$ denote the collection of the the edges of all the leaf boxes, see Figure 6.3. For each box i, we define u^i as the restriction of ϕ to Γ_i, ie.

$$u^i(\mathbf{x}) = \phi(\mathbf{x}) \text{ for } x \in \Gamma_i.$$

Further, we define $v^{(i)}$ as the restriction of the normal derivative across $\Gamma^{(i)}$:

$$v^{(i)}(x) = \begin{cases} [\partial_2 \phi](x) & \text{for } x \in \Gamma^{(i)} \text{ when } \Gamma^{(i)} \text{ is horizontal,} \\ [\partial_1 \phi](x) & \text{for } x \in \Gamma^{(i)} \text{ when } \Gamma^{(i)} \text{ is vertical,} \end{cases}$$

where $\partial_i = \partial / \partial x_i$.

On each line $\Gamma^{(i)}$, we place N_{gauss} Gaussian quadrature nodes. These points are collected in vectors $\gamma^{(i)} \in \mathbb{R}^{4\,N_{\text{gauss}} \times 2}$. By collocating the boundary functions $u^{(i)}$ and $v^{(i)}$ at the Gaussian nodes, we form the vectors $\mathbf{u}^{(i)}, \mathbf{v}^{(i)} \in \mathbb{R}^{N_{\text{gauss}}}$:

$$\mathbf{u}^{(i)} = u^{(i)}(\gamma^{(i)}),$$
$$\mathbf{v}^{(i)} = v^{(i)}(\gamma^{(i)}).$$

Since the functions $u^{(i)}(\mathbf{x})$ and $v^{(i)}(\mathbf{x})$ are smooth, the values of the functions between the Gaussian nodes can be approximated to very high accuracy by interpolation.

6.2 The equilibrium equations

Since the solution to (6.1) is unique, there exist an operator that maps the boundary information to the solution. This is called the Neumann-to-Dirichlet operator. In this section, we formally define this operator and its discrete analog. Techniques for constructing the discrete operator are presented in Section 6.2.2. By taking note of the relationship of the solution and fluxes between neighboring boxes, the Neumann-to-Dirichlet operators can be "merged" to form a global linear system (see Section 6.2.3).

6.2.1 Definition of the Neumann-to-Dirichlet operator

Let $\Omega^{(i)}$ be a subdomain of Ω with edges $\Gamma^{(i_1)}$, $\Gamma^{(i_2)}$, $\Gamma^{(i_3)}$, $\Gamma^{(i_4)}$, as shown in Figure 6.4. We define the boundary potentials and boundary fluxes for $\Omega^{(i)}$ via

$$u^{(i)} = \begin{bmatrix} u^{(i_1)} \\ u^{(i_2)} \\ u^{(i_3)} \\ u^{(i_4)} \end{bmatrix} \qquad \text{and} \qquad v^{(i)} = \begin{bmatrix} v^{(i_1)} \\ v^{(i_2)} \\ v^{(i_3)} \\ v^{(i_4)} \end{bmatrix}.$$

Since equation (6.1) has a unique solution, there exist a unique operator $T^{(i)}$ such that

$$u^{(i)} = T^{(i)}\, v^{(i)}, \tag{6.2}$$

where $u^{(i)}$ and $v^{(i)}$ are derived from any solution ϕ of (6.1). The operator $T^{(i)}$ is mathematically an integral operator called the *Neumann-to-Dirichlet* operator.

The discrete analog of the equation (6.2) is

$$\mathbf{u}^{(i)} = \mathsf{T}^{(i)}\, \mathbf{v}^{(i)}. \tag{6.3}$$

For the proposed method, it is sufficient for the matrix $\mathsf{T}^{(i)}$ to correctly construct $\mathbf{u}^{(i)}$ for any $\mathbf{v}^{(i)}$ that is the restriction of a function in the solution set under consideration.

Written out in components, $\mathsf{T}^{(i)}$ is a 4×4 block matrix that satisfies

$$\begin{bmatrix} \mathbf{u}^{(i_1)} \\ \mathbf{u}^{(i_2)} \\ \mathbf{u}^{(i_3)} \\ \mathbf{u}^{(i_4)} \end{bmatrix} = \begin{bmatrix} \mathsf{T}^{(i,11)} & \mathsf{T}^{(i,12)} & \mathsf{T}^{(i,13)} & \mathsf{T}^{(i,14)} \\ \mathsf{T}^{(i,21)} & \mathsf{T}^{(i,22)} & \mathsf{T}^{(i,23)} & \mathsf{T}^{(i,24)} \\ \mathsf{T}^{(i,31)} & \mathsf{T}^{(i,32)} & \mathsf{T}^{(i,33)} & \mathsf{T}^{(i,34)} \\ \mathsf{T}^{(i,41)} & \mathsf{T}^{(i,42)} & \mathsf{T}^{(i,43)} & \mathsf{T}^{(i,44)} \end{bmatrix} \begin{bmatrix} \mathbf{v}^{(i_1)} \\ \mathbf{v}^{(i_2)} \\ \mathbf{v}^{(i_3)} \\ \mathbf{v}^{(i_4)} \end{bmatrix}. \tag{6.4}$$

6.2.2 Construction of the Neumann-to-Dirichlet operator on a small box

In this section, we present techniques for constructing the Neumann-to-Dirichlet operator on a small box.

Let $\Omega^{(i)}$ be a small box with edges $\Gamma^{(i_1)}$, $\Gamma^{(i_2)}$, $\Gamma^{(i_3)}$, $\Gamma^{(i_4)}$, as shown in Figure 6.4, and consider the task of constructing a matrix $\mathsf{T}^{(i)}$ such that (6.3) holds for all permissible potentials.

Suppose there exist a collection of solutions $\{\phi_j\}_{j=1}^{N_{\text{samp}}}$ that locally span the solution space to the desired precision. For each ϕ_j, we construct the corresponding vectors of boundary values

$$\mathbf{u}_j = \begin{bmatrix} \phi_j(\gamma^{(i_1)}) \\ \phi_j(\gamma^{(i_2)}) \\ \phi_j(\gamma^{(i_3)}) \\ \phi_j(\gamma^{(i_4)}) \end{bmatrix}, \qquad \text{and} \qquad \mathbf{v}_j = \begin{bmatrix} \partial_2\phi_j(\gamma^{(i_1)}) \\ \partial_1\phi_j(\gamma^{(i_2)}) \\ \partial_2\phi_j(\gamma^{(i_3)}) \\ \partial_1\phi_j(\gamma^{(i_4)}) \end{bmatrix}.$$

Via a least squares procedure, a matrix $\mathsf{T}^{(i)}$ is constructed such that the equation

$$[\mathbf{u}_1 \ \mathbf{u}_2 \ \cdots \ \mathbf{u}_{N_{\text{samp}}}] = \mathsf{T}^{(i)} \, [\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \ \mathbf{v}_{N_{\text{samp}}}] \tag{6.5}$$

holds to within the specified tolerance ε.

The sample functions ϕ_j chosen such that each one is a solution to a local problem on a patch Ψ that covers the domain $\Omega^{(i)}$, as shown in Figure 6.7. The local problem reads

$$\begin{cases} -\Delta\phi_j(x) + \tilde{b}(x)\,\phi_j(x) = 0, & x \in \Psi, \\ \partial_n\phi(x) = v_j(x), & x \in \partial\Psi, \end{cases} \tag{6.6}$$

where $\tilde{b}$ is a function chosen so that:

(1) For $x \in \Omega^{(i)}$, we have $\tilde{b}(x) = b(x)$.

(2) The equation (6.6) is easy to solve.

We found it is enough to use 80 sample functions for a given square.

We propose two techniques for constructing the sample functions ϕ_j. The first technique is valid for constant coefficient problems (ie. $b(\mathbf{x}) = c \in \mathbb{C}$). The second technique is valid for all smooth functions $b(\mathbf{x})$.

6.2.2.1 Constant coefficient case

For constant coefficient problems, the fundamental solution $\Phi(\mathbf{x})$ is known. Thus we propose the use of the Method of Fundamental Solutions [33]. Here we give a very brief overview of the method.

Let $\Omega^{(i)}$ have side length a. We place a collection of proxy points $\{\mathbf{x}_j\}_{j=1}^{N_{\text{sam}}}$ along a circle of radius $2\,a$ concentric with $\Omega^{(i)}$, see Figure 6.1. Let $\phi_j(\mathbf{x}) = \Phi(\mathbf{x} - \mathbf{x}_j)$.

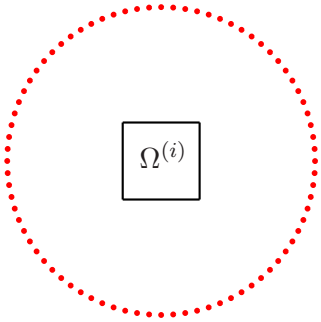


Figure 6.1: Illustration of proxy points (in red).

Since the fundamental solution is translation invariant, the matrix T need only be found for one box of size $\Omega^{(i)}$.

6.2.2.2 Variable coefficient case

Without á priori knowledge of the fundamental solution, a numerical approach to constructing the sample functions is necessary. We propose building ϕ_j via planar wave interpolation. That is

$$\phi_j(x) = \sum_{l=1}^{N_j} c_l e^{\mathbf{i}\mathbf{k}_l \cdot \mathbf{x}}$$

where $\{\mathbf{k}_l\}_{l=1}^{N_j}$ are chosen at random from a normal distribution and $N_j = N_{\text{gauss}}^2$. We choose $\{c_l\}_{l=1}^{N_j}$ such that $\phi_j(x)$ satisfies (6.6) at the two dimensional Gaussian quadrature nodes inside of $\Omega^{(i)}$ (see Figure 6.2). Additionally, we require $\sum_{l=1}^{N_j} c_l = 1$.

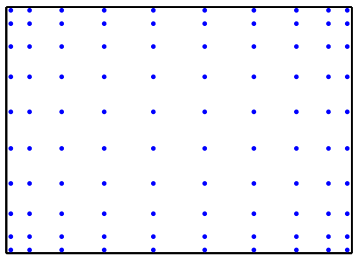


Figure 6.2: Illustration of Gaussian interpolation points.

6.2.3 Assembling a global equilibrium equation

In this section, we formulate a linear equation that relates the following variables:

$$\text{Given data:} \quad \{\mathbf{v}^{(i)} : i \text{ is an edge that is exterior to } \Omega\},$$

$$\text{Sought data:} \quad \{\mathbf{v}^{(i)} : i \text{ is an edge that is interior to } \Omega\}.$$

Let N_{edge} denote the number of interior edges. Then the coefficient matrix of the linear system will consist of $N_{\text{edge}} \times N_{\text{edge}}$ blocks, each of size $N_{\text{gauss}} \times N_{\text{gauss}}$. Each block row in the system will have at most 7 non-zero blocks. To form this matrix, let i denote an interior edge. Suppose i is a vertical edge. Let m and n denote the two boxes that share the edge i, let $\{m_1, m_2, m_3, m_4\}$ denote

the edges of τ_1, and let $\{n_1, n_2, n_3, n_4\}$ denote the edges of τ_2, see Figure 6.5. The Neumann-to-Dirichlet operator for m provides an equation for the boundary fluxes of the left box:

$$\mathbf{u}^{(m_2)} = \mathsf{T}^{(m_1,21)}\,\mathbf{v}^{(m_1)} + \mathsf{T}^{(m_1,22)}\,\mathbf{v}^{(m_2)} + \mathsf{T}^{(m_1,23)}\,\mathbf{v}^{(m_3)} + \mathsf{T}^{(m_1,24)}\,\mathbf{v}^{(m_4)}. \tag{6.7}$$

Analogously, the Neumann-to-Dirichlet operator for n provides the equation

$$\mathbf{u}^{(n_4)} = \mathsf{T}^{(n,41)}\,\mathbf{v}^{(n_1)} + \mathsf{T}^{(n,42)}\,\mathbf{v}^{(n_2)} + \mathsf{T}^{(n,43)}\,\mathbf{v}^{(n_3)} + \mathsf{T}^{(n,44)}\,\mathbf{v}^{(n_4)}. \tag{6.8}$$

Observing that $m_2 = n_2 = i$, we see that $\mathbf{u}^{(m_2)} = \mathbf{u}^{(n_4)}$, and consequently (6.7) and (6.8) can be combined to form the equation

$$\begin{aligned}\mathsf{T}^{(m_1,21)}\,\mathbf{v}^{(m_1)} + \mathsf{T}^{(m,22)}\,\mathbf{v}^{(i)} + \mathsf{T}^{(m,23)}\,\mathbf{v}^{(m_3)} + \mathsf{T}^{(m,24)}\,\mathbf{v}^{(m_4)} \\ = \mathsf{T}^{(n,41)}\,\mathbf{v}^{(n_1)} + \mathsf{T}^{(n,42)}\,\mathbf{v}^{(n_2)} + \mathsf{T}^{(n,43)}\,\mathbf{v}^{(n_3)} + \mathsf{T}^{(n,44)}\,\mathbf{v}^{(i)}.\end{aligned} \tag{6.9}$$

The collection of all equations of the form (6.9) for interior vertical edges, along with the analogous set of equations for all interior horizontal edges forms the global equilibrium equation.

6.3 Efficient direct solvers

This section describes a direct solution technique for the global equilibrium equation constructed in Section 6.2. The idea is to partition the box Ω into a quad-tree of boxes. On each leaf box, the Neumann-to-Dirichlet operator $\mathsf{T}^{(\tau)}$ is constructed via a technique described in Section 6.2.2. We then sweep up through the tree constructing the Neumann-to-Dirichlet operator for a box by merging the operators of its four children boxes.

Let N denote the size of the coefficient matrix. Then Section 6.3.2 describes a procedure with $O(N^{1.5})$ complexity, and Section 6.3.3 outlines how the procedure can be accelerated to $O(N)$ complexity. Before describing the fast solvers, we describe a hierarchical decomposition of the domain in Section 6.3.1.

6.3.1 A quad-tree on the domain

A standard quad-tree is formed on the computational domain Ω as follows: Let $\Omega^{(1)} = \Omega$ be the *root* of the tree, as shown in Figure 6.6(a). Then split $\Omega^{(1)}$ into four boxes that share no interior points

$$\Omega^{(1)} = \Omega^{(2)} \cup \Omega^{(3)} \cup \Omega^{(4)} \cup \Omega^{(5)},$$

as shown in Figure 6.6(b). Continue by splitting each of the four boxes into four smaller equisized boxes:

$$\begin{aligned}
\Omega^{(5)} &= \Omega^{(6)} \cup \Omega^{(7)} \cup \Omega^{(8)} \cup \Omega^{(9)}, \\
\Omega^{(6)} &= \Omega^{(10)} \cup \Omega^{(11)} \cup \Omega^{(12)} \cup \Omega^{(13)}, \\
\Omega^{(7)} &= \Omega^{(14)} \cup \Omega^{(15)} \cup \Omega^{(16)} \cup \Omega^{(17)}, \\
\Omega^{(8)} &= \Omega^{(18)} \cup \Omega^{(19)} \cup \Omega^{(20)} \cup \Omega^{(21)},
\end{aligned}$$

as shown in Figure 6.6(c). The process continues until each box is small enough that the Neumann-to-Dirichlet operator for each leaf can easily be constructed via the procedure described in Section 6.2.2. The levels of the tree are ordered so that $\ell = 0$ is the coarsest level (consisting only of the root), $\ell = 1$ is the level with four boxes, etc. We let L denote the total number of levels in the tree.

6.3.2 Simple construction of the Neumann-to-Dirichlet operator for a parent

Suppose that σ is a box with children ν_1 and ν_3 as shown in Figure 6.8, and that we know the matrices $\mathsf{T}^{(\nu_1)}$ and $\mathsf{T}^{(\nu_3)}$ associated with the children. We seek the matrix $\mathsf{T}^{(\sigma)}$. Recall the equilibrium equations for the two children read

$$\mathbf{u}^{(m_i)} = \sum_{j=1}^{4} \mathsf{T}^{(\nu_1, ij)}\, \mathbf{v}^{(m_j)}, \qquad i = 1,\, 2,\, 3,\, 4, \tag{6.10}$$

$$\mathbf{u}^{(n_i)} = \sum_{j=1}^{4} \mathsf{T}^{(\nu_3, ij)}\, \mathbf{v}^{(n_j)}, \qquad i = 1,\, 2,\, 3,\, 4. \tag{6.11}$$

Observing that $\mathbf{u}^{(m_2)} = \mathbf{u}^{(n_n)}$, we combine (6.10) for $i = 2$ with (6.11) for $i = 4$ to obtain the joint equation

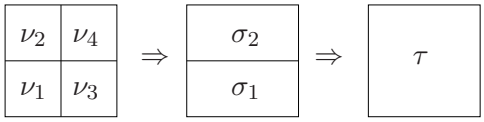
$$\begin{aligned}\mathsf{T}^{(\nu_1,21)}\,\mathbf{v}^{(m_1)} + \mathsf{T}^{(\nu_1,22)}\,\mathbf{v}^{(m_2)} + \mathsf{T}^{(\nu_1,23)}\,\mathbf{v}^{(m_3)} + \mathsf{T}^{(\nu_1,24)}\,\mathbf{v}^{(m_4)} \\ = \mathsf{T}^{(\nu_3,41)}\,\mathbf{v}^{(n_1)} + \mathsf{T}^{(\nu_3,42)}\,\mathbf{v}^{(n_2)} + \mathsf{T}^{(\nu_3,43)}\,\mathbf{v}^{(n_3)} + \mathsf{T}^{(\nu_3,44)}\,\mathbf{v}^{(n_4)}. \quad (6.12)\end{aligned}$$

Further utilizing that $\mathbf{v}^{(m_2)} = \mathbf{v}^{(n_4)}$, we write (6.12) along with (6.10) and (6.11) as

$$\left[\begin{array}{cccccc|c} \mathsf{T}^{(\nu_1,11)} & \mathsf{T}^{(\nu_1,13)} & \mathsf{T}^{(\nu_1,14)} & 0 & 0 & 0 & \mathsf{T}^{(\nu_1,12)} \\ \mathsf{T}^{(\nu_1,31)} & \mathsf{T}^{(\nu_1,33)} & \mathsf{T}^{(\nu_1,34)} & 0 & 0 & 0 & \mathsf{T}^{(\nu_1,32)} \\ \mathsf{T}^{(\nu_1,41)} & \mathsf{T}^{(\nu_1,43)} & \mathsf{T}^{(\nu_1,44)} & 0 & 0 & 0 & \mathsf{T}^{(\nu_1,42)} \\ 0 & 0 & 0 & \mathsf{T}^{(\nu_3,11)} & \mathsf{T}^{(\nu_3,12)} & \mathsf{T}^{(\nu_3,13)} & \mathsf{T}^{(\nu_3,14)} \\ 0 & 0 & 0 & \mathsf{T}^{(\nu_3,21)} & \mathsf{T}^{(\nu_3,22)} & \mathsf{T}^{(\nu_3,23)} & \mathsf{T}^{(\nu_3,24)} \\ 0 & 0 & 0 & \mathsf{T}^{(\nu_3,31)} & \mathsf{T}^{(\nu_3,32)} & \mathsf{T}^{(\nu_3,33)} & \mathsf{T}^{(\nu_3,34)} \\ \hline \mathsf{T}^{(\nu_1,21)} & \mathsf{T}^{(\nu_1,23)} & \mathsf{T}^{(\nu_1,24)} & -\mathsf{T}^{(\nu_3,41)} & -\mathsf{T}^{(\nu_3,42)} & -\mathsf{T}^{(\nu_3,43)} & \mathsf{T}^{(\nu_1,22)} - \mathsf{T}^{(\nu_3,44)} \end{array}\right] \left[\begin{array}{c} \mathbf{v}^{(m_1)} \\ \mathbf{v}^{(m_3)} \\ \mathbf{v}^{(m_4)} \\ \mathbf{v}^{(n_1)} \\ \mathbf{v}^{(n_2)} \\ \mathbf{v}^{(n_3)} \\ \hline \mathbf{v}^{(m_2)} \end{array}\right] = \left[\begin{array}{c} \mathbf{u}^{(m_1)} \\ \mathbf{u}^{(m_3)} \\ \mathbf{u}^{(m_4)} \\ \mathbf{u}^{(n_1)} \\ \mathbf{u}^{(n_2)} \\ \mathbf{u}^{(n_3)} \\ \hline \mathbf{0} \end{array}\right].$$

Eliminating $\mathbf{v}^{(m_2)}$ from the system via a Schur complement yields the operator $\mathsf{T}^{(\sigma)}$.

This procedure merged two boxes. We call it a "merge-two" operation. A "merge-four" operation is obtained by simply combining three merge-two operations. To be precise, suppose that τ is a node with the four children ν_1, ν_2, ν_3, ν_4. We introduce the two "intermediate" boxes σ_1 and σ_2 as shown in the following figure:



Letting the first procedure described earlier in the section be denoted by "`merge_two_horizontal`" and defining an analogous function "`merge_two_vertical`," we then find that the "merge-four" procedure is

$$\begin{aligned}\mathsf{T}^{(\sigma_1)} &= \texttt{merge\_two\_horizontal}(\mathsf{T}^{(\nu_1)}, \mathsf{T}^{(\nu_3)}),\\ \mathsf{T}^{(\sigma_2)} &= \texttt{merge\_two\_horizontal}(\mathsf{T}^{(\nu_2)}, \mathsf{T}^{(\nu_4)}),\\ \mathsf{T}^{(\tau)} &= \texttt{merge\_two\_vertical}(\mathsf{T}^{(\sigma_1)}, \mathsf{T}^{(\sigma_2)}).\end{aligned}$$

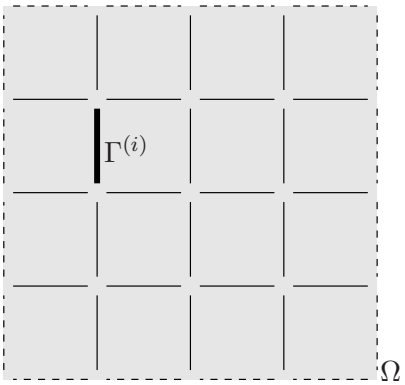


Figure 6.3: The computational box Ω (gray) is split into 16 small boxes. There are a total of 40 edges in the discretization, 24 interior ones (solid lines) and 16 exterior ones (dashed lines). One interior edge $\Gamma^{(i)}$ is marked with a bold line. (Each edge continues all the way to the corner, but has been drawn slightly shortened for clarity.)

6.3.3 Fast construction of the Neumann-to-Dirichlet operator for a parent

The merge operation described in Section 6.3.2 has asymptotic cost $O(N^{1.5})$, where N is the total number of points on the edges of the leaves. To provide a simplified explanation, each merge operation requires a matrix inversion and matrix-matrix-multiplication for dense matrices whose size grows to $O(\sqrt{N}) \times O(\sqrt{N})$. Fortunately, these dense matrices have an internal structure such that the off-diagonal blocks in (6.4) have low rank and the diagonal blocks are Hierarchically Semi-Separable (HSS) matrices. Thus HSS algebra techniques described in Chapter 3 can be utilized resulting in a method that scales linearly.

6.4 Numerical examples

The proposed method was tested on four problems of the form (6.1). The problems considered are:

- **Modified Helmholtz:** Let $b(\mathbf{x}) = k^2$ and $g(\mathbf{x}) = \frac{\partial}{\partial \nu}(\mathcal{K}_1(|k\mathbf{x}|))$ where $\mathcal{K}_1$ is the first order modified Bessel function of the second kind and k is a constant known as the wave number.
- **Helmholtz:** Let $b(\mathbf{x}) = -k^2$ and $g(\mathbf{x}) = \frac{\partial}{\partial \nu}(\mathcal{H}_1(|k\mathbf{x}|))$ where $\mathcal{H}_1$ is the first order Bessel function of the second kind.

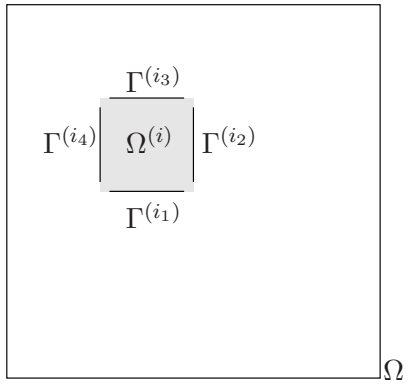


Figure 6.4: The box $\Omega^{(i)}$ is marked in gray. Its edges are $\Gamma^{(i_1)}$, $\Gamma^{(i_2)}$, $\Gamma^{(i_3)}$, $\Gamma^{(i_4)}$.

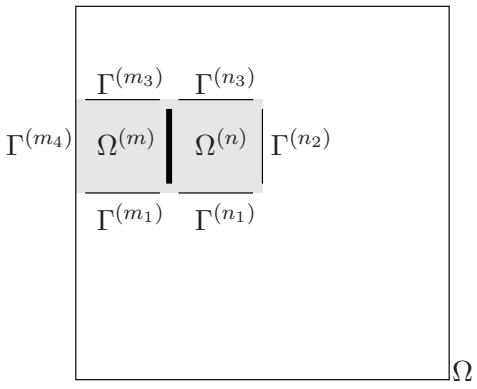


Figure 6.5: Construction of the equilibrium equation for the edge $\Gamma^{(i)}$ in Figure A.1. It is the common edge of the boxes $\Omega^{(m)}$ and $\Omega^{(n)}$, which have edges $\{\Gamma^{(m_1)}, \Gamma^{(m_2)}, \Gamma^{(m_3)}, \Gamma^{(m_4)}\}$, and $\{\Gamma^{(n_1)}, \Gamma^{(n_2)}, \Gamma^{(n_3)}, \Gamma^{(n_4)}\}$, respectively. Observe that $\Gamma^{(i)} = \Gamma^{(m_2)} = \Gamma^{(n_4)}$ (the bold line).

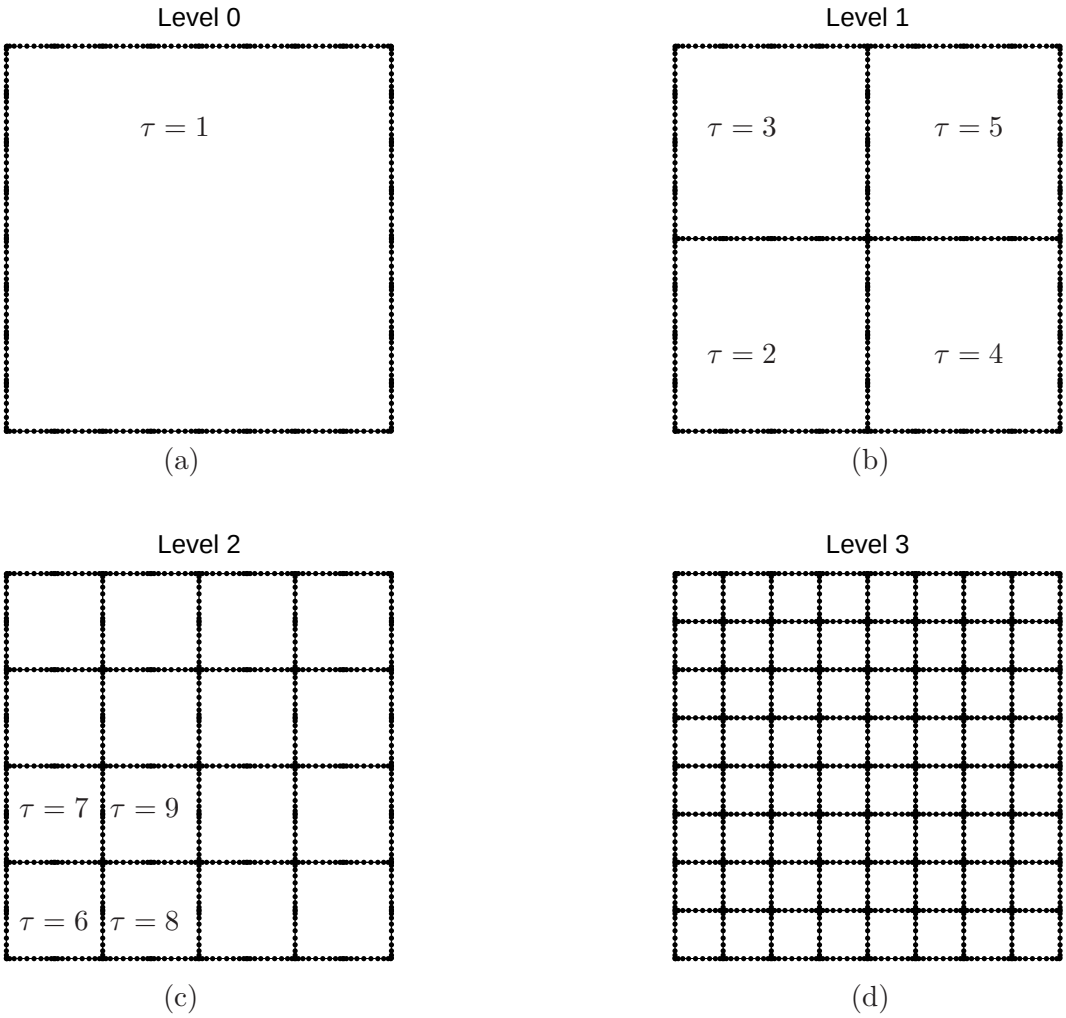


Figure 6.6: Tree structure for a tree with $L = 3$ levels. There are 10 Gaussian nodes on each side of the leaf boxes. The black dots mark the points at which the solution ϕ and its derivative (in the direction normal to the indicated patch boundary) are tabulated.

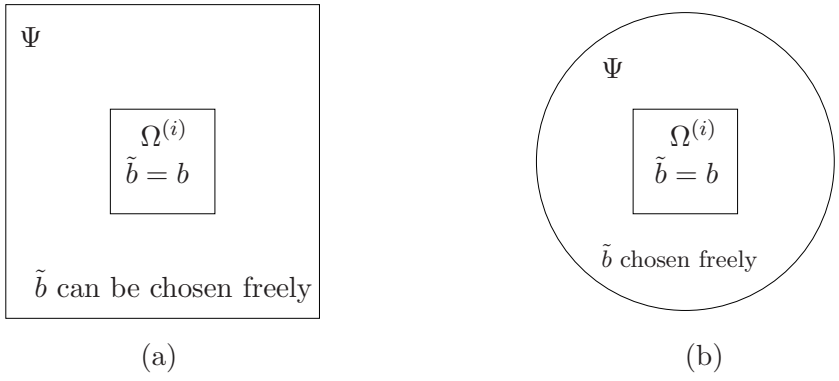


Figure 6.7: Two choices of geometry for the local patch computation. The choice (a) is natural since it conforms to the overall geometry. The advantage of choice (b) is that the FFT can be used in the angular direction.

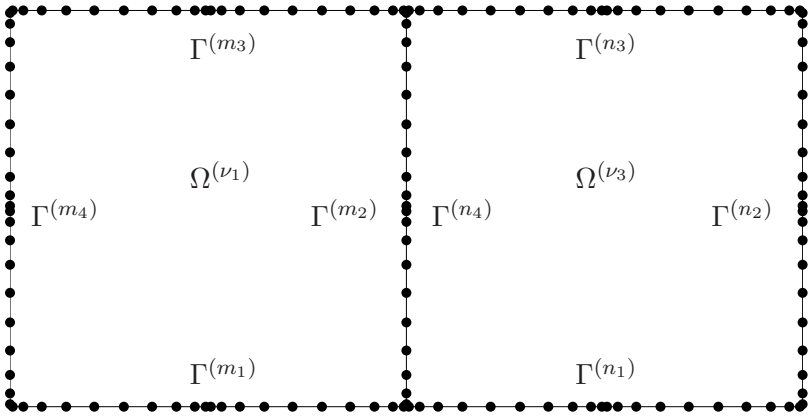


Figure 6.8: Geometry of the *merge* operation.

- **Oscillatory:** Let $b(\mathbf{x}) = \cos(x_1/2)$ and $g(\mathbf{x}) = \frac{\partial}{\partial \nu}(e^{\mathrm{i}k\mathbf{x}\cdot\mathbf{d}})$ where k is a constant, and $\mathbf{d}$ the direction of an incident wave.

- **Decay:** Let $b(\mathbf{x}) = \frac{1}{x_1}$ and $g(\mathbf{x}) = \frac{\partial}{\partial \nu}(e^{\mathrm{i}k\mathbf{x}\cdot\mathbf{d}})$ where k is a constant, and $\mathbf{d}$ the direction of an incident wave.

All codes were executed on a desktop computer with 2.8GHz Intel i7 processor and 12GB of RAM. The method was implemented in Matlab. While this implementation is unoptimized, we believe it is sufficient for illustrating the the potential of the proposed technique.

For the Helmholtz and modified Helmholtz problems, we fix the leaf boxes to have length one and use 10 Gaussian nodes per edge. The Neumann-to-Dirichlet operator need only be computed once per level for these problem. On the leaf level, the approximate operator is computed via the Method of Fundamental Solutions. Figure 6.9 reports the time for the computing the solution on Γ using dense matrix algebra while the side length l of Ω increases from 1 to 64. The error remains approximately 10^{-7}.

For the oscillatory and decay problems, we fix $\Omega = [1, 5]^2$. The leaf boxes start with a side length of 4. At each iteration, we refine the discretization by halving the length of a leaf box. The domain is refined until each leaf box has length 1/8. Figure 6.10 reports the l_∞-norm of difference between the approximate solution on the boundary for two consecutive iterations. Figures 6.11 and 6.12 illustrate the global convergence of the solution.

6.5 Concluding remarks

This chapter presented a new discretization technique for elliptic boundary value problems. Preliminary numerical results were presented. They indicate that the method is capable of solving variable coefficient problems where $b(\mathbf{x})$ is smooth and has bounded variance in magnitude. Additionally, the results indicate that the method is capable of solving constant coefficient problems to high accuracy.

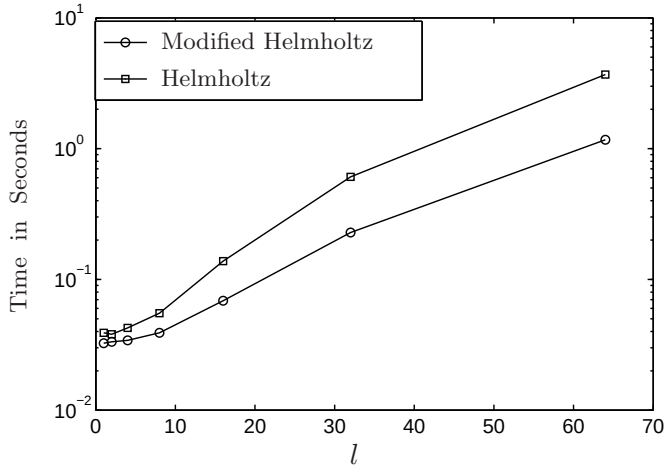


Figure 6.9: Times for constructing the solution on the boundary of $\Omega = [0, l]^2$. $k = 1$

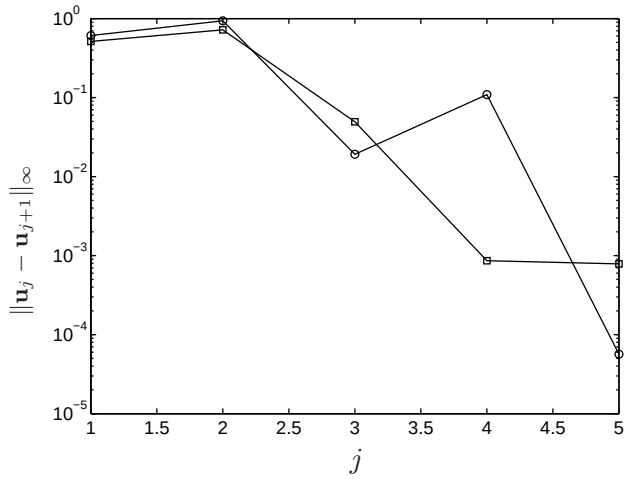


Figure 6.10: l_∞-norm of two consecutive iterations, $k = 1$. Legend: $-\circ-$ - Oscillatory problem $-\square-$ - Decay problem.

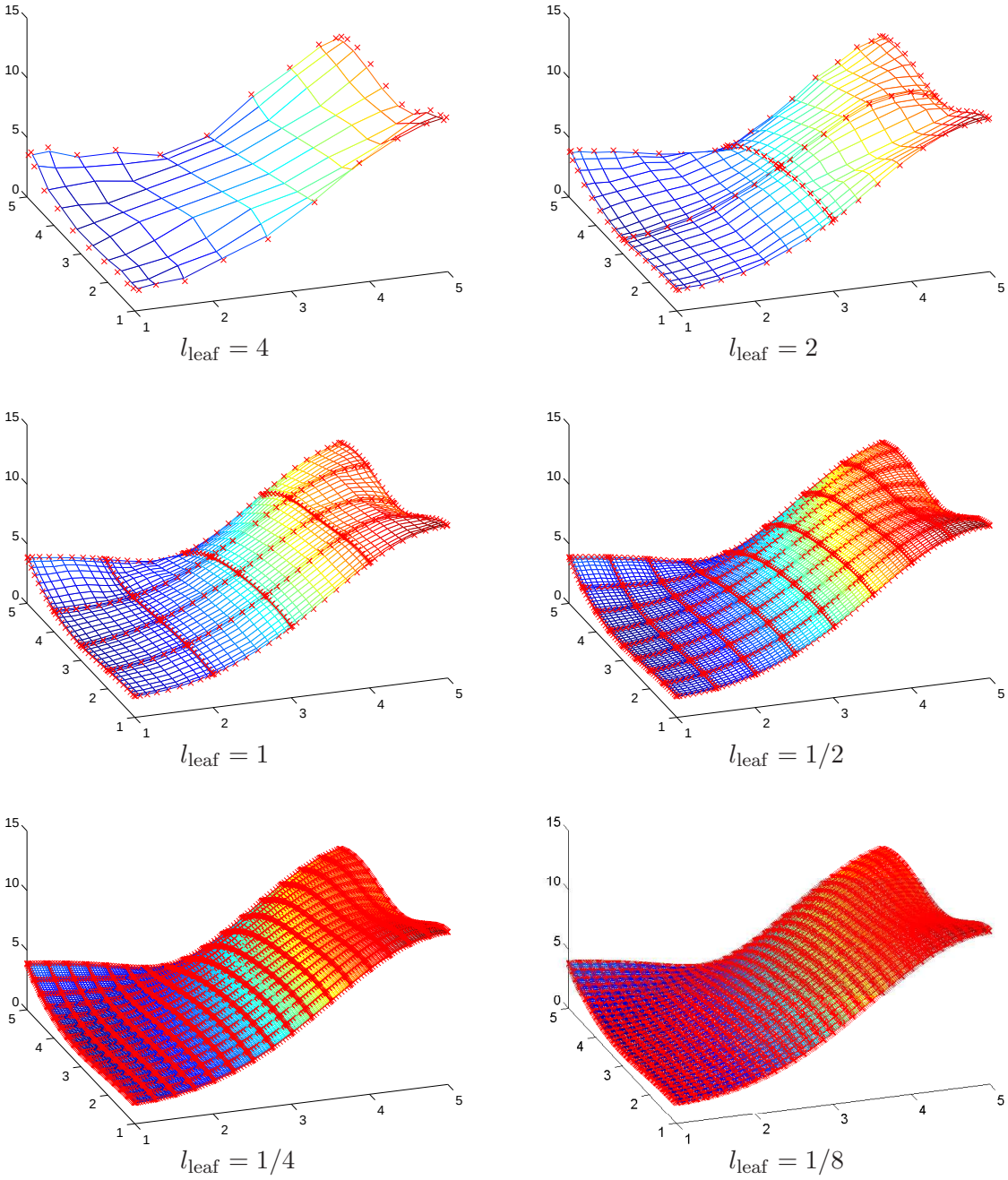


Figure 6.11: Plots of the computed solution to the **Oscillatory** boundary value problem.

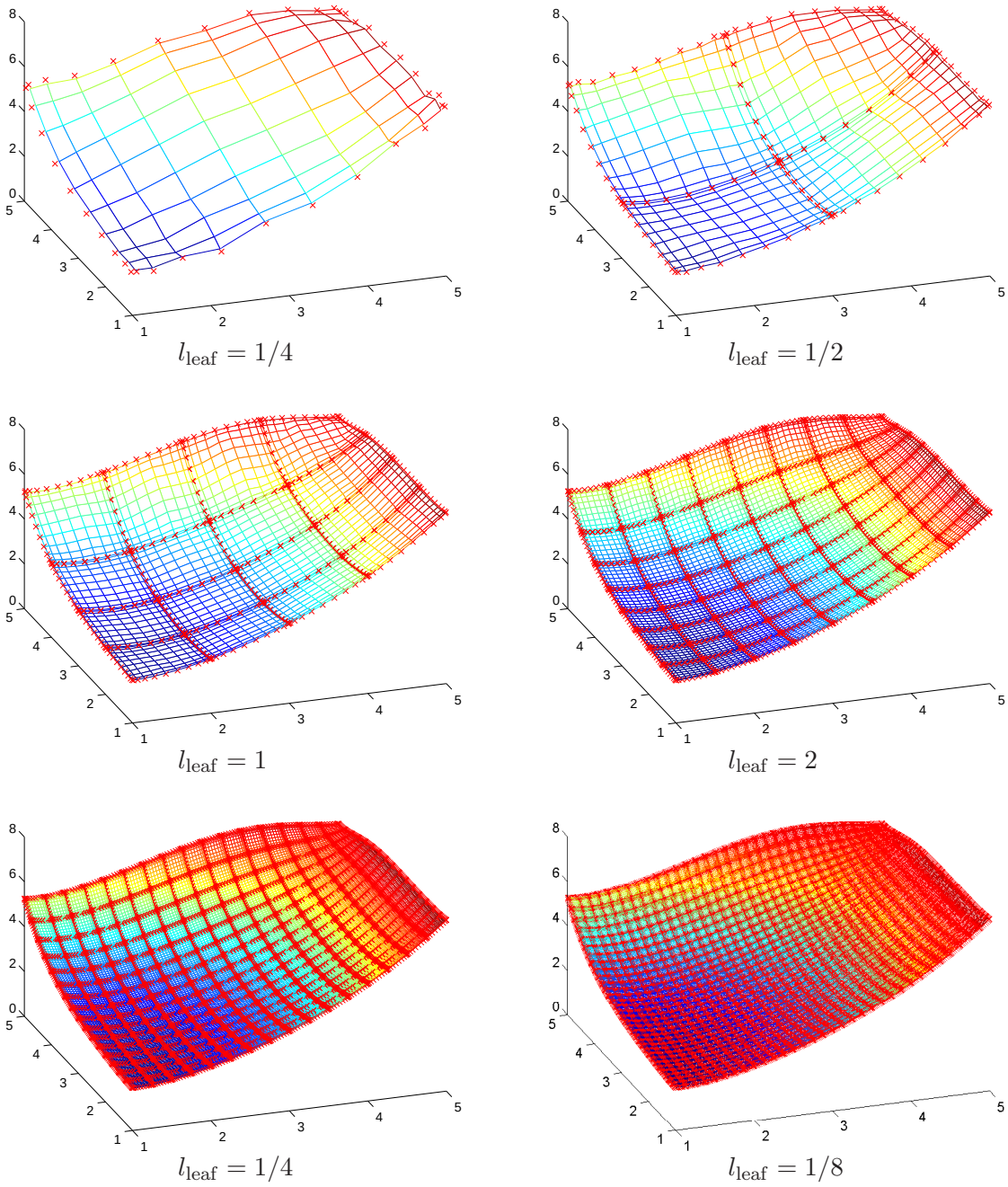


Figure 6.12: Plots of the computed solution to the **Decay** boundary value problem.

Chapter 7

Conclusion

7.1 Summary of key results

A collection of fast direct solvers for elliptic boundary value problems are presented in this dissertation. It is the belief of the author that these solution techniques increase what physical phenomena can be modeled computationally. At the very least, the techniques increase the types of problems that can be solved on standard desktop computer.

The *Hierarchically Semi-Separable* (HSS) solver, presented in Chapter 2, has the ability to solve most one dimensional integral equations in linear time with the added benefit that the cost of each additional solve is minimal. In fact, if the approximate inverse is pre-computed, the solution time can often be one or two orders of magnitude shorter than that of existing state-of-the-art solvers. For example, a linear system corresponding to an elongated domain with corners involving approximately 10^5 discretization points can be solved in about 50 seconds to six digits of accuracy on a standard desktop computer. Each additional solve simply involves a matrix-vector multiply that can be compute in less than one-tenth of a second. Additional numerical results indicate that there is wide range of problems (including high-frequency Helmholtz, boundaries that are "space filling", and integral equations of two dimensional surfaces) for which the method can construct an approximate inverse with $O(N^{1.5})$ computational cost and apply the inverse with a cost that grows linearly with N, where N is the number of discretization points.

For the linear system that arises from the finite element or finite difference discretization of an elliptic PDE, an $O(N^{1.5})$ solution technique, known as the nested dissection method, has existed

since the 1970's. Recently, it was discovered that the intermediate matrices in this method have enough internal structure that a matrix of size $m \times m$ can be stored, multiplied, and inverted in $O(m)$ time. (Several different data sparse formats have been proposed; in this dissertation we use the so-called *Hierarchically Semi-Separable* format.) In Chapter 3, we presented our variation of an accelerated nested dissection method. For many problems, this new technique scales linearly with the number of discretization points and is very fast for solving a pure boundary value problem with multiple right-hand sides. The numerical examples report that the first for a system involving 16 million unknowns takes about 7 minutes on a desktop computer. Each additional solve takes about 0.04 seconds. For moderately ill-conditioned linear systems, the direct solver can be used as preconditioner. Unlike many existing preconditioners, the construction of one using nested dissection techniques is not problem specific. Several groups [70, 18, 53] are currently working on implementing similar solution techniques in this capacity.

In the special case of finite element or finite difference discretization of constant coefficient elliptic problems, we have developed fast techniques that utilize the existence of a discrete fundamental solution. For the free-space problem, the solution is given by a convolution of the fundamental solution with the source charges. The lattice FMM presented in Chapter 4 computes the convolution efficiently. For example, a Poisson equation on a $10^6 \times 10^6$ lattice, of which 10^6 points were loaded, was solved to ten digits of accuracy in three minutes on a desktop computer. In Chapter 5, we propose the use of boundary algebraic equations to solve discrete boundary value problems. Similar to the linear system that arises from the discretization of many boundary integral equation, the boundary algebraic equation has internal structure. This structure is such that the solution techniques of Chapter 2 can be applied. Thus the linear system can be solved with computational cost $O(N_{\text{boundary}})$, where N_{boundary} is the number of points on the boundary.

In addition to the fast solvers, preliminary results for a new discretization scheme were presented in Chapter 6. The method constructs a global solution operator from operators defined on subdomains in a hierarchical fashion. This new scheme is high-order accurate and well-suited for fast direct solvers.

7.2 Extensions and future work

The direct solvers described in this dissertation are immediately applicable to several important applications (see Section 7.1). Moreover, we expect the new techniques to be useful in many additional environments. Directions that are currently under investigation include:

Homogenization methods using fast direct solvers: Appendix A and [38] illustrate how homogenization techniques in conjunction with fast dense matrix algebra for structured matrices can rapidly build approximate solution operators.

Fast direct solvers for volume integral equations in the plane: Consider the linear system that arises from the Nyström discretization of the Lippmann-Schwinger equation. This system has internal structure, however it is different from the structure exploited in building the fast method in Chapter 2. A linear inversion method for this linear system is currently under development. The key to its construction is recursing on dimension.

Fast direct solvers for surface integral equations in $\mathbb{R}^3$*:* This solver will be an extension of the fast direct solver for volume integral equations in the plane.

Fast direct solvers for finite element and finite difference equations on meshes in $\mathbb{R}^3$*:* This solver will be a three-dimensional version of the accelerated nested dissection method. With the increase in dimensionality, the Schur complements will lie on the boundary of three-dimensional boxes with a internal structure similar to that of the discretized volume integral equation in the plane. Thus the acceleration will come from the linear solver developed for discretized integral equations in the plane.

Bibliography

[1] C. R. Anderson. An implementation of the fast multipole method without multipoles. SIAM J. Sci. Statist. Comput., 13(4):923–947, 1992.

[2] U. Andersson, B. Engquist, G. Ledfelt, and O. Runborg. A contribution to wavelet-based subgrid modeling. Appl. Comput. Harmon. Anal., 7(2):151–164, 1999.

[3] K. Atkinson. The numerical solution of integral equations of the second kind. Cambridge University Press, Cambridge, 1997.

[4] N. Bakhvalov and G. Panasenko. Homogenisation: averaging processes in periodic media, volume 36 of Mathematics and its Applications (Soviet Series). Kluwer Academic Publishers Group, Dordrecht, 1989. Mathematical problems in the mechanics of composite materials, Translated from the Russian by D. Leĭtes.

[5] J. Barnes and P. Hut. A hierarchical $o(n \log n)$ force-calculation algorithm. Nature, 324(4), 1986.

[6] L. Beatson and L. Greengard. A short course on fast multipole methods.

[7] G. Beylkin, R. Coifman, and V. Rokhlin. Wavelets in numerical analysis. In Wavelets and their applications, pages 181–210. Jones and Bartlett, Boston, MA, 1992.

[8] S. Börm. $\mathcal{H}^2$-matrix arithmetics in linear complexity. Computing, 77(1):1–28, 2006.

[9] S. Börm. Approximation of solution operators of elliptic partial differential equations by $\mathcal{H}$-and $\mathcal{H}^2$–matrices. Technical Report 85/2007, Max Planck Institute, 2007.

[10] S. Börm. Construction of data-sparse $\mathcal{H}^2$-matrices by hierarchical compression. Technical Report 92/2007, Max Planck Institute, 2007.

[11] S. Börm. Efficient Numerical Methods for Non-local Operators: $\mathcal{H}^2$-Matrix Compression, Algorithms and Analysis. European Mathematics Society, 2010.

[12] J. Bremer and V. Rokhlin. Efficient discretization of Laplace boundary integral equations on polygonal domains. J. Comput. Phys., 229:2507–2525, 2010.

[13] M. E. Brewster and G. Beylkin. A multiresolution strategy for numerical homogenization. Appl. Comput. Harmon. Anal., 2(4):327–349, 1995.

[14] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. SIAM J. Sci. and Stat. Comput., 9(4):669–686, 1988.

[15] S. Chandrasekaran and M. Gu. Fast and stable algorithms for banded plus semiseparable systems of linear equations. SIAM J. Matrix Anal. Appl., 25(2):373–384 (electronic), 2003.

[16] S. Chandrasekaran and M. Gu. A divide-and-conquer algorithm for the eigendecomposition of symmetric block-diagonal plus semiseparable matrices. Numer. Math., 96(4):723–731, 2004.

[17] S. Chandrasekaran, M. Gu, X.S. Li, and J. Xia. Some fast algorithms for hierarchically semiseparable matrices. Technical Report 08-24, UCLA/CAM, 2008.

[18] S. Chandrasekaran, M. Gu, X.S. Li, and J. Xia. Superfast multifrontal method for large structured linear systems of equations. SIAM J. Matrix Anal. Appl., 31:1382–1411, 2009.

[19] S. Chandrasekaran, M. Gu, X.S. Li, and J Xia. Fast algorithms for hierarchically semiseparable matrices. Numer. Linear Algebra Appl., 17:953–976, 2010.

[20] S. Chandrasekaran, M. Gu, and W. Lyons. A fast adaptive solver for hierarchically semiseparable representations. Calcolo, 42(3-4):171–185, 2005.

[21] H. Cheng, Z. Gimbutas, P. G. Martinsson, and V. Rokhlin. On the compression of low rank matrices. SIAM J. Sci. Comput., 26(4):1389–1404, 2005.

[22] D. Cioranescu and J. Saint Jean Paulin. Homogenization of reticulated structures, volume 136 of Applied Mathematical Sciences. Springer-Verlag, New York, 1999.

[23] T. Degrands and C. De Tar. Lattice Methods for Quantum Chromodynamics. World Scientific Publishing Company, Hackensack, NJ, 2006.

[24] V.S. Deshpande, N.A. Fleck, and M.F. Ashby. Effective properties of the octet-truss lattice material. Journal of the Mechanics and Physics of Solids, 49(8):1747 – 1769, 2001.

[25] P. Dewilde and S. Chandrasekaran. A hierarchical semi-separable Moore-Penrose equation solver. In Wavelets, multiscale systems and hypercomplex analysis, volume 167 of Oper. Theory Adv. Appl., pages 69–85. Birkhäuser, Basel, 2006.

[26] M. Dorobantu and B. Engquist. Wavelet-based numerical homogenization. SIAM J. Numer. Anal., 35(2):540–559 (electronic), 1998.

[27] R. J. Duffin. Discrete potential theory. Duke Math. J., 20:233–251, 1953.

[28] R. J. Duffin and E. P. Shelly. Difference equations of polyharmonic type. Duke Math. J., 25:209–238, 1958.

[29] Y. Efendiev and T. Hou. Multiscale finite element methods, volume 4 of Surveys and Tutorials in the Applied Mathematical Sciences. Springer, New York, 2009. Theory and applications.

[30] B. Engquist and O. Runborg. Wavelet-based numerical homogenization with applications. In Multiscale and multiresolution methods, volume 20 of Lect. Notes Comput. Sci. Eng., pages 97–148. Springer, Berlin, 2002.

[31] B. Engquist and O. Runborg. Wavelet-based numerical homogenization. In Highly oscillatory problems, volume 366 of London Math. Soc. Lecture Note Ser., pages 98–126. Cambridge Univ. Press, Cambridge, 2009.

[32] M.H. Ernst and P.F.J. Van Velthoven. Random walks on cubic lattices with bond disorder. Journal Journal of Statistical Physics, 45(5-6):1001–1030, 1986.

[33] G. Fairweather and A. Karageorghis. The method of fundamental solutions for elliptic boundary value problems. Advances in Computational Mathematics, 9:69–95, 1998.

[34] J. Fish and A. Wagiman. Multiscale finite element method for a locally nonperiodic heterogeneous medium. Computational Mechanics, 12:164–180, 1993. 10.1007/BF00371991.

[35] A. George. Nested dissection of a regular finite element mesh. SIAM J. Numer. Anal., 10:345–363, 1973.

[36] A. Gillman and P.G. Martinsson. Fast and accurate numerical methods for solving elliptic difference equations defined on lattices. J. Comput. Phys., 229(24):9026–9041, 2010.

[37] A. Gillman and P.G. Martinsson. A fast solver for poisson problems on infinite regular lattices. 2011. arXiv.org report 1105.3505.

[38] A. Gillman, P. Young, and P.G. Martinsson. Numerical homogenization via approximation of the solution operator. In Multiscale Modeling and Simulation in Science, 82. Lecture Notes in Computational Science and Engineering, 2011.

[39] A. Gillman, P. Young, and P.G. Martinsson. A direct solver with $o(n)$ complexity for integral equations on one-dimensional domains. In review.

[40] Z. Gimbutas and V. Rokhlin. A generalized fast multipole method for nonoscillatory kernels. SIAM J. Sci. Comput., 24(3):796–817 (electronic), 2002.

[41] G. Golub and C. Van Loan. Matrix computations. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[42] L. Greengard. The rapid evaluation of potential fields in particle systems. ACM Distinguished Dissertations. MIT Press, Cambridge, MA, 1988.

[43] L. Greengard, D. Gueyffier, P.G. Martinsson, and V. Rokhlin. Fast direct solvers for integral equations in complex three-dimensional domains. Acta Numerica, 18:243–275, 2009.

[44] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. J. Comput. Phys., 73(2):325–348, 1987.

[45] L. Greengard and V. Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. In Acta numerica, 1997, volume 6 of Acta Numer., pages 229–269. Cambridge Univ. Press, Cambridge, 1997.

[46] M. Gu and S. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. SIAM J. Sci. Comput., 17(4):848–869, 1996.

[47] A. Hansen H. J. Herrmann and S. Roux. Fracture of disordered, elastic lattices in two dimensions. Phys. Rev. B, 39(1):637–648, 1989.

[48] W. Hackbusch. The panel clustering technique for the boundary element method (invited contribution). In Boundary elements IX, Vol. 1 (Stuttgart, 1987), pages 463–474. Comput. Mech., Southampton, 1987.

[49] W. Hackbusch. A sparse matrix arithmetic based on H-matrices; Part I: Introduction to H-matrices. Computing, 62:89–108, 1999.

[50] W. Hackbusch, B. Khoromskij, and S. Sauter. On $\mathcal{H}^2$-matrices. In Lectures on Applied Mathematics, pages 9–29. Springer Berlin, 2002.

[51] J. Helsing and R. Ojala. Corner singularities for elliptic problems: Integral equations, graded meshes, quadrature, and compressed inverse preconditioning. J. Comput. Phys., 227:8820–8840, 2008.

[52] S. Kapur and V. Rokhlin. High-order corrected trapezoidal quadrature rules for singular functions. SIAM J. Numer. Anal., 34:1331–1356, 1997.

[53] R. Kriemann L. Grasedyck and S. Le Borne. Domain decomposition based $\mathcal{H}$-LU preconditioning. Numer. Math., 112(4):565–600, 2009.

[54] J. Makino. Yet another fast multipole method without multipoles—pseudoparticle multipole method. J. Comput. Phys., 151(2):910–920, 1999.

[55] A. A. Maradudin, E. W. Montroll, G. H. Weiss, Robert Herman, and H. W. Milnes. Green's functions for monatomic simple cubic lattices. Acad. Roy. Belg. Cl. Sci. Mém. Coll. in-4 deg. (2), 14(7):176, 1960.

[56] P.G. Martinsson. Fast multiscale methods for lattice equations. PhD thesis, University of Texas at Austin, Computational and Applied Mathematics, 2002.

[57] P.G. Martinsson. A fast algorithm for compressing a matrix into a data-sparse format via randomized sampling. Technical report, 2008. arXiv.org report 0806.2339.

[58] P.G. Martinsson. A fast direct solver for a class of elliptic partial differential equations. J. Sci. Comput., 38(3):316–330, 2009.

[59] P.G Martinsson and G. J. Rodin. Boundary algebraic equations for lattice problems. Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci., 465(2108):2489–2503, 2009.

[60] P.G. Martinsson and G.J. Rodin. Asymptotic expansions of lattice green's functions. Proc. Royal Soc. A, 458(2027):2609 – 2622, 2002.

[61] P.G. Martinsson and V. Rokhlin. A fast direct solver for boundary integral equations in two dimensions. J. Comp. Phys., 205(1):1–23, 2005.

[62] P.G. Martinsson and V. Rokhlin. A fast direct solver for boundary integral equations in two dimensions. J. Comput. Phys., 205(1):1–23, 2005.

[63] P.G. Martinsson and V. Rokhlin. An accelerated kernel independent fast multipole method in one dimension. SIAM Journal of Scientific Computing, 29(3):1160–11178, 2007.

[64] P.G. Martinsson and V. Rokhlin. A fast direct solver for scattering problems involving elongated structures. Journal of Computational Physics, 221:288 – 302, 2007.

[65] P.G. Martinsson, V. Rokhlin, and M. Tygert. On interpolation and integration in finite-dimensional spaces of bounded functions. Communications in Applied Mathematics and Computational Science, 1, 2006.

[66] E. Michielssen, A. Boag, and W. C. Chew. Scattering from elongated objects: direct solution in $O(N \log^2 N)$ operations. IEE Proc. Microw. Antennas Propag., 143(4):277 – 283, 1996.

[67] I. Montvay and G. Mnster. Quantum Fields on a Lattice. Cambridge Monographs on Mathematical Physics. Cambridge University Press, Cambridge, 1997.

[68] S. T. O'Donnell and V. Rokhlin. A fast algorithm for the numerical evaluation of conformal mappings. SIAM J. Sci. Stat. Comput., 10:475–487, May 1989.

[69] E. Schlangen and E. J. Garboczi. New method for simulating fracture using an elastically uniform random geometry lattice. Int. J. Engng. Sci., 34(10):1131–1144, 1996.

[70] P. Schmitz and L. Ying. A fast direct solver for elliptic problems on general meshes in 2d, 2010. In review.

[71] Z. Sheng, P. Dewilde, and S. Chandrasekaran. Algorithms to solve hierarchically semi-separable systems. In System theory, the Schur algorithm and multidimensional analysis, volume 176 of Oper. Theory Adv. Appl., pages 255–294. Birkhäuser, Basel, 2007.

[72] P. Starr and V. Rokhlin. On the numerical solution of two-point boundary value problems. II. Comm. Pure Appl. Math., 47(8):1117–1159, 1994.

[73] V. E. Henson W. Briggs and S. McCormick. A multigrid tutorial. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2000.

[74] J. C. Wallach and L. J. Gibson. Mechanical behavior of a three-dimensional truss material. International Journal of Solids and Structures, 38(40-41):7181 – 7196, 2001.

[75] Y. Wang and P. Mora. Macroscopic elastic properties of regular lattices. J. Mech. Phys. Solids, 56(12):2459–3474, 2008.

[76] H. Xiao, V. Rokhlin, and N. Yarvin. Prolate spheroidal wavefunctions, quadrature and interpolation. Inverse Problems, 17(4), 2001.

[77] J. Xu and L. Zikatanov. On an energy minimizing basis for algebraic multigrid methods. Computing and Visualization in Science, 7:121–127, 2004. 10.1007/s00791-004-0147-y.

[78] L. Ying, G. Biros, and D. Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. J. Comput. Phys., 196(2):591–626, 2004.

[79] P. Young and P.G. Martinsson. A direct solver for the rapid solution of boundary integral equations on axisymmetric surfaces in three dimensions, 2009. arxiv.org report #1002.2001.

Appendix A

Numerical homogenization via approximation of the solution operator

A.1 Introduction

A.1.1 Background

The purpose of this report is to draw attention to a number of recent developments in computational harmonic analysis that may prove helpful to the construction of simplified models for heterogeneous media. We consider problems modeled by elliptic PDEs such as electrostatics and linear elasticity in composite materials, and Stokes' flow in porous media.

Many different solution approaches have been proposed for the type of problems under consideration. A classical technique that works relatively well in situations where there is a clear separation of length-scales is to derive so-called *homogenized equations* which accurately model the macro-scale behavior of the constitutive equations without fully resolving the micro-structure. The homogenized equations can sometimes be derived analytically, but they are typically obtained from numerically solving a set of equations defined on a *Representative Volume Element* (RVE). An unfortunate aspect of this approach is that its accuracy is held hostage to many factors that are outside of the control of the modeler. Phenomena that tend to lead to less accurate solutions include:

(1) Concentrated loads.

(2) Boundaries, in particular non-smooth boundaries.

(3) Irregular micro-structures.

The accuracy cannot readily be improved using generic techniques, but a number of strategies for developing coarse-grained models for specific situations have been developed. A popular class of such methods consists of variations of finite element methods in which a discretization on the macro-scale is constructed by solving a set of local problems defined on a representative collection of patches of fully resolved micro-structure [29, 34, 77].

We contend that it is in many situations advantageous to approximate the *solution operator*, rather than the *differential operator.* For the elliptic problems under consideration in this paper, the solution operator takes the form of an integral operator with the Green's function of the problem as its kernel. That such operators should in principle allow compressed representations has been known for some time (at least since [7]), but efficient techniques for actually computing them have become available only recently.

To illustrate the viability of the proposed techniques, we demonstrate how they apply to a couple of archetypical model problems. We first consider situations in which the micro-structure needs to be fully resolved and a coarse-grained model be constructed computationally. We show that this computation can be executed efficiently, and that once it has been, the reduced model allows for very fast solves, and is highly accurate even in situations that are challenging to existing coarse-graining methods. We then show that the proposed methods can fully exploit the simplifications possible when an accurate model of the material can be derived from computations on an RVE.

A.1.2 Mathematical problem formulation

While the ideas described are applicable in a broad range of environments, we will for expositional clarity focus on scalar elliptic boundary value problems defined on some regular domain $\Omega \subset \mathbb{R}^2$ with boundary Γ. Specifically, we consider Neumann problems of the form

$$\begin{cases} -\nabla \cdot \big(a(x) \cdot \nabla u(x)\big) = 0, & x \in \Omega, \\ u_n(x) = f(x), & x \in \Gamma, \end{cases} \tag{A.1}$$

where $a : \Omega \rightarrow \mathbb{R}^{2\times 2}$ is a matrix-valued function that varies "rapidly" (on the length-scale of the micro-structure), and where $u_n(x)$ denotes the normal derivative of u at $x \in \Gamma$. Our objective is to

rapidly construct $u|_\Gamma$, from a given boundary function f. We are interested both in the situation where we are allowed a pre-computation involving some given function a, and in the situation in which a is specified probabilistically.

Some of our numerical work will focus on the special case where (A.1) represents a two-phase material. To be precise, we suppose that Ω can be partitioned into two disjoint "phases," $\bar{\Omega} = \bar{\Omega}_1 \cup \bar{\Omega}_2$, and that there exist constants a_1 and a_2 such that

$$a(x) = \begin{cases} a_1 I, & x \in \Omega_1, \\ a_2 I, & x \in \Omega_2, \end{cases}$$

where I is the identity matrix. We further suppose that $\bar{\Omega}_2$ is wholly contained inside Ω, and let Γ_{int} denote the boundary between Ω_1 and Ω_2, see Figure A.1. Then (A.1) can more clearly be written

$$\begin{cases} -a_1 \Delta u(x) = 0, & x \in \Omega_1, \\ -a_2 \Delta u(x) = 0, & x \in \Omega_2, \\ [u](x) = 0, & x \in \Gamma_{\text{int}}, \\ [a\, u_n](x) = 0, & x \in \Gamma_{\text{int}}, \\ u_n(x) = f(x), & x \in \Gamma, \end{cases} \tag{A.2}$$

where for $x \in \Gamma$, $[u](x)$ and $[a\, u_n](x)$ denote the jumps in the potential and in the *flow* $-a(x)\nabla u(x)$ in the normal direction, respectively.

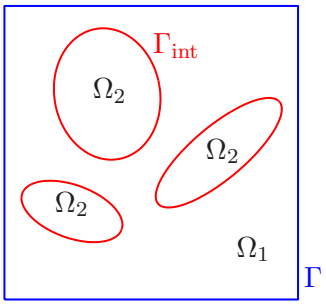


Figure A.1: A two phase domain.

While the current paper concerns only situations modeled by equations of the types (A.1)

and (A.2), the methodology extends to more general elliptic differential equations, see Section A.5.

A.1.3 Coarse-graining of the differential operator (homogenization)

A classical technique [4, 22] for handling a problem such as (A.1) with a rapidly varying coefficient function a is to construct a function a_{hom} that varies on the macroscale only (or may even be constant) such that the solution u is in some sense approximated by the solution u_{hom} to

$$\begin{cases} -\nabla \cdot \big(a_{\text{hom}}(x) \cdot \nabla u_{\text{hom}}(x)\big) = 0, & x \in \Omega, \\ \partial_n \, u_{\text{hom}}(x) = f(x), & x \in \Gamma. \end{cases} \tag{A.3}$$

The derivation of an equation such as (A.3) typically relies on fairly strong assumptions on separation of length-scales, rendering this technique problematic in situations involving boundary effects, concentrated loads, multiple or undifferentiated length-scales, *etc.* A common technique for ameliorating these difficulties is to preserve a piece of the fully resolved micro-structure near the boundary, or the concentrated load, and then to "glue" the two models together.

Another common approach is to forego the construction of a coarse-grained continuum model and construct an equation involving a discretized differential operator whose solution in some sense captures the macro-scale behavior of the solution of (A.3), see *e.g.* [29]. The elements of the discretized matrix are typically constructed via local computations on patches of micro-structure.

A.1.4 Coarse-graining of the solution operator

The premise of our work is that it is possible, and often advantageous, to approximate the *solution operator* of (A.1), rather than the differential operator itself. We will demonstrate that with this approach, many of the difficulties encountered in common coarse-graining strategies can be side-stepped entirely. To be precise, we note that mathematically, the solution to (A.1) takes the form

$$u(x) = [K\, f](x) = \int_\Gamma G(x,y)\, f(y)\, ds(y), \qquad x \in \Gamma, \tag{A.4}$$

where G is a kernel function that depends both on the function a, and on the domain Ω. It is known analytically only in the most trivial cases (such as a being constant, and Ω being a square or

a circle). However, it turns out that the solution operator can be constructed numerically relatively cheaply, and that it admits very data-sparse representations.

Roughly speaking, our proposal is that instead of seeking an approximation of the form (A.3) of (A.1), it is often advantageous to seek an approximation of the form

$$u_{\text{hom}}(x) = [K_{\text{hom}}\, f](x) = \int_\Gamma G_{\text{hom}}(x,y)\, f(y)\, ds(y), \qquad x \in \Gamma.$$

of (A.4). The purpose of the manuscript is to demonstrate the basic viability and desirability of this approach. Specifically, we seek to:

(1) Demonstrate via numerical examples that the solution operators can to high precision be approximated by "data-sparse" representations.

(2) Illustrate a framework in which highly accurate reduced models can be constructed even for situations involving boundary effects, and concentrated loads.

(3) Demonstrate that the reduced models can in many instances be computed inexpensively from statistical experiments on RVEs.

(4) Demonstrate that in situations where the full micro-structure needs to be resolved, there exist highly efficient techniques for doing so, and that the resulting reduced models form natural building blocks in computational models.

Remark 32. *In this paper, we focus on problems with no body load, such as (A.1). However, the ideas set out can equally well be applied to problems such as*

$$\begin{cases} -\nabla \cdot \big(a(x) \cdot \nabla u(x)\big) = h(x), & x \in \Omega, \\ \qquad\qquad u_n(x) = f(x), & x \in \Gamma. \end{cases} \tag{A.5}$$

The mathematical solution operator then contains two terms, one corresponding to each of the two data functions f and h,

$$u(x) = \int_\Gamma G(x,y)\, f(y)\, ds(y) + \int_\Omega K(x,y)\, h(y)\, dA(y), \qquad x \in \Omega. \tag{A.6}$$

The second term in (A.6) is compressible in a manner very similar to that of the first.

Remark 33. *A reason why approximation of the solution operator may prove advantageous compared to approximating the differential operator is hinted at by the spectral properties of the problem. For a bounded domain, an elliptic operator A such as the one defined by equation (A.1) or (A.2) typically has a discrete spectrum $(\lambda_n)_{n=1}^{\infty}$, where $\lambda_n \to \infty$, and where eigenfunctions get more oscillatory the larger λ_n is. In up-scaling A, we seek to construct an operator $A_{\rm hom}$ whose low eigenvalues and eigenfunctions approximate those of A. Measuring success is tricky, however, since the operator $A - A_{\rm hom}$ is in many ways dominated by the high eigenvalues. One way of handling this is to consider multi-scale representations of the operators, see, e.g., [2, 13, 26, 30, 31]. Another way is to try to approximate the inverse of the operator. We observe that A^{-1} is typically compact, and its dominant eigenmodes are precisely those that we seek to capture. Roughly speaking, we advocate the numerical construction of a finite dimensional operator T such that $||A^{-1} - T||$ is small.*

Remark 34. *Our goal with this paper is not to set up a mathematical analysis of the properties of kernels such as the function G in (A.4). However, to give a sense of the type of questions that arise, let us consider a situation where the function a in (A.1) represents a micro-structure with a characteristic length-scale λ. We then let d denote a cut-off parameter that separates the near-field from the far-field, say $d = 5\,\lambda$, and set*

$$G_{\rm near}(x,y) = \begin{cases} G(x,y), & |x-y| \leq d, \\ 0, & |x-y| > d, \end{cases} \qquad G_{\rm far}(x,y) = \begin{cases} 0, & |x-y| \leq d, \\ G(x,y), & |x-y| > d, \end{cases}$$

and

$$u_{\rm near}(x) = \int_\Gamma G_{\rm near}(x,y)\, f(y)\, ds(y), \qquad u_{\rm far}(x) = \int_\Gamma G_{\rm far}(x,y)\, f(y)\, ds(y).$$

The function $y \mapsto G_{\rm near}(x,y)$ depends strongly on the local micro-structure near x, and cannot easily be compressed. This part of the operator must be resolved sufficiently finely to fully represent the micro-structure. However, this is a local interaction, and $u_{\rm near}$ can be evaluated cheaply once $G_{\rm near}$ has been determined. In contrast, $G_{\rm far}$ is compressible. If Γ_1 and Γ_2 are two non-touching pieces of the boundary, then the integral operator

$$[T_{\Gamma_1 \leftarrow \Gamma_2}\sigma](x) = \int_{\Gamma_2} G_{\rm far}(x,y)\, \sigma(y)\, ds(y), \qquad x \in \Gamma_1,$$

is not only compact, but its singular values typically decay exponentially fast, with the rate of decay depending on the sizes of Γ_1 *and* Γ_2*, and on the distance between them. More careful analysis of these issues in an appropriate multi-scale framework can be found in [53].*

A.2 Data-sparse matrices

A ubiquitous task in computational science is to rapidly perform linear algebraic operations involving very large matrices. Such operations typically exploit special *structure* in the matrix since the costs for methods capable of handling general matrices tend to scale prohibitively fast with matrix size: For a general $N \times N$ matrix, it costs $O(N^2)$ operations to perform a matrix-vector multiplication, $O(N^3)$ operations to perform Gaussian elimination or to invert the matrix, *etc.* A well-known form of structure in a matrix is sparsity. When at most a few entries in each row of the matrix are non-zero (as is the case, *e.g.*, for matrices arising upon the discretization of differential equations, or representing the link structure of the World Wide Web) matrix-vector multiplications can be performed in $O(N)$ operations instead of $O(N^2)$. The description *data-sparse* applies to a matrix that may be dense, but that shares the key characteristic of a sparse matrix that some linear algebraic operations, typically the matrix-vector multiplication, can to high precision be executed in fewer than $O(N^2)$ operations (often in close to linear time).

There are many different types of data-sparse representations of a matrix. In this paper, we will utilize techniques for so-called *Hierarchically Semi-Separable* (HSS) matrices [17, 20, 71], which arise upon the discretization of many of the integral operators of mathematical physics, in signal processing, in algorithms for inverting certain finite element matrices, and in many other applications, see *e.g.* [18, 58, 71]. An HSS matrix is a dense matrix whose off-diagonal blocks are rank-deficient in a certain sense. Without going into details, we for now simply note that an HSS matrix A can be expressed via a recursive formula in L levels,

$$\mathsf{A}^{(\ell)} = \mathsf{U}^{(\ell)}\, \mathsf{A}^{(\ell-1)}\, \mathsf{V}^{(\ell)} + \mathsf{B}^{(\ell)}, \qquad \ell = 2,\, 3,\, \ldots,\, L, \tag{A.7}$$

where $\mathsf{A} = \mathsf{A}^{(L)}$, and the sequence $\mathsf{A}^{(L)}$, $\mathsf{A}^{(L-1)}$, $\ldots$, $\mathsf{A}^{(1)}$ consists of matrices that are successively

smaller (typically, $\mathsf{A}^{(\ell-1)}$ is roughly half the size of $\mathsf{A}^{(\ell)}$). In (A.7), the matrices $\mathsf{U}^{(\ell)}$, $\mathsf{V}^{(\ell)}$ and $\mathsf{B}^{(\ell)}$ are all block-diagonal, so the formula directly leads to a fast technique for evaluating a matrix-vector product. The HSS property is similar to many other data-sparse representations in that it exploits rank-deficiencies in off-diagonal blocks to allow matrix-vector products to be evaluated rapidly; the Fast Multipole Method [44, 45], Barnes-Hut [5], and panel clustering [48] are all similar in this regard. The HSS property is different from these other formats in that it also allows the rapid computation of a matrix inverse, of an LU factorization, *etc*, [15, 17, 25, 62, 72]. The ability to perform algebraic operations other than the matrix-vector multiplication is also characteristic of the $\mathcal{H}$-matrix format of Hackbusch [49].

Remark 35. *There currently is little consistency in terminology when it comes to "data-sparse" matrices. The property that we refer to as the "HSS" property has appeared under different names in, e.g., [62, 63, 66, 72]. It is closely related to the "$\mathcal{H}^2$-matrix" format [8, 9, 10, 50] which is more restrictive than the $\mathcal{H}$-matrix format, and often admits $O(N)$ algorithms.*

Remark 36. *This remark describes in which sense the off-diagonal blocks of a matrix that is compressible in the HSS-sense have low rank; it can safely be by-passed as the material here is referenced only briefly in Section A.3.3. Let A denote an $N \times N$ HSS matrix A. Let I denote an index vector*

$$I = [n+1,\, n+2,\, \ldots,\, n+m],$$

where n and m are positive integers such that $n + m \leq N$. Then we define the HSS row block R_I as the $m \times N$ matrix

$$\mathsf{R}_I = \left[\begin{array}{cccc|cccc|cccc} a_{n+1,1} & a_{n+1,2} & \cdots & a_{n+1,n} & 0 & 0 & \cdots & 0 & a_{n+1,n+m+1} & a_{n+1,n+m+2} & \cdots & a_{n+1,N} \\ a_{n+2,1} & a_{n+2,2} & \cdots & a_{n+2,n} & 0 & 0 & \cdots & 0 & a_{n+2,n+m+1} & a_{n+2,n+m+2} & \cdots & a_{n+2,N} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{n+m,1} & a_{n+m,2} & \cdots & a_{n+m,n} & 0 & 0 & \cdots & 0 & a_{n+m,n+m+1} & a_{n+m,n+m+2} & \cdots & a_{n+m,N} \end{array}\right]$$

In other words, R_I is an $m \times N$ sub-matrix of A corresponding to the rows marked by the index vector I, but with the diagonal block corresponding to I replaced by a zero matrix. The HSS column

block C_I *is analogously defined as the* $N \times m$ *matrix consisting of* m *columns of* A *with the diagonal block excised. The principal criterion for a matrix* A *to be compressible in the HSS sense is that its HSS blocks should have numerically low rank.*

A.3 Case study: A discrete Laplace equation on a square

In this section, we illustrate how the coarse-graining techniques outlined in Section A.1.4 can be applied to a discrete equation closely related to (A.1). This discrete equation can be viewed either as the result of discretizing (A.1) via a finite difference method, or as an equation that in its own right models, for instance, electro-statics on a discrete grid.

A.3.1 Problem formulation

Given a positive integer N_{side}, we let Ω denote the $N_{\text{side}} \times N_{\text{side}}$ square subset of $\mathbb{Z}^2$ given by

$$\Omega = \{m = (m_1, m_2) \in \mathbb{Z}^2 : 1 \leq m_1 \leq N_{\text{side}} \text{ and } 1 \leq m_2 \leq N_{\text{side}}\}. \tag{A.8}$$

Figure A.2(a) illustrates the definition. For a node $m \in \Omega$, we let $\mathbb{B}_m$ denote a list of of all nodes in Ω that directly connect to m. For instance, an interior node such as the node m shown in Figure A.2(b) would have the neighbor list

$$\mathbb{B}_m = \{m_{\text{s}}, m_{\text{e}}, m_{\text{n}}, m_{\text{w}}\},$$

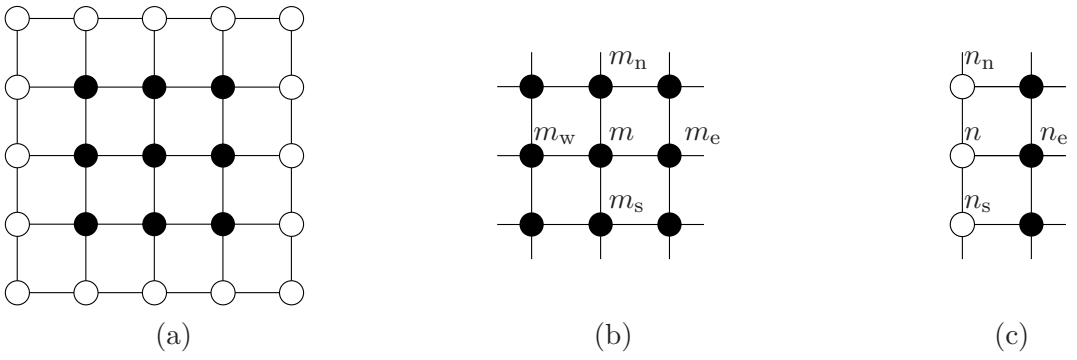


Figure A.2: Geometry of the lattice problem in Section A.3.1. (a) The full lattice for $N_{\text{side}} = 5$. The boundary nodes in Ω_{b} are white and the interior nodes in Ω_{i} are black. (b) The four neighbors of an interior node m. (c) The three neighbors of a boundary node n.

while a node on a "western" boundary like n in Figure A.2(c) would have the neighbor list

$$\mathbb{B}_n = \{n_{\mathrm{s}},\, n_{\mathrm{e}},\, n_{\mathrm{n}}\}.$$

For each pair $\{m,\, n\}$ of connected nodes, we let $\alpha_{m,n}$ denote a parameter indicating the *conductivity* of the link. For a function $u = u(m)$ where $m \in \Omega$, the *discrete Laplace operator* is then defined via

$$[\mathsf{A}\, u](m) = \sum_{n \in \mathbb{B}_m} \alpha_{m,n} \big[u(m) - u(n)\big]. \tag{A.9}$$

Example: For the case where $\alpha_{m,n} = 1$ for all connected nodes, we retrieve the standard five-point stencil associated with discretization of the Laplace operator. For instance, with column-wise ordering of the nodes in the lattice shown in Figure A.2(a), we obtain the 25×25 matrix

$$\mathsf{A} = \begin{bmatrix} \mathsf{C} & -\mathsf{I} & 0 & 0 & 0 \\ -\mathsf{I} & \mathsf{D} & -\mathsf{I} & 0 & 0 \\ 0 & -\mathsf{I} & \mathsf{D} & -\mathsf{I} & 0 \\ 0 & 0 & -\mathsf{I} & \mathsf{D} & -\mathsf{I} \\ 0 & 0 & 0 & -\mathsf{I} & \mathsf{C} \end{bmatrix}, \quad \text{where} \quad \mathsf{C} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 \\ 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}, \quad \text{where} \quad \mathsf{D} = \begin{bmatrix} 3 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 3 \end{bmatrix}, \tag{A.10}$$

and where I is the 5×5 identity matrix.

We let Ω_{b} denote the *boundary nodes* and we let Ω_{i} denote the *interior nodes* (*cf.* Figure A.2(a)). Partitioning the matrix A accordingly, the discrete analog of (A.1) becomes

$$\begin{bmatrix} \mathsf{A}_{\mathrm{b,b}} & \mathsf{A}_{\mathrm{b,i}} \\ \mathsf{A}_{\mathrm{i,b}} & \mathsf{A}_{\mathrm{i,i}} \end{bmatrix} \begin{bmatrix} u_{\mathrm{b}} \\ u_{\mathrm{i}} \end{bmatrix} = \begin{bmatrix} f_{\mathrm{b}} \\ 0 \end{bmatrix}. \tag{A.11}$$

Solving for the boundary values of the potential, u_{b}, we find that[1]

$$u_{\mathrm{b}} = \big(\mathsf{A}_{\mathrm{b,b}} - \mathsf{A}_{\mathrm{b,i}}\, \mathsf{A}_{\mathrm{i,i}}^{-1}\, \mathsf{A}_{\mathrm{i,b}}\big)^{-1} f_{\mathrm{b}}.$$

In consequence, the discrete analog of the solution operator (in this case a discrete analog of the

[1] Strictly speaking, the matrix $\mathsf{A}_{\mathrm{b,b}} - \mathsf{A}_{\mathrm{b,i}}\, \mathsf{A}_{\mathrm{i,i}}^{-1}\, \mathsf{A}_{\mathrm{i,b}}$ has a one-dimensional null-space formed by the constant functions and is not invertible. This is easily dealt with by a regularization that restricts attention to functions summing to zero. In what follows, such regularization will be employed where appropriate without further mention.

Neumann-to-Dirichlet operator) is

$$\mathsf{T} = \left(\mathsf{A}_{\mathrm{b,b}} - \mathsf{A}_{\mathrm{b,i}}\,\mathsf{A}_{\mathrm{i,i}}^{-1}\,\mathsf{A}_{\mathrm{i,b}}\right)^{-1}. \tag{A.12}$$

The operator T defined by (A.12) is dense, but turns out to be *data-sparse* in the sense described in Section A.2. We will in this section substantiate this claim via numerical examples, and also outline strategies for rapidly constructing such an operator in different environments.

A.3.2 Model problems

The compressibility of the solution operator T defined by (A.12) was investigated in the following five model environments:

Case A: Constant conductivities. In this model, all conductivities are identically one,

$$\alpha_{m,n} = 1 \qquad \text{for each connected pair } \{m,\, n\}. \tag{A.13}$$

For $N_{\mathrm{side}} = 5$, the resulting matrix A is the one given as an example in (A.10). Since in this case the matrix A can be viewed as a discretization of the Laplace operator $-\Delta$ on a square, the solution operator T can be viewed as a discrete analog of the standard Neumann-to-Dirichlet operator associated with Laplace's equation.

Case B: Smooth periodic conductivities. This case is a discrete analog of the equation

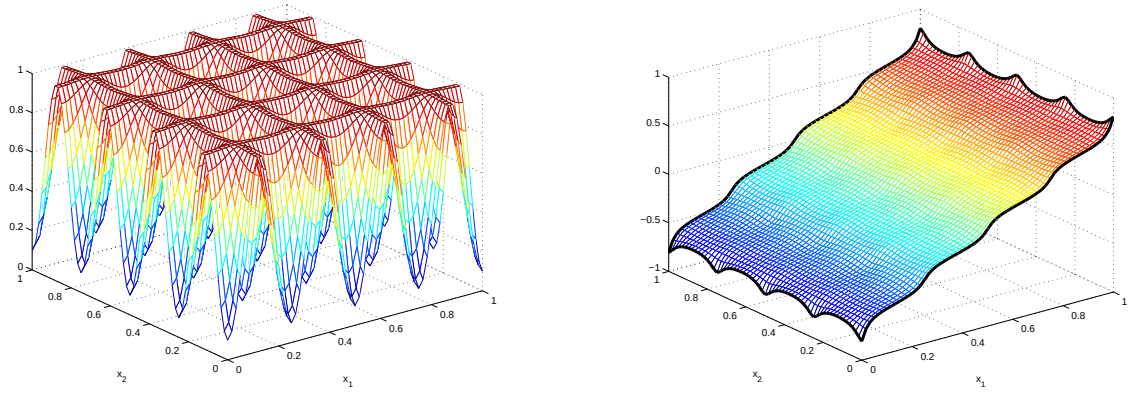
$$-\nabla \cdot \big(b(x)\nabla u(x)\big) = f(x), \qquad x \in [0,\, 1]^2, \tag{A.14}$$

where b is a periodic function defined by

$$b(x) = 1 - 0.9\left(\cos(\pi\, N_{\mathrm{cells}}\, x_1)\right)^2 \left(\cos(\pi\, N_{\mathrm{cells}}\, x_2)\right)^2, \qquad x = (x_1,\, x_2) \in [0,\, 1]^2. \tag{A.15}$$

In other words, (A.14) models a medium whose conductivity repeats periodically across $N_{\mathrm{cells}} \times N_{\mathrm{cells}}$ cells in the square $[0,\, 1]^2$. Figure A.3(a) illustrates the function b for $N_{\mathrm{cells}} = 4$. A discrete analog of (A.14) is now obtained by setting

$$\alpha_{m,n} = b\left(\frac{m+n-2}{2\,(N_{\mathrm{side}}-1)}\right) \qquad \text{for each connected pair } \{m,\, n\}.$$



(a) (b)

Figure A.3: The periodic problem described as Case B in Section A.3.2 with $N_{\text{cells}} = 4$ and $N_{\text{side}} = 61$. (a) The function $b = b(x)$ defined by (A.15). (b) A solution to the Neumann problem (A.11) with a constant inflow at $x_1 = 1$ and a constant outflow at $x_1 = 0$.

In our experiments, we chose N_{cell} so that 25 nodes were used to resolve each period, $N_{\text{cell}} = (N_{\text{side}} - 1)/25$ (for clarity, Figure A.3 shows a solution with only 15 nodes per period). In this case, the solutions are typically oscillatory on the boundary, *cf.* Figure A.3(b). This is a basic two-scale problem that should be amenable to traditional homogenization techniques provided there is a sufficient separation of length-scales.

Case C: Random conductivities. The conductivities $\alpha_{m,n}$ are for each connected pair of nodes $\{m, n\}$ drawn independently from a uniform probability distribution on [1, 2]. In this case, there is no local regularity, but we would expect traditional homogenization to give accurate results whenever the length-scales are sufficiently separated.

Case D: Sparsely distributed missing bars. In this model, all bars are assigned conductivity 1 (as in Case A), but then a small percentage p of bars are completely removed (in the examples reported, $p = 4\%$). In other words,

$$\alpha_{m,n} = \begin{cases} 1, & \text{with probability } 1-p \quad \text{if } \{m, n\} \text{ is a connected pair,} \\ 0, & \text{with probability } p \quad \text{if } \{m, n\} \text{ is a connected pair,} \\ 0, & \text{if } \{m, n\} \text{ is not a connected pair.} \end{cases}$$

As in Case C, there is no local regularity, but we would expect traditional homogenization to give

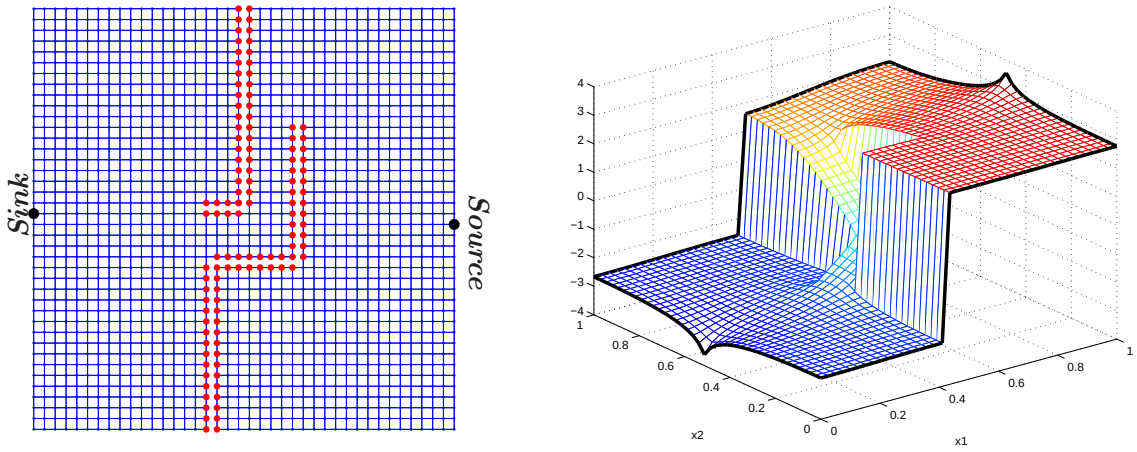
accurate results whenever the length-scales are sufficiently separated.

Case E: A lattice with two long cracks. This model is similar to Case D in that a small number of links have been cut, and all the remaining ones have unit conductivity. However, we organized the cut links into two long cracks running through the lattice. Figure A.4(a) illustrates for a case where $N_{\text{side}} = 50$. In larger lattices, the cracks have the same proportions, but the gap between the two cracks is kept constant at four links. In this case, solutions may exhibit major discontinuities. Figure A.4(b) illustrate the electric field resulting from placing oppositely signed unit sources at the locations marked *source* and *sink* in Figure A.4(a). We would expect analytic derivation of a simplified model to be very hard work in a situation such as this.

A.3.3 Compressibility of the solution operator

While the operator T defined by (A.12) is dense, it is in many situations of interest *data-sparse* in the sense described in Section A.2. To illustrate this point, we computed the matrix T by brute force for several different lattices, compressed it into the HSS format to ten digits of accuracy (we enforced that local truncation errors be less than 10^{-10}), and looked at how much memory was required to store the result. Tables A.1 and A.2 show our findings for each of the five different models described in Section A.3.2, and for differently sized lattices. To provide more detail, Table A.3 reports the average ranks of the so-called "HSS blocks" (as defined in Remark 36) of a $6\,396 \times 6\,396$ matrix T associated with a $1\,600 \times 1\,600$ square domain for each of the five examples.

An interesting aspect of the reported data is that the matrix T associated with the classical five-point stencil (represented by Case A) is highly compressible. To store it to ten digits of accuracy, less than 100 floating point numbers are required for each degree of freedom (see Table A.2). This fact has been exploited in a series of recent papers, including [18, 53, 58]. What is perhaps more remarkable is that the compressibility property is extremely robust to small changes in the micro-structure. As the tables A.1, A.2, and A.3 show, there is almost no discernible



(a) (b)

Figure A.4: (a) The lattice with cracks described as *Case E* in Section A.3.2 for $N_{\text{side}} = 40$. (b) A solution to the Neumann problem (A.11) with a unit inflow at the location marked *source* in (a), and a unit outflow at the location marked *sink*.

Memory requirements in KB

	N_{block}				
	100	200	400	800	1600
General matrix	1.23e3	4.95e3	1.99e4	7.98e4	3.20e5
Case A (constant conductivities)	3.02e2	6.13e2	1.22e3	2.42e3	4.78e3
Case B (periodic conductivities)	2.97e2	6.06e2	1.21e3	2.38e3	4.69e3
Case C (random conductivities)	3.03e2	6.20e2	1.23e3	2.43e3	4.80e3
Case D (random cuts)	2.96e2	6.06e2	1.20e3	2.38e3	4.70e3
Case E (cracks)	2.96e2	6.10e2	1.22e3	2.42e3	4.77e3

Table A.1: The table shows the amount of memory (in KB) required for storing the matrix T defined by (A.12) for different problem sizes N_{size}. The first line gives the memory required for storing a general dense matrix of size $4(N_{\text{side}}-1)\times 4(N_{\text{side}}-1)$. The following lines give the amount of memory required to store T in the "HSS" data-sparse format described in Section A.2 for each each of the five cases described in Section A.3.2, to within precision 10^{-10}.

Memory requirements in words per degree of freedom

	N_{block}				
	100	200	400	800	1600
General matrix	396	796	1596	3196	6396
Case A (constant conductivities)	97.7	98.6	98.1	96.8	95.7
Case B (periodic conductivities)	95.9	97.4	96.7	95.4	93.9
Case C (random conductivities)	97.8	99.7	98.8	97.5	96.0
Case D (random cuts)	95.5	97.5	96.6	95.4	94.1
Case E (cracks)	95.7	98.1	97.7	96.8	95.5

Table A.2: The table shows the same data given in Table A.1, but now scaled to demonstrate that the memory requirement scales linearly with problem size. To be precise, the entries given are the number of "words" (the memory required to store a floating point number to double precision accuracy) required per node on the boundary.

HSS ranks of the Schur complements for a matrix of size 6396×6396

	N_{block}					
	50	100	200	400	800	1600
General matrix	50	100	200	400	800	1600
Case A (constant conductivities)	19.3	22.7	26.0	31.0	39.0	53.0
Case B (periodic conductivities)	18.8	21.6	24.8	29.3	37.0	50.0
Case C (random conductivities)	19.3	22.8	26.8	31.6	39.8	54.0
Case D (random cuts)	18.7	21.9	25.5	30.8	38.8	52.5
Case E (cracks)	19.2	22.7	25.9	30.9	38.8	52.5

Table A.3: The table shows the HSS-ranks (as described in Remark 36) of blocks in the solution operator for the different models. The reported rank was the average numerical rank (at precision 10^{-10}) over all HSS blocks of size N_{block} that arise in the compressed representation.

difference in compressibility between the five models considered.

Once the compressed solution operator has been computed, it can be applied to a vector more or less instantaneously. For our simple implementation, we found the following for the time t_{solve} (in seconds) required for a single solve:

N_{side}	200	400	800	1600	3200
t_{solve} (sec)	4.4e-3	8.7e-3	1.8e-2	3.4e-2	7.1e-2

These numbers refer to a reduced model that is precise to within ten digits, and we would like to emphasize that the largest example reported, which requires 0.07 seconds for one solve, involves a problem whose micro-structure was originally resolved using $3\,200 \times 3\,200 \approx 10^7$ nodes.

The results reported in tables A.1, A.2, and A.3 indicate that reduced models that are precise to within ten digits of accuracy in principle exist, even in the presence of the following complications:

- Solutions that are oscillatory on the boundary, even when the period of the oscillation is not very much smaller than the size of the domain (as in Case B).
- Solutions that are highly irregular on the boundary (as in Cases C, D, and E).
- Boundary loads that exhibit no smoothness. (We observe that the solution operator is constructed under no assumption on smoothness of the boundary data.)
- Solutions that involve significant discontinuities (as shown in Figure A.4(b)).

In Sections A.3.4, A.3.5, and A.3.6, we will describe practical techniques for inexpensively computing such reduced models.

A.3.4 Techniques for computing the solution operator that fully resolve the micro-structure

Given a realization of a lattice model, the operator T defined by (A.12) can of course be computed with brute force. While Gaussian elimination has an $O(N_{\text{side}}^6)$ asymptotic cost that quickly becomes prohibitive, substantially more efficient techniques exist. Appendix A describes

a variation of the classical *nested dissection* method which in the present environment requires $O(N_{\text{side}}^3)$ floating point operations (flops) and $O(N_{\text{side}}^2)$ memory. This technique is exact up to rounding errors, and is very easy to implement. It was used to calculate the numbers reported in Section A.3.3 and is sufficiently fast that the solution operator associated with an 800×800 lattice can be determined in 40 seconds via a Matlab implementation running on a standard desktop PC.

More recently, techniques have been developed that compute an operator such as T in $O(N_{\text{side}}^2)$ time (or possibly $O(N_{\text{side}}^2(\log N_{\text{side}})^\kappa)$ for a small integer κ), which is optimal since there are $O(N_{\text{side}}^2)$ links in the lattice [18, 53, 58]. These techniques are highly efficient, and enable the brute force calculation of a reduced model in many important environments in both two and three dimensions. For a brief introduction, see Section A.6.3.

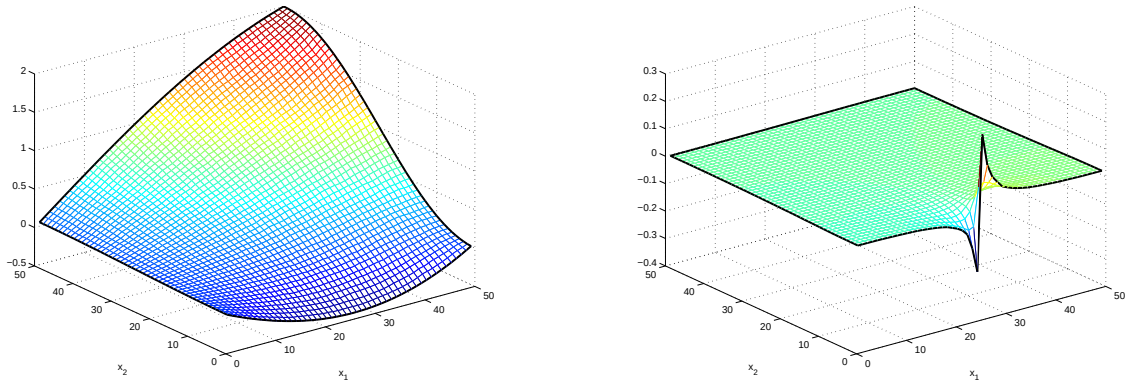
A.3.5 Techniques accelerated by collecting statistics from a representative volume element

In situations where there is a good separation of length-scales, variations of classical homogenization techniques can be used to dramatically accelerate the computation of a compressed boundary operator. To illustrate, let us investigate Case C in Section A.3.2 (the case of random conductivities, drawn uniformly from the interval [1, 2]). The most basic "homogenized equation" is in this case a lattice in which all links have the same conductivity. Through experiments on an RVE, we determined that this conductivity should be

$$c_3 = 1.4718\cdots$$

We let T_{hom} denote the solution operator (*i.e.* the lattice Neumann-to-Dirichlet operator) for the homogenized lattice. We measured the discrepancy between the homogenized operator T_{hom}, and the operator associated with the original lattice T, using the measures:

$$E_{\text{N2D}} = \frac{||\mathsf{T}_{\text{hom}} - \mathsf{T}||}{||\mathsf{T}||}, \qquad \text{and} \qquad E_{\text{D2N}} = \frac{||\mathsf{T}_{\text{hom}}^{-1} - \mathsf{T}^{-1}||}{||\mathsf{T}^{-1}||}. \tag{A.16}$$



(a) (b)

Figure A.5: Solutions for non-homogenized equation. (a) Solution resulting from the smooth boundary data f_{smooth}. (b) Solution resulting from the rough boundary data f_{rough}.

The first column of Table A.4 gives the results for a particular realization of a 50×50 lattice. In addition to the discrepancies measured in operator norm, the table also provides the errors

$$E_{\text{smooth}} = \frac{||\left(\mathsf{T}_{\text{hom}} - \mathsf{T}\right) f_{\text{smooth}}||}{||\mathsf{T}\, f_{\text{smooth}}||}, \qquad \text{and} \qquad E_{\text{rough}} = \frac{||\left(\mathsf{T}_{\text{hom}} - \mathsf{T}\right) f_{\text{rough}}||}{||\mathsf{T}\, f_{\text{rough}}||}, \tag{A.17}$$

associated with two particular Neumann data vectors f_{smooth} and f_{rough}. The solutions associated with these data vectors are shown in Figure A.5. These examples show that as one would expect, the homogenized equation provides quite high accuracy for a smooth solution, and very poor accuracy for a rough one. (Table A.4 also reports errors associated with improved "buffered" homogenization schemes, which will be introduced in Section A.3.6.)

We next repeated all experiments for Case D (as defined in Section A.3.2). In this case, numerical experiments indicated that the homogenized conductivity is

$$c_4 = 1 - \frac{1}{2}p + O(p^2).$$

The first column of Table A.5 shows the errors associated with a realization of "Case D" on a 50×50 grid, with $p = 0.04$, and $c_4 = 0.98$.

Remark 37 (Computational cost)**.** *The solution operator* T_{hom} *associated with a constant coefficient lattice can be computed in time proportional to* $O(N_{\text{side}})$ *(in other words, in time proportional*

Errors in homogenized operator for "Case C"

	Homogenization with no buffer	Homogenization with buffer of width b					
		$b = 1$	$b = 2$	$b = 3$	$b = 4$	$b = 5$	$b = 10$
E_{D2N}	1.9e-01	5.4e-03	1.2e-03	3.9e-04	3.3e-04	1.3e-04	6.6e-05
E_{N2D}	1.1e-02	7.5e-03	5.6e-03	5.7e-03	4.3e-03	4.9e-03	2.4e-03
E_{smooth}	7.3e-03	4.1e-03	4.1e-03	4.1e-03	2.8e-03	2.6e-03	1.4e-03
E_{rough}	1.5e-01	2.1e-02	1.1e-02	2.2e-03	8.8e-04	3.5e-03	9.2e-04

Table A.4: Discrepancy between the solution operator of an given lattice, and the homogenized solution operator. These numbers refer to the model described as "Case C" in Section A.3.2 (random conductivities). The errors E_{D2N}, E_{N2D}, E_{smooth}, and E_{rough} are defined in equations (A.16) and (A.17).

Errors in homogenized operator for "Case D"

	Homogenization with no buffer	Homogenization with buffer of width b					
		$b = 1$	$b = 2$	$b = 3$	$b = 4$	$b = 5$	$b = 10$
E_{D2N}	4.4e-01	1.5e-02	4.5e-03	1.7e-03	1.2e-03	7.6e-04	3.3e-04
E_{N2D}	8.7e-02	6.1e-02	5.6e-02	5.2e-02	4.5e-02	4.4e-02	2.8e-02
E_{smooth}	7.4e-02	5.9e-02	5.4e-02	4.8e-02	4.2e-02	4.1e-02	2.7e-02
E_{rough}	1.0e-01	7.0e-02	6.8e-02	6.2e-02	5.1e-02	5.0e-02	3.4e-02

Table A.5: Discrepancy between the solution operator of an given lattice, and the homogenized solution operator. These numbers refer to the model described as "Case D" in Section A.3.2 (randomly cut bars). The errors E_{D2N}, E_{N2D}, E_{smooth}, and E_{rough} are defined in equations (A.16) and (A.17).

to the number of nodes on the boundary). This means that very large lattices can be handled rapidly. It was demonstrated in [36] that the solution operator associated with a lattice with 10^{10} *nodes can be computed in less than two minutes on a standard desktop PC. (Observe that only the* $4 \cdot 10^5$ *nodes on the boundary actually need to enter the calculation.)*

A.3.6 Fusing a homogenized model to a locally fully resolved region

In the environments under consideration here, domains are loaded only on the border. This of course raises the possibility of improving the accuracy in the homogenized model by preserving the actual micro-structure in a thin strip along the boundary, and use the homogenized equations only in the interior. In the frame-work proposed here, where the simplified model consists of a solution operator rather than a differential operator (or in the present case, difference operator), it is extra ordinarily simple to do so.

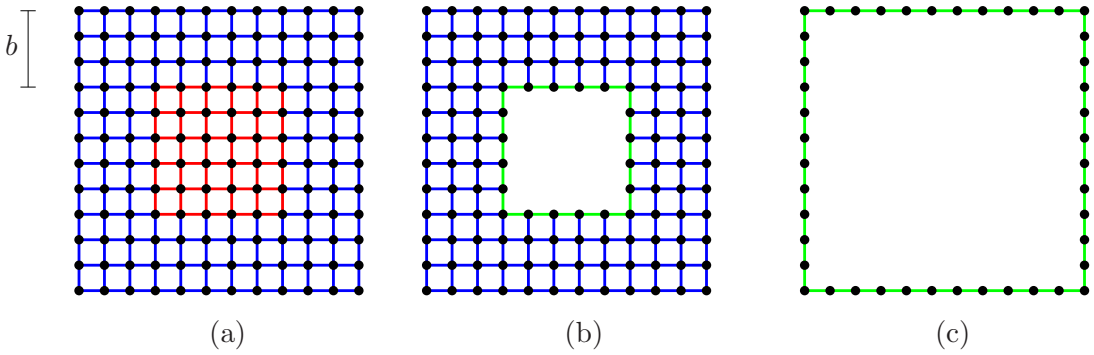


Figure A.6: Construction of a highly accurate reduced model by fusing a homogenized region with a region in which the micro-structure is fully resolved. (a) The blue links are within distance b of the boundary, and maintain their original conductivity. The red links are all assigned the "homogenized" conductivity. (b) All red links are eliminated from the model. This requires the construction of the solution operator for a constant coefficient lattice at cost $O(N_{\text{side}})$ (see Remark 37). (c) The few remaining links are eliminated to construct a highly approximate approximation to the solution operator.

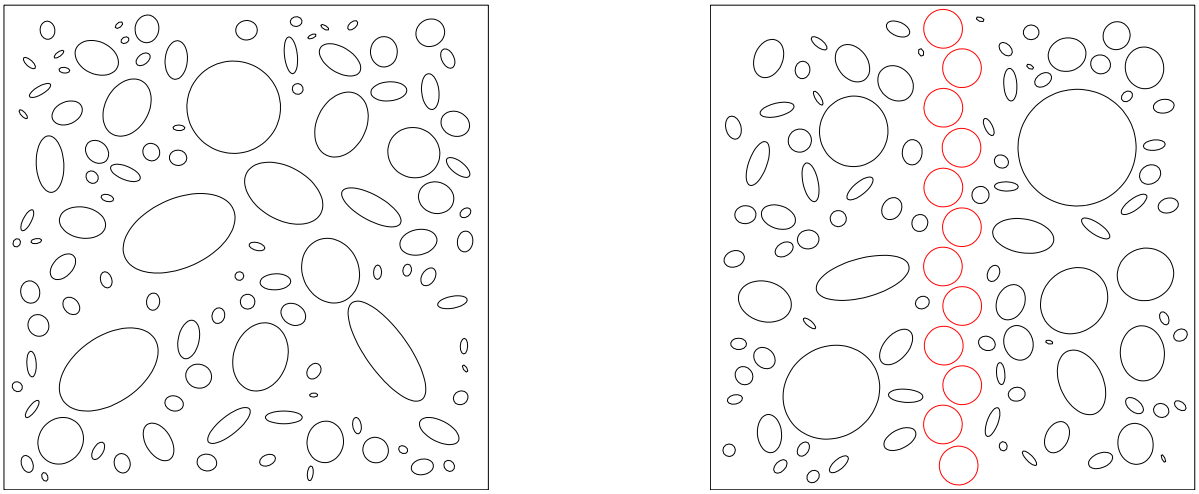
To illustrate, suppose that we are given a realization of an $N_{\text{side}} \times N_{\text{side}}$ lattice with heterogeneous conductivities. We fix a parameter b that indicates how broad of a band of cells we preserve, and then replace all bars that are more than b cells away from the boundary by bars

with the homogenized conductivity, as illustrated in Figure A.6(a). Then use the techniques of Section A.3.5 to compute the Neumann-to-Dirichlet operator for the constant coefficient lattice of size $(N_{\text{side}} - 2b) \times (N_{\text{side}} - 2b)$ in the center. As observed in Remark 37, the cost is only $O(N_{\text{side}})$, and the new reduced model involves only $O(N_{\text{side}})$ degrees of freedom. As Tables A.4 and A.5 demonstrate, for our model problems ("Case C" and "Case D") keeping only five layers of the original lattice leads to a reduced model that is accurate to three or four digits.

Remark 38 (Accuracy of Neumann vs. Dirichlet problems)**.** *Tables A.4 and A.5 show that when "unbuffered" homogenization is used, the resulting error E_{D2N} associated with Dirichlet problems is significantly larger than the error E_{N2D} associated with Neumann problems. The tables also show that the accuracy of Dirichlet problems improve dramatically upon the introduction of even a very thin boundary layer. This is as one would expect since the Dirichlet-to-Neumann operator is dominated by short range interactions.*

A.4 Case study: Two-phase media

In this section, we briefly investigate the compressibility of the Neumann-to-Dirichlet operator for a two-phase material modeled by equation (A.2). The two geometries we consider are shown in Figure A.7, with the conductivity of the inclusions set to zero. In this case, the operator under consideration is a boundary integral operator T supported on the square outer boundary. Using techniques described in Remark 39, we constructed an 1144×1144 matrix T that approximated T. With this number of nodes, any Neumann data generated by point sources up to a distance of 0.5% of the side length of the square can be resolved to eight digits of accuracy. We compressed the matrix T into the HSS format described in Section A.2 to a relative precision of 10^{-10}. The resulting data required 1.19KB of memory to store for the geometry shown in Figure A.7(a), and 1.22KB of memory for the geometry shown in Figure A.7(b). This corresponds to about 135 words of storage per row in the matrix. The HSS-ranks (as defined in Remark 36) are reported in Table A.6. We make three observations:



(a) (b)

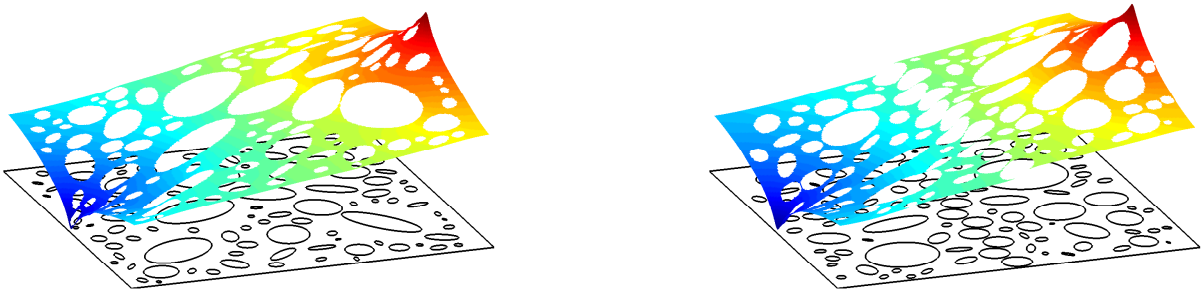
Figure A.7: Geometry for computations in Section A.4. (a) A perforated material. (b) A perforated material with a chain of holes that almost line up.

- A compressed version of the boundary operator can in this case be stored using about the same amount of memory (100 words per degree of freedom) as the operators associated with the discrete problems described in Section A.3.

- The two geometries shown in Figure A.7 require about the same amount of memory. This is note-worthy since the one labeled (b) corresponds to an almost singular geometry in which the domain is very close to being split in two halves. The effect is illustrated the solution shown in Figure A.8(b) where steep gradients are seen in middle of the piece. Standard assumptions used when homogenizing an elliptic differential operator are violated in this case.

- In Table A.6, the ranks of HSS-blocks of size 143 are *larger* than those of HSS-blocks of size 286. We speculate that this unusual situation can be traced to the fact that the larger blocks are larger than the inclusions, and largely do not "see" the heterogeneities.

Remark 39 (Details of computation)**.** *To derive our approximation to the Neumann-to-Dirichlet operator, we recast the Neumann Laplace equation (A.2) as a BIE defined on the joint boundary*



(a) (b)

Figure A.8: Solutions to the Laplace's equation with Neumann boundary conditions on the geometries (a) and (b) shown in Figure A.7. The boundary flux is set to be identically zero, except for two point sources of strengths ± 1.

Average ranks of HSS blocks for composite material example in Section A.4

	$N_{\text{block}} = 36$	$N_{\text{block}} = 71$	$N_{\text{block}} = 143$	$N_{\text{block}} = 286$
Geometry shown in Figure A.7(a)	18.2	27.0	39.5	25.8
Geometry shown in Figure A.7(b)	18.3	27.3	41.1	28.0

Table A.6: The average HSS-ranks (as defined in Remark 36) for the blocks in a data-sparse representation of the Neumann-to-Dirichlet operator for the geometries shown in Figure A.7.

$\Gamma \cup \Gamma_{\text{int}}$. *In the present case with non-conducting inclusions, the boundary condition on all interior boundaries simplifies to a homogeneous Neumann condition. We represented the solution as a single layer representation supported on both the outer boundary* Γ *and the interior boundary* Γ_{int}. *In other words, we sought a solution of the form*

$$u(x) = \int_{\Gamma} \log|x-y|\,\sigma(y)\,ds(y) + \int_{\Gamma_{\text{int}}} \log|x-y|\;\tau(y)\,ds(y). \tag{A.18}$$

The resulting BIE was discretized using a Nyström method combined with trapezoidal quadrature on the interior holes, and a Gaussian quadrature on the exterior boundary supported on 44 panels with 26 nodes each. The quadrature rule was locally modified as described in [12] to maintain eight digit accuracy in the presence of corners. This resulted in a large linear system from which all degrees of freedom associated with internal nodes (those associated with the density τ *in (A.18)) were eliminated. The resulting Schur complement was multiplied by a matrix representing evaluation of a single layer potential on the boundary to produce the final discrete approximation* T *to the "true" analytic Neumann-to-Dirichlet operator* T.

A.5 Generalizations

This report focused on problems modeled by simple Laplace-type problems in two dimensions involving no body loads. However, the techniques can be extended to much more general environments:

Other boundary conditions: While we focused on problems with Neumann boundary conditions, the extension to Dirichlet or mixed boundary conditions is trivial.

Other elliptic equations: The methods described extend readily to other elliptic equations whose kernels are non-oscillatory such as Stokes, elasticity, Yukawa, *etc.* The extension to wave problems modeled by Helmholtz equation, or the time-harmonic version of Maxwell, is more complicated for two reasons: (1) The presence of resonances (both true ones corresponding to the actual physics, and artificial ones present in the mathematical model only) must be dealt with. This can be done,

but requires careful attention. (2) As the wave-number increases, the compressibility of the solution operator deteriorates, and eventually renders the proposed approach wholly unaffordable.

Body loads: The extension to problems involving body loads is in principle straight-forward (see Remark 32). However, the compressed solution operator becomes more expensive to store.

Problems in three dimensions: In principle, the methodology proposed extends straightforwardly to problems in three dimensions. However, the construction of the solution operator does become more expensive, and the method might be best suited for environments where a precomputation is possible, or where the construction of the solution operator can be accelerated via the use of homogenized models in parts of the domain (as illustrated in Section A.3.6). Moreover, for problems in three dimensions involving body loads, memory requirements may become prohibitive.

A.6 Conclusions

The purpose of this report is to attempt to draw attention to recent developments in numerical analysis that could be very useful in modeling heterogeneous media. Specifically, it has become possible to inexpensively compute an approximation to the solution operator associated with many elliptic PDEs, and to perform various operations involving such solution operators: addition, multiplication, inversion, merging operators for different sub-domains, *etc.* We argue that such solution operators form excellent "reduced models" for many problems that have proven difficult to handle using traditional homogenization techniques.

Constructing reduced models by approximating the solution operator is particularly advantageous in the following environments:

Domains that are loaded on the boundary only: For problems that involve no body load, the solution operator is defined on the boundary only. This reduction in dimensionality means that once the operator is computed, it can be stored very efficiently, and applied to vectors sufficiently fast that real time simulations become possible. For some problems in this category, the actual

construction of the solution operator requires a large-scale (but very efficient) computation involving the entire micro-structure, but as shown in Section A.3.6, the solution operator can sometime be dramatically accelerated by using a homogenized model in the interior of the domain.

Situations where a pre-computation is possible: When the entire micro-structure needs to be resolved (as happens when the problem involves a body load, or a micro-structure not suitable for homogenization methods), the initial construction of the solution operator can become somewhat expensive, in particular for problems in three dimensions. However, once it has been constructed, it can usually be applied to a vector very rapidly. This raises the possibility of pre-computing a library of compressed models which can then be used as building blocks in computational simulations.

Problems in two dimensions (whether involving volume loads or not): Given current trends in algorithmic and hardware development, we predict that for a great many problems in two dimensions, it will soon become entirely affordable to resolve the entire micro-structure, and computationally derive a reduced model of the solution operator. The automatic nature of such a procedure would save much human effort, and would be very robust in the sense that the computed model would be guaranteed to be accurate to whichever tolerance was requested.

Appendix A: Efficient computation of the Neumann-to-Dirichlet operator

In this appendix, we describe an efficient technique for computing the Neumann-to-Dirichlet operator T defined by (A.12). It is a variation of the classical *nested dissection* techniques [35]. Throughout the appendix, Ω is a rectangular lattice, as defined by (A.8), and A is an associated discrete Laplace operator, as defined by (A.9).

To be precise, the technique we will describe does not compute the Neumann-to-Dirichlet operator T, but rather the *Schur complement* S, defined via

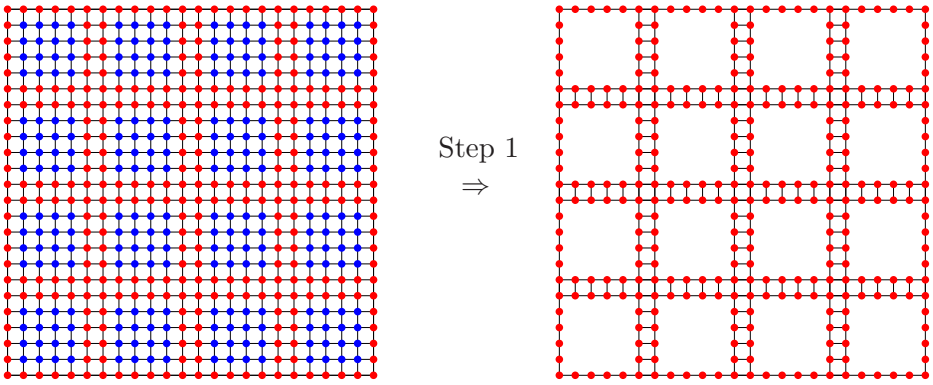
$$\mathsf{S} = \mathsf{A}_{\mathrm{b,b}} - \mathsf{A}_{\mathrm{b,i}}\,\mathsf{A}_{\mathrm{i,i}}^{-1}\,\mathsf{A}_{\mathrm{i,b}}. \tag{A.19}$$

Comparing (A.12) and (A.19), we see that $\mathsf{T} = \mathsf{S}^{-1}$.

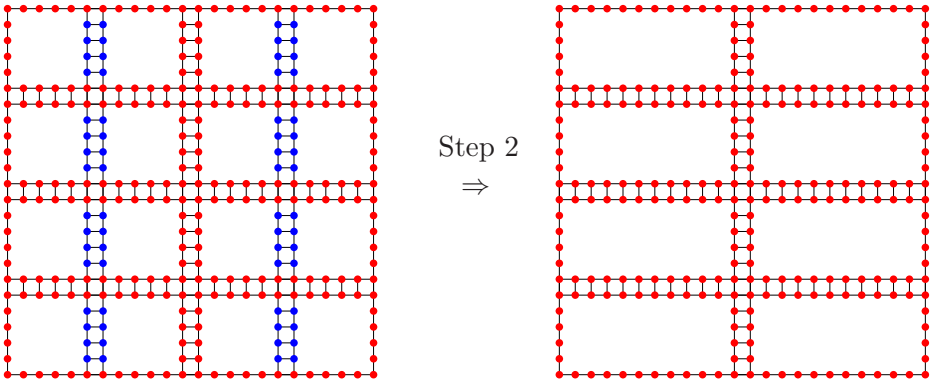
A.6.1 Outline

The technique is a divide-and-conquer scheme in which the computational domain Ω is first split into $2^L \times 2^L$ roughly equisized small boxes. The parameter L is chosen so that each of the small boxes is sufficiently small that its Schur complement can be computed by evaluating (A.19) via brute force. (In practice, we found that letting the smallest boxes be of size roughly 50×50, or $L \approx \log_2(N_{\text{side}}/50)$, works well.) Then it turns out to be possible to merge the Schur complements of two small adjacent boxes to form the Schur complement of the larger box; the process is described in Section A.6.2. The scheme proceeds by continuing the merge process to form the Schur complements of larger and larger boxes until eventually the entire box Ω has been processed. To illustrate, we describe the process graphically for a 24×24 domain that is originally split into 4×4 boxes, each containing 6×6 nodes.

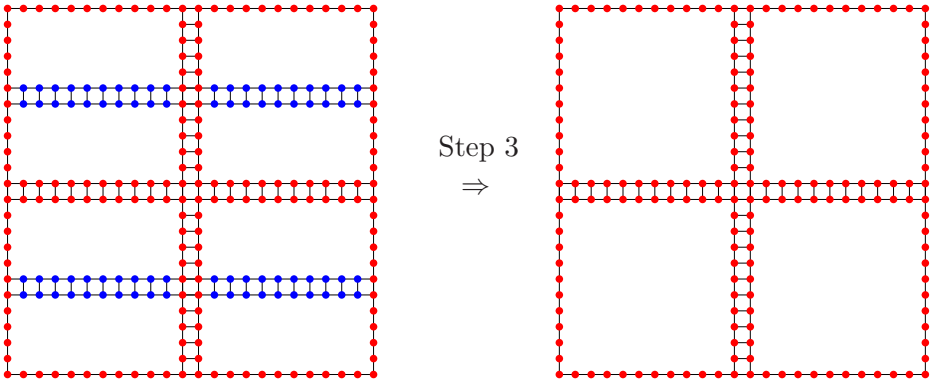
Step 1: Partition the box Ω into 16 small boxes. For each box, identify the internal nodes (marked in blue) and eliminate them using formula (A.19).



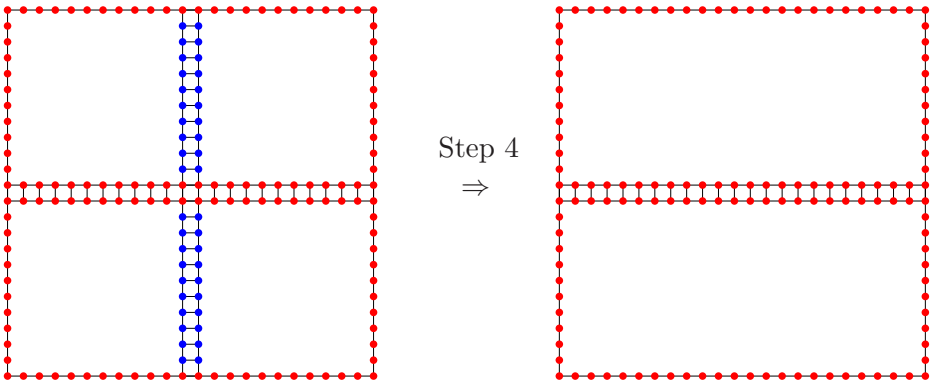
Step 2: Join the small boxes by pairs to form the Schur complements of boxes holding twice the number of nodes via the process to be described in Section A.6.2. The effect is to eliminate the interior nodes (marked in blue) of the newly formed larger boxes.



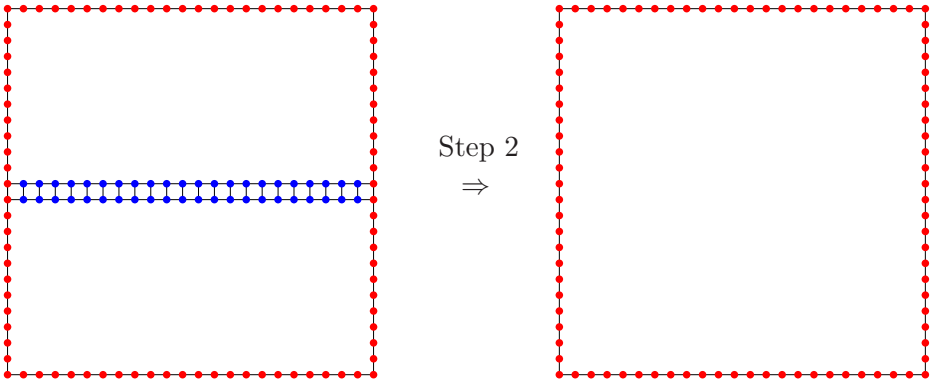
Step 3: Merge the boxes created in Step 2 in pairs, again via the process described in Section A.6.2.



Step 4: Repeat the merge process once more.

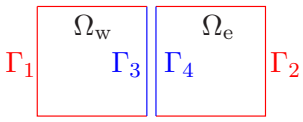


Step 5: Repeat the merge process one final time to obtain the Schur complement associated with the top level box Ω.



A.6.2 Merging of two Schur complements

Suppose that Ω is a box consisting of the two smaller boxes Ω_w and Ω_e (as in west and east):



Suppose further that we know the corresponding Schur complements S_w and S_e and seek the Schur complement S of Ω. In effect, we need to remove the "interior" points along the middle lines (marked in blue in the figure).

First partition the nodes in Γ_w into the subsets Γ_1 and Γ_3, and partition Γ_e into Γ_2 and Γ_4 as shown in the figure. The Schur complements S_w and S_e are partitioned accordingly,

$$\mathsf{S}_\mathrm{w} = \begin{bmatrix} \mathsf{S}_{11} & \mathsf{S}_{13} \\ \mathsf{S}_{31} & \mathsf{S}_{33} \end{bmatrix}, \qquad \text{and} \qquad \mathsf{S}_\mathrm{e} = \begin{bmatrix} \mathsf{S}_{22} & \mathsf{S}_{24} \\ \mathsf{S}_{42} & \mathsf{S}_{44} \end{bmatrix}.$$

Since the interior edges are unloaded, the joint equilibrium equation for the two boxes now reads

$$\left[\begin{array}{cc|cc} \mathsf{S}_{11} & \mathsf{A}_{12} & \mathsf{S}_{13} & 0 \\ \mathsf{A}_{21} & \mathsf{S}_{22} & 0 & \mathsf{S}_{24} \\ \hline \mathsf{S}_{31} & 0 & \mathsf{S}_{33} & \mathsf{A}_{34} \\ 0 & \mathsf{S}_{24} & \mathsf{A}_{43} & \mathsf{S}_{44} \end{array}\right] \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ 0 \\ 0 \end{bmatrix}, \tag{A.20}$$

where A_{ij} are the relevant submatrices of the original discrete Laplacian A. To be precise, with A denoting the global discrete Laplace operator, and with J_i denoting an index vector marking the

nodes in Γ_i, we have $\mathsf{A}_{ij} = \mathsf{A}(J_i, J_j)$. We observe that all matrices A_{ij} are very sparse (indeed, A_{12} and A_{21} have only two non-zero elements each). From (A.20), it is clear that the Schur complement of the large box is

$$\mathsf{S} = \begin{bmatrix} \mathsf{S}_{11} & \mathsf{A}_{12} \\ \mathsf{A}_{21} & \mathsf{S}_{22} \end{bmatrix} - \begin{bmatrix} \mathsf{S}_{13} & 0 \\ 0 & \mathsf{S}_{24} \end{bmatrix} \begin{bmatrix} \mathsf{S}_{33} & \mathsf{A}_{34} \\ \mathsf{A}_{43} & \mathsf{S}_{44} \end{bmatrix}^{-1} \begin{bmatrix} \mathsf{S}_{31} & 0 \\ 0 & \mathsf{S}_{42} \end{bmatrix}. \tag{A.21}$$

A.6.3 Accelerations

The scheme described in Section A.6.1 and A.6.2 requires $O(N_{\text{side}}^3)$ floating point operations, and $O(N_{\text{side}}^2)$ storage, just like the original nested dissection scheme. This cost is incurred by the repeated evaluation of the formula (A.21) which involve matrices S_{ij} that are dense. However, as discussed at length in Section A.3.3, these matrices have internal structure that allows operations such as matrix inversion, and matrix-matrix multiplication, to be evaluated in linear time. Incorporating such accelerated procedures reduces the overall cost (both floating point operations and memory) of the scheme to $O(N_{\text{side}}(\log N_{\text{side}})^\kappa)$. For recent work in this direction, see, *e.g.* [18, 53, 58].

Remark 40. *The process described in Section A.6.1 requires all Schur complements associated with one level to be kept in memory at one time. It is straight-forward to change the order in which the boxes are processed so that at most four Schur complements on each level must be kept in memory. When dense linear algebra is used, either approach requires $O(N_{\text{side}}^2)$ memory, but when data-sparse matrix formats are used, such an ordering reduces the memory requirement from $O(N_{\text{side}}^2)$ to $O(N_{\text{side}}(\log N_{\text{side}})^\kappa)$.*

Remark 41. *Even without accelerations, the scheme described in Section A.6.1 can handle moderate size problems quite efficiently. For a rudimentary implementation in Matlab executed on a standard desktop (with an Intel i7 CPU running at 2.67 GHz), the time t required to compute T was:*

N_{side}	*100*	*200*	*400*	*800*	*1600*	*3200*
t (sec)	*2.6e-1*	*1.2e0*	*6.4e0*	*4.5e1*	*5.0e2*	*6.7e3*

Note that less than a minute is required to process a lattice involving $800^2 = 640\,000$ *nodes.*

Appendix B

Efficient storage of Schur complement matrices

The accelerated nested dissection method presented in Chapter 3 utilizes the HSS properties of the Schur complement or boundary-to-boundary operators to achieve linear complexity. In this appendix, we describe a technique for storing the Schur complement matrices in such a way that the acceleration techniques presented in Section 3.4 are easily executed. We also present additional HSS algebra techniques that allow for efficient processing of the Schur complement matrices between merge steps in the fast direct solver.

B.1 Storage of the Schur complements

Recall that at each step in direct solver, block matrices are first excised from the Schur complements of small boxes in order to build the Schur complement for a larger box. The process of excising submatrices from an HSS factorized matrix is computationally expensive. Instead we choose to store each Schur complement in the following block format:

$$\mathsf{S} = \begin{bmatrix} \mathsf{S}_{ss} & \mathsf{S}_{se} & \mathsf{S}_{sn} & \mathsf{S}_{sw} & \mathsf{S}_{sc} \\ \mathsf{S}_{es} & \mathsf{S}_{ee} & \mathsf{S}_{en} & \mathsf{S}_{ew} & \mathsf{S}_{ec} \\ \mathsf{S}_{ns} & \mathsf{S}_{ne} & \mathsf{S}_{nn} & \mathsf{S}_{nw} & \mathsf{S}_{nc} \\ \mathsf{S}_{ws} & \mathsf{S}_{we} & \mathsf{S}_{wn} & \mathsf{S}_{ww} & \mathsf{S}_{wc} \\ \mathsf{S}_{cs} & \mathsf{S}_{ce} & \mathsf{S}_{cn} & \mathsf{S}_{cw} & \mathsf{S}_{cc} \end{bmatrix}, \tag{B.1}$$

where S_{ss} denotes the block matrix that describes the Schur complement interaction of the south edge with itself, etc. Figure B.1 illustrated the labeling of the local box. The corners are ordered

counter clockwise starting from the southwest corner, ie $c = \{c_{sw}, c_{se}, c_{ne}, c_{nw}\}$.

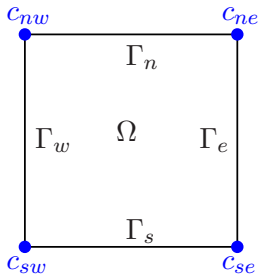


Figure B.1: Labeling of nodes for one box.

The diagonal blocks S_{ss}, S_{ee}, S_{nn}, and S_{ww} are stored in HSS factorized form and the off-diagonal blocks in (B.1) are stored in low-rank factorized form. While the blocks describing Schur complement interactions involving the corners are stored in dense form.

B.2 Review of the merge-two process

The process of returning the Schur complement built during a merge step to the form (B.1) requires additional HSS algebra. For illustrative purpose, suppose that the Schur complements S^1 and S^2 are given on Ω_1 and Ω_2 respectively, illustrated in Figure B.2. We seek to compute the Schur complement on the boundary of $\Omega = \Omega_1 \cup \Omega_2$.

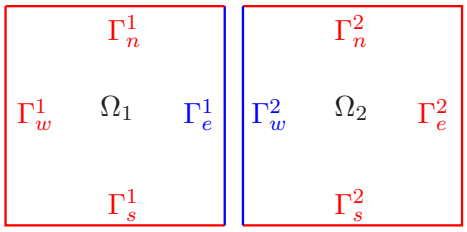


Figure B.2: Labeling of nodes when merging two boxes.

From Chapter 3, we know the Schur complement S is given by the formula:

$$\mathsf{S} = \begin{bmatrix} \mathsf{S}_{\text{ext}^1} & \mathsf{A}_{\text{ext}^1,\text{ext}^2} \\ \mathsf{A}_{\text{ext}^2,\text{ext}^1} & \mathsf{S}_{\text{ext}^2} \end{bmatrix} - \begin{bmatrix} \mathsf{S}_{\text{ext}^1,e^2} & 0 \\ 0 & \mathsf{S}_{\text{ext}^2,w^2} \end{bmatrix} \begin{bmatrix} \mathsf{S}_{e^1e^1} & \mathsf{A}_{e^1,w^2} \\ \mathsf{A}_{w^2,e^1} & \mathsf{S}_{w^2w^2} \end{bmatrix}^{-1} \begin{bmatrix} \mathsf{S}_{e^1,\text{ext}^1} & 0 \\ 0 & \mathsf{S}_{w^2,\text{ext}^2} \end{bmatrix}, \tag{B.2}$$

where the exterior of Ω_1 (denoted ext^1) is the set of points on $\Gamma_s^1 \cup \Gamma_n^1 \cup \Gamma_w^1 \cup c^1$, and the exterior of Ω_2 (denoted ext^2) is the set of points on $\Gamma_s^2 \cup \Gamma_w^2 \cup \Gamma_n^2 \cup c^2$.

Thus, using the block notation in (B.1), the matrices in (B.2) can be expanded. For example,

$$\mathsf{S}_{\text{ext}^1} = \begin{bmatrix} \mathsf{S}_{ss}^1 & \mathsf{S}_{sn}^1 & \mathsf{S}_{sw}^1 & \mathsf{S}_{sc}^1 \\ \mathsf{S}_{ns}^1 & \mathsf{S}_{nn}^1 & \mathsf{S}_{nw}^1 & \mathsf{S}_{nc}^1 \\ \mathsf{S}_{ws}^1 & \mathsf{S}_{wn}^1 & \mathsf{S}_{ww}^1 & \mathsf{S}_{wc}^1 \\ \mathsf{S}_{cs}^1 & \mathsf{S}_{cn}^1 & \mathsf{S}_{cw}^1 & \mathsf{S}_{cc}^1 \end{bmatrix}, \mathsf{S}_{\text{ext}^2} = \begin{bmatrix} \mathsf{S}_{ss}^2 & \mathsf{S}_{se}^2 & \mathsf{S}_{sn}^2 & \mathsf{S}_{sc}^2 \\ \mathsf{S}_{es}^2 & \mathsf{S}_{ee}^2 & \mathsf{S}_{en}^2 & \mathsf{S}_{ec}^2 \\ \mathsf{S}_{ns}^2 & \mathsf{S}_{ne}^2 & \mathsf{S}_{nn}^2 & \mathsf{S}_{nc}^2 \\ \mathsf{S}_{cs}^2 & \mathsf{S}_{ce}^2 & \mathsf{S}_{cn}^2 & \mathsf{S}_{cc}^2 \end{bmatrix}, \mathsf{A}_{e^1w^2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathsf{A}_{c^1,c^2} \end{bmatrix},$$

$$\mathsf{A}_{w^2e^1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathsf{A}_{c^2,c^1} \end{bmatrix}, \mathsf{S}_{\text{ext}^1e} = \begin{bmatrix} \mathsf{S}_{se}^1 \\ \mathsf{S}_{ne}^1 \\ \mathsf{S}_{we}^1 \\ \mathsf{S}_{ce}^1 \end{bmatrix}, \text{ etc.}$$

The techniques described in Section 3.4 in Chapter 3 compute the diagonal blocks of S corresponding the self-interactions of south, north and west edges of Ω_1 and the south, east, and north edges of Ω_2. All other blocks are either low-rank or small dense matrices and can be computed using standard techniques.

B.3 Post processing the Schur complement

In this section, we describe techniques for transforming S into the block form (B.1). Since the west edge of Ω_1 is also the west edge of Ω and no additional computations are required to build S_{ww}. Likewise, no additional computations are required to build S_{ee}. The off-diagonal blocks are easily constructed via classic techniques for low-rank matrices.

It remains to build S_{ss} and S_{nn} in HSS factorized form. Since the technique is the same for both edges, we chose to only present the construction of S_{ss}.

The south edge of Ω is $\Gamma_s = \Gamma_s^1 \cup \{c_{se}^1, c_{sw}^2\} \cup \Gamma_s^2$. Thus S_{ss} in block form is

$$\mathsf{S}_{ss} = \begin{bmatrix} \mathsf{S}_{s^1 s^1} & \mathsf{S}_{s^1 c_{se}^1} & \mathsf{S}_{s^1 c_{sw}^2} & \mathsf{S}_{s^1 s^2} \\ \mathsf{S}_{c_{se}^1 s^1} & \mathsf{S}_{c_{se}^1 c_{se}^1} & \mathsf{S}_{c_{se}^1 c_{sw}^2} & \mathsf{S}_{c_{se}^1 s^2} \\ \mathsf{S}_{c_{sw}^2 s^1} & \mathsf{S}_{c_{sw}^2 c_{se}^1} & \mathsf{S}_{c_{sw}^2 c_{sw}^2} & \mathsf{S}_{c_{sw}^2 s^2} \\ \mathsf{S}_{s^2 s^1} & \mathsf{S}_{s^2 c_{se}^1} & \mathsf{S}_{s^2 c_{sw}^2} & \mathsf{S}_{s^2 s^2} \end{bmatrix}. \tag{B.3}$$

The process of taking this matrix made up of a combination of HSS, dense and low-rank factorized matrices and returning one large HSS factorized matrix is a three step process. First, the HSS factorized matrix $\mathsf{S}_{s^1 s^1}$ is expanded in each direction by one via Algorithm 10 returning

$$\begin{bmatrix} \mathsf{S}_{s^1 s^1} & \mathsf{S}_{s^1 c_{se}^1} \\ \mathsf{S}_{c_{se}^1 s^1} & \mathsf{S}_{c_{se}^1 c_{se}^1} \end{bmatrix}$$

in HSS factorized form. Using a similar technique, the HSS factorized matrix $\mathsf{S}_{s^2 s^2}$ is expanded such that

$$\begin{bmatrix} \mathsf{S}_{c_{sw}^2 c_{sw}^2} & \mathsf{S}_{c_{sw}^2 s^2} \\ \mathsf{S}_{s^2 c_{sw}^2} & \mathsf{S}_{s^2 s^2} \end{bmatrix}$$

is in HSS factorized form. Then these two HSS factorized matrices along with low-rank factorized of the off-diagonal blocks of (B.3) are merged into one HSS factorized matrix S_{ss} via Algorithm 11.

ALGORITHM 10 (Expand an HSS matrix by one column and row)

Given an HSS factorized $N \times N$ matrix A, the coefficient vectors $\mathbf{r}$ and column $\mathbf{c}$ as well as the constant d, this algorithm expands A such that $\tilde{\mathsf{A}} = \begin{bmatrix} \mathsf{A} & \mathbf{c} \\ \mathbf{r} & d \end{bmatrix}$

loop over all levels, finer to coarser, $\ell = L,\ L-1,\ \ldots, 1$
  loop over all boxes τ on level ℓ,
    Let *ind* denote the indices associated with τ
    if τ is a leaf node
      if the last entry in *ind* is N

$$\tilde{\mathsf{D}}_\tau = \begin{bmatrix} \mathsf{D}_\tau & \mathbf{c}(ind) \\ \mathbf{r}(ind) & d \end{bmatrix}$$

$$\tilde{\mathsf{U}}_\tau = \begin{bmatrix} \mathsf{U}_\tau & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \tilde{\mathsf{V}}_\tau = \begin{bmatrix} \mathsf{V}_\tau & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$$

      else

$$\tilde{\mathsf{D}}_\tau = \mathsf{D}_\tau$$

$$\tilde{\mathsf{U}}_\tau = \begin{bmatrix} \mathsf{U}_\tau, \mathbf{r}(ind)^T \end{bmatrix} \qquad \tilde{\mathsf{V}}_\tau = [\mathsf{V}_\tau, \mathbf{c}(ind)]$$

      end if
    else
      Let σ_1 and σ_2 denote the children τ.
      if the last entry in *ind* is N

$$\tilde{\mathsf{U}}_\tau = \begin{bmatrix} \mathsf{U}_\tau^{\sigma_1} & \mathbf{0} \\ \mathbf{0} & 0 \\ \mathsf{U}_\tau^{\sigma_2} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \tilde{\mathsf{V}}_\tau = \begin{bmatrix} \mathsf{V}_\tau^{\sigma_1} & \mathbf{0} \\ \mathbf{0} & 0 \\ \mathsf{V}_\tau^{\sigma_2} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$$

$$\tilde{\mathsf{B}}_{\sigma_1} = \begin{bmatrix} \mathsf{B}_{\sigma_1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \tilde{\mathsf{B}}_{\sigma_2} = \begin{bmatrix} \mathsf{B}_{\sigma_2} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$$

      else

$$\tilde{\mathsf{U}}_\tau = \begin{bmatrix} \mathsf{U}_\tau^{\sigma_1} & \mathbf{0} \\ \mathbf{0} & 1 \\ \mathsf{U}_\tau^{\sigma_2} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \tilde{\mathsf{V}}_\tau = \begin{bmatrix} \mathsf{V}_\tau^{\sigma_1} & \mathbf{0} \\ \mathbf{0} & 1 \\ \mathsf{V}_\tau^{\sigma_2} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$$

$$\tilde{\mathsf{B}}_{\sigma_1} = \begin{bmatrix} \mathsf{B}_{\sigma_1} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \qquad \tilde{\mathsf{B}}_{\sigma_2} = \begin{bmatrix} \mathsf{B}_{\sigma_2} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix}$$

      end if
    end if
  end loop
end loop

$$\tilde{\mathsf{B}}_2 = \begin{bmatrix} \mathsf{B}_2 & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \tilde{\mathsf{B}}_3 = \begin{bmatrix} \mathsf{B}_3 & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$$

Recompress the matrix via Algorithm 9.
Note: $\mathsf{U}_\tau^{\sigma_1}$ denotes the restriction of U_τ acting on σ_1.

ALGORITHM 11 (Merge two HSS matrices into one large matrix)

Given two HSS factorized matrices A *and* B*, and QR-factorizations of the interaction matrices* $\mathsf{Q}_{AB}\,\mathsf{R}_{AB}$ *and* $\mathsf{Q}_{BA}\,\mathsf{R}_{BA}$*, this algorithm merges the matrices together such that* $\tilde{\mathsf{A}} = \begin{bmatrix} \mathsf{A} & \mathsf{Q}_{AB}\,\mathsf{R}_{AB} \\ \mathsf{Q}_{BA}\,\mathsf{R}_{BA} & \mathsf{B} \end{bmatrix}$

loop over all levels, finer to coarser, $\ell = L,\ L-1,\ \ldots, 2$
  loop over all boxes τ on level ℓ,
    if τ is a leaf node
    Let ind^o denote the indices from the original matrix A or B
    $\tilde{\mathsf{D}}_\tau = \mathsf{D}^o_\tau$
      if τ was originally in A
        $\mathsf{U}_{\text{tmp}} = [\mathsf{U}^o_\tau, \mathsf{Q}_{AB}(ind^o, :)]$ $\quad$ $\mathsf{V}_{\text{tmp}} = \left[\mathsf{V}^o_\tau, \mathsf{R}_{BA}(:, ind^o)^T\right]$
      else
        $\mathsf{U}_{\text{tmp}} = [\mathsf{U}_\tau, \mathsf{Q}_{BA}(ind^o, :)]$ $\quad$ $\mathsf{V}_{\text{tmp}} = \left[\mathsf{V}_\tau, \mathsf{R}_{AB}(:, ind^o)^T\right]$
      end if
    else
      Let σ_1 and σ_2 denote the children τ.
      Let $k_{\sigma_1} = \text{size}(\mathsf{U}_{\sigma_1}, 2)$ and $k_{\sigma_2} = \text{size}(\mathsf{U}_{\sigma_2}, 2)$.
      $\tilde{\mathsf{B}}_{\sigma_1} = \mathsf{P}_{\sigma_1}(:, 1\colon k_{\sigma_1})\mathsf{B}^o_{\sigma_1}\mathsf{W}_{\sigma_2}(:, 1\colon k_{\sigma_2})^T$
      $\tilde{\mathsf{B}}_{\sigma_2} = \mathsf{P}_{\sigma_2}(:, 1\colon k_{\sigma_2})\mathsf{B}^o_{\sigma_2}\mathsf{W}_{\sigma_1}(:, 1\colon k_{\sigma_1})^T$
      $\mathsf{U}_{\text{tmp1}} = \begin{bmatrix} \mathsf{P}_{\sigma_1}(:, 1\colon k_{\sigma_1}) & \mathsf{0} \\ \mathsf{0} & \mathsf{P}_{\sigma_2}(:, 1\colon k_{\sigma_2}) \end{bmatrix} \mathsf{U}_\tau$
      $\mathsf{V}_{\text{tmp1}} = \begin{bmatrix} \mathsf{W}_{\sigma_1}(:, 1\colon k_{\sigma_1}) & \mathsf{0} \\ \mathsf{0} & \mathsf{W}_{\sigma_2}(:, 1\colon k_{\sigma_2}) \end{bmatrix} \mathsf{V}_\tau$
      $\mathsf{U}_{\text{tmp2}} = \begin{bmatrix} \mathsf{P}_{\sigma_1}(:, k_{\sigma_1}+1\colon end) \\ \mathsf{P}_{\sigma_2}(:, k_{\sigma_2}+1\colon end) \end{bmatrix}$
      $\mathsf{V}_{\text{tmp2}} = \begin{bmatrix} \mathsf{W}_{\sigma_1}(:, k_{\sigma_1}+1\colon end) \\ \mathsf{W}_{\sigma_2}(:, k_{\sigma_2}+1\colon end) \end{bmatrix}$
      $\tilde{\mathsf{U}}_{\text{tmp}} = [\mathsf{U}_{\text{tmp1}}, \mathsf{U}_{\text{tmp2}}]$ $\quad$ $\tilde{\mathsf{V}}_{\text{tmp}} = [\mathsf{V}_{\text{tmp1}}, \mathsf{V}_{\text{tmp2}}]$
    end if
    $[\mathsf{U}_\tau, \mathsf{P}_\tau] = \text{rrqr}(U_{\text{tmp}})$ $\quad$ $[\mathsf{V}_\tau, \mathsf{W}_\tau] = \text{rrqr}(V_{\text{tmp}})$
  end loop
end loop
loop over boxes τ on level $\ell = 1$
  Let σ_1 and σ_2 denote the children τ.
  Let $k_{\sigma_1} = \text{size}(\mathsf{U}_{\sigma_1}, 2)$ and $k_{\sigma_2} = \text{size}(\mathsf{U}_{\sigma_2}, 2)$.
  $\tilde{\mathsf{B}}_{\sigma_1} = \mathsf{P}_{\sigma_1}(:, 1\colon k_{\sigma_1})\mathsf{B}^o_{\sigma_1}\mathsf{W}_{\sigma_2}(:, 1\colon k_{\sigma_2})^T$
  $\tilde{\mathsf{B}}_{\sigma_2} = \mathsf{P}_{\sigma_2}(:, 1\colon k_{\sigma_2})\mathsf{B}^o_{\sigma_2}\mathsf{W}_{\sigma_1}(:, 1\colon k_{\sigma_1})^T$
end loop
$\mathsf{U}_{\text{tmp}} = \begin{bmatrix} \mathsf{P}_4 \\ \mathsf{P}_5 \end{bmatrix} \begin{bmatrix} \mathsf{W}_6 \\ \mathsf{W}_7 \end{bmatrix}^T$ $\quad$ $\mathsf{V}_{\text{tmp}} = \begin{bmatrix} \mathsf{P}_6 \\ \mathsf{P}_7 \end{bmatrix} \begin{bmatrix} \mathsf{W}_4 \\ \mathsf{W}_5 \end{bmatrix}^T$
$\begin{bmatrix} \tilde{\mathsf{U}}_2, \mathsf{P}_2 \end{bmatrix} = \text{rrqr}(\mathsf{U}_{\text{tmp}})$ $\quad$ $\begin{bmatrix} \tilde{\mathsf{V}}_2, \mathsf{W}_2 \end{bmatrix} = \text{rrqr}(\mathsf{V}_{\text{tmp}})$
$\begin{bmatrix} \tilde{\mathsf{U}}_3, \mathsf{P}_3 \end{bmatrix} = \text{rrqr}(\mathsf{W}_2^T)$ $\quad$ $\begin{bmatrix} \tilde{\mathsf{V}}_3, \mathsf{W}_3 \end{bmatrix} = \text{rrqr}(\mathsf{P}_2)$
$\tilde{\mathsf{B}}_2 = \mathsf{W}_3^T$ $\quad$ $\tilde{\mathsf{B}}_3 = \mathsf{P}_3$

Note: B^o_σ denotes the B matrix corresponding to box σ in the HSS factorization of the matrix which has σ as a subset. For example, B^o_4 corresponds to B_2 in the HSS factorization of A.
The acronym rrqr corresponds to the rank revealing QR factorization algorithm.